# Ambari 1.7.0 Reference Guide

Hortonworks Data Platform
Ambari 1.7.0
2014-12-02

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.**

For more information on Hortonworks technology, Please visit the Hortonworks Data Platform page. 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

Ambari Reference Guide Installing Ambari Agents Manually ........ 6
- Download the Ambari Repo ........ 6
- Install the Ambari Agents Manually ........ 10

Customizing HDP Services ........ 13
- Customizing Services for a HDP 2.x Stack ........ 13
  - Defining Service Users and Groups for HDP 2.x ........ 13
  - Setting Properties That Depend on Service Usernames/Groups ........ 14
- Customizing Services for a HDP 1.x Stack ........ 15
  - Defining Service Users and Groups for HDP 1.x ........ 15
  - Setting Properties That Depend on Service Usernames/Groups ........ 16
  - Recommended Memory Configurations for the MapReduce Service ........ 17

Configuring LZO Compression ........ 18
- Configure core-site.xml for LZO ........ 18
- Running Compression with Hive Queries ........ 18
  - Create LZO Files ........ 18
  - Write Custom Java to Create LZO Files ........ 19

Using Non-Default Databases ........ 20
- Using Non-Default Databases - Ambari ........ 20
  - Using Ambari with Oracle ........ 20
  - Using Ambari with MySQL ........ 21
  - Using Ambari with PostgreSQL ........ 22
  - Troubleshooting Ambari ........ 23
- Using Non-Default Databases - Hive ........ 24
  - Using Hive with Oracle ........ 24
  - Using Hive with MySQL ........ 25
  - Using Hive with PostgreSQL ........ 27
  - Troubleshooting Hive ........ 28
- Using Non-Default Databases - Oozie ........ 30
  - Using Oozie with Oracle ........ 30
  - Using Oozie with MySQL ........ 30
  - Using Oozie with PostgreSQL ........ 31
  - Troubleshooting Oozie ........ 32

Setting up an Internet Proxy Server for Ambari ........ 34

How To Set Up an Internet Proxy Server for Ambari ..... 34
Configuring Network Port Numbers ..... 35
Default Network Port Numbers - Ambari ..... 35
Ganglia Ports ..... 36
Nagios Ports ..... 36
Optional: Changing the Default Ambari Server Port ..... 37
Changing the JDK Version on an Existing Cluster ..... 38
How to change the JDK Version for an Existing Cluster ..... 38
Configuring NameNode High Availability ..... 39
How To Set Up NameNode High Availability ..... 39
How to Roll Back NameNode HA ..... 45
Stop HBase ..... 45
Checkpoint the Active NameNode ..... 45
Stop All Services ..... 46
Prepare the Ambari Server Host for Rollback ..... 46
Restore the HBase Configuration ..... 47
Delete ZooKeeper Failover Controllers ..... 48
Modify HDFS Configurations ..... 49
Recreate the Standby NameNode ..... 51
Re-enable the Standby NameNode ..... 51
Delete All JournalNodes ..... 52
Delete the Additional NameNode ..... 52
Verify the HDFS Components ..... 53
Start HDFS ..... 53
Configuring ResourceManager High Availability ..... 54
How to Set Up ResourceManager High Availability ..... 54
Configuring RHEL HA for Hadoop 1.x ..... 56
Deploy the scripts ..... 56
Configure Ambari properties across the HA cluster ..... 56
Troubleshooting RHEL HA ..... 57
Using Ambari Blueprints ..... 59
Overview: Ambari Blueprints ..... 59
Configuring HDP Stack Repositories for Red Hat Satellite ..... 60
How To Configure HDP Stack Repositories for Red Hat Satellite ..... 60
Configuring Storm for Supervision ..... 61

How To Configure Storm for Supervision ... 61
Tuning Ambari Performance ... 64
How To Tune Ambari Performance ... 64
Refreshing YARN Capacity Scheduler ... 65
How to refresh the YARN Capacity Scheduler ... 65
Rebalancing HDFS ... 66
How to rebalance HDFS ... 66

# Ambari Reference Guide

# Installing Ambari Agents Manually

## Download the Ambari Repo

Select the OS family running on your installation host.

Follow instructions in the section for the operating system that runs on your installation host.
Use a command line editor to perform each instruction.

### RHEL/CentOS/Oracle Linux 6

1 Log in to your host as `root`. For example, type:

```
ssh <username>@<fqdn>
sudo su -
```

where `<username>` is your user name and `<fqdn>` is the fully qualified domain name of your server host.

2 Download the Ambari repository file to a directory on your installation host.

```
wget -nv http://public-repo-
1.hortonworks.com/ambari/centos6/1.x/updates/1.7.0/ambari.repo -O
/etc/yum.repos.d/ambari.repo
```

⚠ Do not modify the `ambari.repo` file name. This file is expected to be available on the Ambari Server host during Agent registration.

3 Confirm that the repository is configured by checking the repo list.

```
yum repolist
```

You should see values similar to the following for Ambari repositories in the list.

Version values vary, depending on the installation.

| repo id | repo name | status |
|---|---|---|
| AMBARI.1.7.0-1.x | Ambari 1.x | 5 |
| base | CentOS-6 - Base | 6,518 |
| extras | CentOS-6 - Extras | 15 |
| updates | CentOS-6 - Updates | 209 |

4 Install the Ambari bits. This also installs the default PostgreSQL Ambari database.

```
yum install ambari-server
```

5 Enter `y` when prompted to to confirm transaction and dependency checks.

A successful installation displays output similar to the following:

```
Installing : postgresql-libs-8.4.20-1.el6_5.x86_64
1/4
Installing : postgresql-8.4.20-1.el6_5.x86_64
2/4
Installing : postgresql-server-8.4.20-1.el6_5.x86_64
3/4
Installing : ambari-server-1.7.0-135.noarch
4/4
Verifying  : postgresql-server-8.4.20-1.el6_5.x86_64
1/4
Verifying  : postgresql-libs-8.4.20-1.el6_5.x86_64
2/4
Verifying  : ambari-server-1.7.0-135.noarch
3/4
Verifying  : postgresql-8.4.20-1.el6_5.x86_64
4/4

Installed:
  ambari-server.noarch 0:1.7.0-135

Dependency Installed:
  postgresql.x86_64 0:8.4.20-1.el6_5             postgresql-libs.x86_64
0:8.4.20-1.el6_5
  postgresql-server.x86_64 0:8.4.20-1.el6_5

Complete!
```

Accept the warning about trusting the Hortonworks GPG Key.  That key will be automatically downloaded and used to validate packages from Hortonworks. You will see the following message:

```
Importing GPG key 0x07513CAD:
 Userid: "Jenkins (HDP Builds) <jenkin@hortonworks.com>"
 From   :
http://s3.amazonaws.com/dev.hortonworks.com/ambari/centos6/RPM-GPG-
KEY/RPM-GPG-KEY-Jenkins
```

## SLES 11

1 Log in to your host as `root`.  For example, type:

```
ssh <username>@<fqdn>
sudo su -
```

where `<username>` is your user name and `<fqdn>` is the fully qualified domain name of your server host.

2 Download the Ambari repository file to a directory on your installation host.

```
wget -nv http://public-repo-
1.hortonworks.com/ambari/suse11/1.x/updates/1.7.0/ambari.repo -O
/etc/zypp/repos.d/ambari.repo
```

⚠ Do not modify the `ambari.repo` file name. This file is expected to be available on the Ambari Server host during Agent registration.

3 Confirm the downloaded repository is configured by checking the repo list.

```
zypper repos
```

You should see the Ambari repositories in the list.

Version values vary, depending on the installation.

| Alias | Name | Enabled | Refresh |
|---|---|---|---|
| AMBARI.1.7.0-1.x | Ambari 1.x | Yes | No |
| http-demeter.uni-regensburg.de-c997c8f9 | SUSE-Linux-Enterprise-Software-Development-Kit-11-SP1 11.1.1-1.57 | Yes | Yes |
| opensuse | OpenSuse | Yes | Yes |

4 Install the Ambari bits. This also installs PostgreSQL.

```
zypper install ambari-server
```

5 Enter `y` when prompted to to confirm transaction and dependency checks.

```
A successful installation displays output similar to the following:
Retrieving package postgresql-libs-8.3.5-1.12.x86_64 (1/4), 172.0 KiB
(571.0 KiB unpacked)
Retrieving: postgresql-libs-8.3.5-1.12.x86_64.rpm [done (47.3 KiB/s)]
Installing: postgresql-libs-8.3.5-1.12 [done]
Retrieving package postgresql-8.3.5-1.12.x86_64 (2/4), 1.0 MiB (4.2 MiB
unpacked)
Retrieving: postgresql-8.3.5-1.12.x86_64.rpm [done (148.8 KiB/s)]
Installing: postgresql-8.3.5-1.12 [done]
Retrieving package postgresql-server-8.3.5-1.12.x86_64 (3/4), 3.0 MiB
(12.6 MiB unpacked)
Retrieving: postgresql-server-8.3.5-1.12.x86_64.rpm [done (452.5
KiB/s)]
Installing: postgresql-server-8.3.5-1.12 [done]
Updating etc/sysconfig/postgresql...
Retrieving package ambari-server-1.7.0-135.noarch (4/4), 99.0 MiB
(126.3 MiB unpacked)
Retrieving: ambari-server-1.7.0-135.noarch.rpm [done (3.0 MiB/s)]
Installing: ambari-server-1.7.0-135 [done]
ambari-server             0:off  1:off  2:off  3:on   4:off  5:on
6:off
```

## UBUNTU 12

1 Log in to your host as `root`. For example, type:

```
ssh <username>@<fqdn>
sudo su -
```

where `<username>` is your user name and `<fqdn>` is the fully qualified domain name of your server host.

2 Download the Ambari repository file to a directory on your installation host.

```
wget -nv http://public-repo-
1.hortonworks.com/ambari/ubuntu12/1.x/updates/1.7.0/ambari.list -O
/etc/apt/sources.list.d/ambari.list

apt-key adv --recv-keys --keyserver keyserver.ubuntu.com
B9733A7A07513CAD

apt-get update
```

Do not modify the `ambari.list` file name. This file is expected to be available on the Ambari Server host during Agent registration.

3 Confirm that Ambari packages downloaded successfully by checking the package name list.

```
apt-cache pkgnames
```

You should see the Ambari packages in the list.

Version values vary, depending on the installation.

| Alias | Name |
|---|---|
| AMBARI-dev-2.x | Ambari 2.x |

4 Install the Ambari bits. This also installs PostgreSQL.

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

## RHEL/CentOS/ORACLE Linux 5 (DEPRECATED)

1 Log in to your host as `root`. For example, type:

```
ssh <username>@<fqdn>
sudo su -
```

where `<username>` is your user name and `<fqdn>` is the fully qualified domain name of your server host.

2 Download the Ambari repository file to a directory on your installation host.

```
wget -nv http://public-repo-
1.hortonworks.com/ambari/centos5/1.x/updates/1.7.0/ambari.repo -O
/etc/yum.repos.d/ambari.repo
```

Do not modify the `ambari.repo` file name. This file is expected to be available on the Ambari Server host during Agent registration.

3 Confirm the repository is configured by checking the repo list.

```
yum repolist
```

You should see the Ambari repositories in the list.

```
AMBARI.1.7.0-1.x                                    |  951 B     00:00
AMBARI.1.7.0-1.x/primary                       | 1.6 kB     00:00
AMBARI.1.7.0-1.x                                     5/5
epel                                                        | 3.7 kB     00:00
epel/primary_db                                     | 3.9 MB     00:01
```

| repo Id | repo Name | status |
|---|---|---|
| AMBARI.1.7.0-1.x | Ambari 1.x | 5 |
| base | CentOS-5 - Base | 3,667 |
| epel | Extra Packages for Enterprise Linux 5 - x86_64 | 7,614 |
| puppet | Puppet | 433 |
| updates | CentOS-5 - Updates | 118 |

4 Install the Ambari bits. This also installs PostgreSQL.

```
yum install ambari-server
```

When deploying HDP on a cluster having limited or no Internet access, you should provide access to the bits using an alternative method.

- For more information about setting up local repositories, see Optional: Configure Local Repositories.
- For more information about obtaining JCE policy archives for secure authentication, see Deploying JCE Policy Archives on the Ambari Server.

Ambari Server by default uses an embedded PostgreSQL database. When you install the Ambari Server, the PostgreSQL packages and dependencies must be available for install. These packages are typically available as part of your Operating System repositories. Please confirm you have the appropriate repositories available for the postgresql-server packages.

## Install the Ambari Agents Manually

Use the instructions specific to the OS family running on your agent hosts.

RHEL/CentOS/Oracle Linux 6

1 Install the Ambari Agent on every host in your cluster.

```
yum install ambari-agent
```

2. Using a text editor, configure the Ambari Agent by editing the `ambari-agent.ini` file as shown in the following example:

   ```
   vi /etc/ambari-agent/conf/ambari-agent.ini [server]
   hostname=<your.ambari.server.hostname>url_port=8440
   secured_url_port=8441
   ```

3. Start the agent on every host in your cluster.

   ```
   ambari-agent start
   ```

   The agent registers with the Server on start.

SLES 11

1. Install the Ambari Agent on every host in your cluster.

   ```
   yum install ambari-agent
   ```

2. Using a text editor, configure the Ambari Agent by editing the `ambari-agent.ini` file as shown in the following example:

   ```
   vi /etc/ambari-agent/conf/ambari-agent.ini [server]
   hostname=<your.ambari.server.hostname>url_port=8440
   secured_url_port=8441
   ```

3. Start the agent on every host in your cluster.

   ```
   ambari-agent start
   ```

   The agent registers with the Server on start.

UBUNTU 12

1. Install the Ambari Agent on every host in your cluster.

   ```
   yum install ambari-agent
   ```

2. Using a text editor, configure the Ambari Agent by editing the `ambari-agent.ini` file as shown in the following example:

   ```
   vi /etc/ambari-agent/conf/ambari-agent.ini [server]
   hostname=<your.ambari.server.hostname>url_port=8440
   secured_url_port=8441
   ```

3. Start the agent on every host in your cluster.

   ```
   ambari-agent start
   ```

   The agent registers with the Server on start.

RHEL/CentOS/Oracle Linux 5 (DEPRECATED)

1. Install the Ambari Agent on every host in your cluster.

   ```
   yum install ambari-agent
   ```

2. Using a text editor, configure the Ambari Agent by editing the `ambari-agent.ini` file as shown in the following example:

   ```
   vi /etc/ambari-agent/conf/ambari-agent.ini [server]
   hostname=<your.ambari.server.hostname>url_port=8440
   secured_url_port=8441
   ```

3. Start the agent on every host in your cluster.

   ```
   ambari-agent start
   ```

   The agent registers with the Server on start.

# Customizing HDP Services

You can override the default service settings established by the Ambari install wizard. For information about customizing service settings for your HDP Stack version, see one of the following topics:

- Customizing Services for a HDP 2.x Stack
- Customizing Services for a HDP 1.x Stack

## Customizing Services for a HDP 2.x Stack

Generally, you can customize services for the HDP 2.x Stack by overriding default settings that appear in **Services > Configs** for each Service in the Ambari Web GUI.

### Defining Service Users and Groups for HDP 2.x

The individual services in Hadoop run under the ownership of their respective Unix accounts. These accounts are known as service users. These service users belong to a special Unix group. "Smoke Test" is a service user dedicated specifically for running smoke tests on components during installation using the **Services** View of the Ambari Web GUI. You can also run service checks as the "Smoke Test" user on-demand after installation. You can customize any of these users and groups using the **Misc** tab during the **Customize Services** installation step.

Use the **Skip Group Modifications** option to not modify the Linux groups in the cluster. Choosing this option is typically required if your environment manages groups using LDAP and not on the local Linux machines.

If you choose to customize names, Ambari checks to see if these custom accounts already exist. If they do not exist, Ambari creates them. The default accounts are always created during installation whether or not custom accounts are specified. These default accounts are not used and can be removed post-install.

All new service user accounts, and any existing user accounts used as service users, must have a UID >= 1000.

*Table 1. Service Users*

| Service* | Component | Default User Account |
|---|---|---|
| Knox | Knox Gateway | knox |
| Kafka | Kafka Broker | kafka |
| HDFS | NameNode SecondaryNameNode DataNode | hdfs |
| YARN | NodeManager ResourceManager | yarn |
| MapReduce2 | HistoryServer | mapred |
| Tez | Tez clients | tez (Tez is available with HDP 2.1 or 2.2 Stack.) |
| HBase | MasterServer RegionServer | hbase |
| Hive | Hive Metastore, HiveServer2 | hive |

| | | |
|---|---|---|
| HCat | HCatalog Client | hcat |
| WebHCat | WebHCat Server | hcat |
| Falcon | Falcon Server | falcon<br>(Falcon is available with HDP 2.1 or 2.2 Stack.) |
| Storm | Masters (Nimbus, DRPC Server, Storm REST API, Server, Storm UI Server) Slaves (Supervisors, Logviewers) | storm<br>(Storm is available with HDP 2.1 or 2.2 Stack.) |
| Oozie | Oozie Server | oozie |
| Ganglia | Ganglia Server Ganglia Monitors | nobody |
| Ganglia | RRDTool (with Ganglia Server) | rrdcahed<br>(Created as part of installing RRDTool, which is used to store metrics data collected by Ganglia.) |
| Ganglia | Apache HTTP Server | apache<br>(Created as part of installing Apache HTTP Server, which is used to serve the Ganglia web UI.) |
| PostgreSQL | PostgreSQL (with Ambari Server) | postgres<br>(Created as part of installing the default PostgreSQL database with Ambari Server. If you are not using the Ambari PostgreSQL database, this user is not needed.) |
| Nagios | Nagios Server | nagios<br>(If you plan to use an existing user account named “nagios”, that “nagios” account must either be in a group named “nagios” or you must customize the Nagios Group.) |
| ZooKeeper | ZooKeeper | zookeeper |

*For all components, the Smoke Test user performs smoke tests against cluster services as part of the install process. It also can perform these on-demand, from the Ambari Web UI. The default user account for the smoke test user is ambari-qa.

*Table 2. Service Group*

| Service | Components | Default Group Account |
|---|---|---|
| All | All | hadoop |
| Nagios | Nagios Server | nagios |
| Ganglia | Ganglia Server Ganglia Monitor | nobody |
| Knox | Knox Gateway | knox |

## Setting Properties That Depend on Service Usernames/Groups

Some properties must be set to match specific service user names or service groups. If you have set up non-default, customized service user names for the HDFS or HBase service or the Hadoop group name, you must edit the following properties, using **Services > Service.Name > Configs > Advanced hdfs-ste:**

*Table 3. HDFS Settings: Advanced*

| Property Name | Value |
|---|---|
| dfs.permissions.superusergroup | The same as the HDFS username. The default is "hdfs" |
| dfs.cluster.administrators | A single space followed by the HDFS username. |
| dfs.block.local-path-access.user | The HBase username. The default is "hbase". |

*Table 4. MapReduce Settings: Advanced*

| Property Name | Value |
|---|---|
| mapreduce.cluster.administrators | A single space followed by the Hadoop group name. |

## Customizing Services for a HDP 1.x Stack

Generally, you can customize services for the HDP 1.x Stack by overriding default settings that appear in the Management Header for each Service in the Ambari Web GUI.

### Defining Service Users and Groups for HDP 1.x

The individual services in Hadoop run under the ownership of their respective Unix accounts. These accounts are known as service users. These service users belong to a special Unix group. "Smoke Test" is a service user dedicated specifically for running smoke tests on components during installation using the **Services** View of the Ambari Web GUI. You can also run service checks as the "Smoke Test" user on-demand after installation. You can customize any of these users and groups using the **Misc** tab during the **Customize Services** installation step.

Use the **Skip Group Modifications** option to not modify the Linux groups in the cluster. Choosing this option is typically required if your environment manages groups using LDAP and not on the local Linux machines.

If you choose to customize names, Ambari checks to see if these custom accounts already exist. If they do not exist, Ambari creates them. The default accounts are always created during installation whether or not custom accounts are specified. These default accounts are not used and can be removed post-install.

All new service user accounts, and any existing user accounts used as service users, must have a UID >= 1000.

*Table 5. Service Users*

| Service | Component | Default User Account |
|---|---|---|
| HDFS | NameNode SecondaryNameNode DataNode | hdfs |
| MapReduce | JobTracker HistoryServer TaskTracker | mapred |
| Hive | Hive Metastore HiveServer2 | hive |

| | | |
|---|---|---|
| HCat | HCatalog Server | hcat |
| WebHCat | WebHCat Server | hcat |
| Oozie | Oozie Server | oozie |
| HBase | MasterServer RegionServer | hbase |
| ZooKeeper | ZooKeeper | zookeeper |
| Ganglia | Ganglia Server Ganglia Collectors | nobody |
| Nagios | Nagios Server | nagios<br><br>(If you plan to use an existing user account named "nagios", that "nagios" account must be in a group named "nagios". If you customize this account, that account will be created and put in a group "nagios".) |
| Smoke Test | All | ambari-qa<br><br>(The Smoke Test user performs smoke tests against cluster services as part of the install process. It also can perform these on-demand from the Ambari Web GUI.) |

*Table 6. Service Group*

| Service | Components | Default Group Account |
|---|---|---|
| All | All | hadoop |

## Setting Properties That Depend on Service Usernames/Groups

Some properties must be set to match specific service user names or service groups. If you have set up non-default, customized service user names for the HDFS or HBase service or the Hadoop group name, you must edit the following properties, using **Services > Service.Name > Configs > Advanced**:

*Table 7. HDFS Settings: Advanced*

| Property Name | Value |
|---|---|
| dfs.permissions.supergroup | The same as the HDFS username. The default is "hdfs" |
| dfs.cluster.administrators | A single space followed by the HDFS username. |
| dfs.block.local-path-access.user | The HBase username. The default is "hbase". |

*Table 8. MapReduce Settings: Advanced*

| Property Name | Value |
|---|---|
| mapreduce.tasktracker.group | The Hadoop group name. The default is "hadoop". |
| mapreduce.cluster.administrators | A single space followed by the Hadoop group name. |

## Recommended Memory Configurations for the MapReduce Service

The following recommendations can help you determine appropriate memory configurations based on your usage scenario:

- Make sure that there is enough memory for all of the processes. Remember that system processes take around 10% of the available memory.
- For co-deploying an HBase RegionServer and MapReduce service on the same node, reduce the RegionServer's heap size (use the **HBase Settings > RegionServer > HBase Region Servers maximum Java heap size** property to modify the RegionServer heap size).
- For co-deploying an HBase RegionServer and the MapReduce service on the same node, or for memory intensive MapReduce applications, modify the map and reduce slots as suggested in the following example:

**EXAMPLE:** For co-deploying an HBase RegionServer and the MapReduce service on a machine with 16GB of available memory, the following would be a recommended configuration:

2 GB: system processes

8 GB: MapReduce slots. 6 Map + 2 Reduce slots per 1 GB task

4 GB: HBase RegionServer

1 GB: TaskTracker

1 GB: DataNode

To change the number of Map and Reduce slots based on the memory requirements of your application, use the following properties:

**MapReduce Settings: TaskTracker :** `Number of Map slots per node`

**MapReduce Settings: TaskTracker :** `Number of Reduce slots per node`

# Configuring LZO Compression

LZO is a lossless data compression library that favors speed over compression ratio. Ambari does not install nor enable LZO Compression by default.
To enable LZO compression in your HDP cluster, you must Configure core-site.xml for LZO.

Optionally, you can implement LZO to optimize Hive queries in your cluster for speed.
For more information about using LZO compression with Hive, see Running Compression with Hive Queries.

## Configure core-site.xml for LZO

1. Browse to **Ambari Web > Services > HDFS > Configs**, then expand **Advanced core-site.**
2. Find the `io.compression.codecs` property key.
3. Append to the `io.compression.codecs` property key, the following value: `com.hadoop.compression.lzo.LzoCodec,com.hadoop.compression.lzo.LzopCodec`
4. Add a description of the config modification, then choose Save.
5. Expand the `Custom core-site.xml` section.
6. Select **Add Property.**
7. Add to `Custom core-site.xml` the following property key and value

| Property Key | Property Value |
|---|---|
| io.compression.codec.lzo.class | com.hadoop.compression.lzo.LzoCodec |

8. Choose `Save`.
9. Add a description of the config modification, then choose Save.
10. Restart the HDFS, MapReduce2 and YARN services.

If performing a Restart or a Restart All does not start the required package install, you may need to stop, then start the HDFS service to install the necessary LZO packages. Restart is only available for a service in the "Runnning" or "Started" state.

## Running Compression with Hive Queries

Running Compression with Hive Queries requires creating LZO files. To create LZO files, use one of the following procedures:

### Create LZO Files

1. Create LZO files as the output of the Hive query.
2. Use `lzop` command utility or your custom Java to generate `lzo.index` for the `.lzo` files.

### Hive Query Parameters

Prefix the query string with these parameters:

```
SET
mapreduce.output.fileoutputformat.compress.codec=com.hadoop.compression.lzo.L
zopCodec
SET hive.exec.compress.output=true
SET mapreduce.output.fileoutputformat.compress=true
```

For example:

```
hive -e "SET
mapreduce.output.fileoutputformat.compress.codec=com.hadoop.compression.lzo.L
zopCodec;SET               hive.exec.compress.output=true;SET
mapreduce.output.fileoutputformat.compress=true;"
```

## Write Custom Java to Create LZO Files

1. Create text files as the output of the Hive query.
2. Write custom Java code to
    - convert Hive query generated text files to `.lzo` files
    - generate `lzo.index` files for the `.lzo` files

### Hive Query Parameters

Prefix the query string with these parameters:

```
SET hive.exec.compress.output=false
SET mapreduce.output.fileoutputformat.compress=false
```

For example:

```
hive -e "SET hive.exec.compress.output=false;SET
mapreduce.output.fileoutputformat.compress=false;<query-string>"
```

# Using Non-Default Databases

Use the following instructions to prepare a non-default database for Ambari, Hive/HCatalog, or Oozie. You must complete these instructions before you set up the Ambari Server by running `ambari-server setup`.

- Using Non-Default Databases - Ambari
- Using Non-Default Databases - Hive
- Using Non-Default Databases - Oozie

## Using Non-Default Databases - Ambari

The following sections describe how to use Ambari with an existing database, other than the embedded PostgreSQL database instance that Ambari Server uses by default.

- Using Ambari with Oracle
- Using Ambari with MySQL
- Using Ambari with PostgreSQL
- Troubleshooting Non-Default Databases with Ambari

### Using Ambari with Oracle

To set up Oracle for use with Ambari:

1 On the Ambari Server host, install the appropriate `JDBC.jar` file.

   1 Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.

   2 Select `Oracle Database 11g Release 2 - ojdbc6.jar`.

   3 Copy the .jar file to the Java share directory.

   ```
   cp ojdbc6.jar /usr/share/java
   ```

   4 Make sure the .jar file has the appropriate permissions - 644.

2 Create a user for Ambari and grant that user appropriate permissions.

   For example, using the Oracle database admin utility, run the following commands:

   ```
   # sqlplus sys/root as sysdba
   CREATE USER <AMBARIUSER> IDENTIFIED BY <AMBARIPASSWORD> default
   tablespace
   "USERS" temporary tablespace "TEMP";
   GRANT unlimited tablespace to <AMBARIUSER>;
   GRANT create session to <AMBARIUSER>;
   GRANT create TABLE to <AMBARIUSER>;
   GRANT create SEQUENCE to <AMBARIUSER>;
   QUIT;
   ```

   Where `<AMBARIUSER>` is the Ambari user name and `<AMBARIPASSWORD>` is the Ambari user password.

3 Load the Ambari Server database schema.

   1 You must pre-load the Ambari database schema into your Oracle database using the schema script.

   ```
   sqlplus <AMBARIUSER>/<AMBARIPASSWORD> < Ambari-DDL-Oracle-
   CREATE.sql
   ```

   2 Find the Ambari-DDL-Oracle-CREATE.sql file in the `/var/lib/ambari-server/resources/` directory of the Ambari Server host after you have installed Ambari Server.

4 When setting up the Ambari Server, select `Advanced Database Configuration > Option [2] Oracle` and respond to the prompts using the username/password credentials you created in step 2.

## Using Ambari with MySQL

To set up MySQL for use with Ambari:

1 On the Ambari Server host, install the connector.

   1 Install the connector

   **RHEL/CentOS/Oracle Linux**

   ```
   yum install mysql-connector-java
   ```

   **SLES**

   ```
   zypper install mysql-connector-java
   ```

   **UBUNTU**

   ```
   apt-get install mysql-connector-java
   ```

   2 Confirm that `.jar` is in the Java share directory.

   ```
   ls /usr/share/java/mysql-connector-java.jar
   ```

   3 Make sure the .jar file has the appropriate permissions - 644.

2 Create a user for Ambari and grant it permissions.

   • For example, using the MySQL database admin utility:

   ```
   # mysql -u root -p
   CREATE USER '<AMBARIUSER>'@'%' IDENTIFIED BY '<AMBARIPASSWORD>';
   GRANT ALL PRIVILEGES ON *.* TO '<AMBARIUSER>'@'%';
   CREATE USER '<AMBARIUSER>'@'localhost' IDENTIFIED BY
   '<AMBARIPASSWORD>';
   GRANT ALL PRIVILEGES ON *.* TO '<AMBARIUSER>'@'localhost';
   CREATE USER'<AMBARIUSER>'@'<AMBARISERVERFQDN>' IDENTIFIED BY
   '<AMBARIPASSWORD>';
   GRANT ALL PRIVILEGES ON *.* TO
   '<AMBARIUSER>'@'<AMBARISERVERFQDN>';
   FLUSH PRIVILEGES;
   ```

- Where <AMBARIUSER> is the Ambari user name, <AMBARIPASSWORD> is the Ambari user password and <AMBARISERVERFQDN> is the Fully Qualified Domain Name of the Ambari Server host.

3 Load the Ambari Server database schema.

- You must pre-load the Ambari database schema into your MySQL database using the schema script.

```
mysql -u <AMBARIUSER> -p CREATE DATABASE <AMBARIDATABASE>;
USE <AMBARIDATABASE>;
SOURCE Ambari-DDL-MySQL-CREATE.sql;
```

- Where <AMBARIUSER> is the Ambari user name and <AMBARIDATABASE> is the Ambari database name.

  Find the `Ambari-DDL-MySQL-CREATE.sql` file in the `/var/lib/ambari-server/resources/` directory of the Ambari Server host after you have installed Ambari Server.

4 When setting up the Ambari Server, select `Advanced Database Configuration > Option [3] MySQL` and enter the credentials you defined in Step 2. for user name, password and database name.

## Using Ambari with PostgreSQL

To set up PostgreSQL for use with Ambari:

1 Create a user for Ambari and grant it permissions.

- Using the PostgreSQL database admin utility:

```
# sudo -u postgres psql
CREATE DATABASE <AMBARIDATABASE>;
CREATE USER <AMBARIUSER> WITH PASSWORD '<AMBARIPASSWORD>';
GRANT ALL PRIVILEGES ON DATABASE <AMBARIDATABASE> TO
<AMBARIUSER>;
\connect <AMBARIDATABASE>;
CREATE SCHEMA <AMBARISCHEMA> AUTHORIZATION <AMBARIUSER>;
ALTER SCHEMA <AMBARISCHEMA> OWNER TO <AMBARIUSER>;
ALTER ROLE <AMBARIUSER> SET search_path to '<AMBARISCHEMA>',
'public';
```

- Where <AMBARIUSER> is the Ambari user name <AMBARIPASSWORD> is the Ambari user password, <AMBARIDATABASE> is the Ambari database name and <AMBARISCHEMA> is the Ambari schema name.

2 Load the Ambari Server database schema.

- You must pre-load the Ambari database schema into your PostgreSQL database using the schema script.

```
# psql -U <AMBARIUSER> -d <AMBARIDATABASE>
\connect <AMBARIDATABASE>;
\i Ambari-DDL-Postgres-CREATE.sql;
```

- Find the `Ambari-DDL-Postgres-CREATE.sql` file in the `/var/lib/ambari-server/resources/` directory of the Ambari Server host after you have installed Ambari Server.

3 When setting up the Ambari Server, select `Advanced Database Configuration > Option[4] PostgreSQL` and enter the credentials you defined in Step 2. for user name, password, and database name.

## Troubleshooting Ambari

Use these topics to help troubleshoot any issues you might have installing Ambari with an existing Oracle database.

### *Problem: Ambari Server Fails to Start: No Driver*

Check `/var/log/ambari-server/ambari-server.log` for the following error:

```
ExceptionDescription:Configurationerror.Class[oracle.jdbc.driver.OracleDriver
] not found.
```

The Oracle JDBC.jar file cannot be found.

#### Solution

Make sure the file is in the appropriate directory on the Ambari server and re-run `ambari-server setup`. Review the load database procedure appropriate for your database type in Using Non-Default Databases - Ambari.

### *Problem: Ambari Server Fails to Start: No Connection*

Check `/var/log/ambari-server/ambari-server.log` for the following error:

```
The Network Adapter could not establish the connection Error Code: 17002
```

Ambari Server cannot connect to the database.

#### Solution

Confirm that the database host is reachable from the Ambari Server and is correctly configured by reading `/etc/ambari-server/conf/ambari.properties`.

```
server.jdbc.url=jdbc:oracle:thin:@oracle.database.hostname:1521/ambaridb
server.jdbc.rca.url=jdbc:oracle:thin:@oracle.database.hostname:1521/ambari
```

### *Problem: Ambari Server Fails to Start: Bad Username*

Check `/var/log/ambari-server/ambari-server.log` for the following error:

```
Internal Exception: java.sql.SQLException:ORA-01017: invalid
username/password; logon denied
```

You are using an invalid username/password.

#### Solution

Confirm the user account is set up in the database and has the correct privileges. See Step 3 above.

### *Problem: Ambari Server Fails to Start: No Schema*

Check `/var/log/ambari-server/ambari-server.log` for the following error:

```
Internal Exception: java.sql.SQLSyntaxErrorException: ORA-00942: table or
view does not exist
```

The schema has not been loaded.

Solution

Confirm you have loaded the database schema. Review the load database schema procedure appropriate for your database type in Using Non-Default Databases - Ambari.

## Using Non-Default Databases - Hive

The following sections describe how to use Hive with an existing database, other than the MySQL database instance that Ambari installs by default.

- Using Hive with Oracle
- Using Hive with MySQL
- Using Hive with PostgreSQL
- Troubleshooting Non-Default Databases with Hive

### Using Hive with Oracle

To set up Oracle for use with Hive:

1. On the Ambari Server host, stage the appropriate JDBC driver file for later deployment.
    1. Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.
    2. Select `Oracle Database 11g Release 2 - ojdbc6.jar` and download the file.
    3. Make sure the .jar file has the appropriate permissions - 644.
    4. Execute the following command, adding the path to the downloaded .jar file:

    ```
    ambari-server setup --jdbc-db=oracle --jdbc-
    driver=/path/to/downloaded/ojdbc6.jar
    ```
2. Create a user for Hive and grant it permissions.
    - Using the Oracle database admin utility:

    ```
    # sqlplus sys/root as sysdba
    CREATE USER <HIVEUSER> IDENTIFIED BY <HIVEPASSWORD>;
    GRANT SELECT_CATALOG_ROLE TO <HIVEUSER>;
    GRANT CONNECT, RESOURCE TO <HIVEUSER>;
    QUIT;
    ```
    - Where `<HIVEUSER>` is the Hive user name and `<HIVEPASSWORD>` is the Hive user password.
3. Load the Hive database schema.
    - For a **HDP 2.2 Stack**

⚠ Ambari sets up the Hive Metastore database schema automatically.
You do not need to pre-load the Hive Metastore database schema into your Oracle database for a HDP 2.2 Stack.

- For a **HDP 2.1 Stack**

  You must pre-load the Hive database schema into your Oracle database using the schema script, as follows:
  sqlplus <HIVEUSER>/<HIVEPASSWORD> < hive-schema-0.13.0.oracle.sql

  Find the `hive-schema-0.13.0.oracle.sql` file in the `/var/lib/ambari-server/resources/stacks/HDP/2.1/services/HIVE/etc/` directory of the Ambari Server host after you have installed Ambari Server.
- For a **HDP 2.0 Stack**

  You must pre-load the Hive database schema into your Oracle database using the schema script, as follows:
  sqlplus <HIVEUSER>/<HIVEPASSWORD> < hive-schema-0.12.0.oracle.sql

  Find the `hive-schema-0.12.0.oracle.sql` file in the `/var/lib/ambari-server/resources/stacks/HDP/2.0.6/services/HIVE/etc/` directory of the Ambari Server host after you have installed Ambari Server.
- For a **HDP 1.3 Stack**

  You must pre-load the Hive database schema into your Oracle database using the schema script, as follows:
  sqlplus <HIVEUSER>/<HIVEPASSWORD> < hive-schema-0.10.0.oracle.sql

  Find the `hive-schema-0.10.0.oracle.sql` file in the `/var/lib/ambari-server/resources/stacks/HDP/1.3.2/services/HIVE/etc/` directory of the Ambari Server host after you have installed Ambari Server.

## Using Hive with MySQL

To set up MySQL for use with Hive:

1. On the Ambari Server host, stage the appropriate MySQL connector for later deployment.
   1. Install the connector.

      **RHEL/CentOS/Oracle Linux**
      ```
      yum install mysql-connector-java*
      ```

      **SLES**
      ```
      zypper install mysql-connector-java*
      ```

**UBUNTU**

```
apt-get install mysql-connector-java*
```

2 Confirm that `mysql-connector-java.jar` is in the Java share directory.

```
ls /usr/share/java/mysql-connector-java.jar
```

3 Make sure the .jar file has the appropriate permissions - 644.

4 Execute the following command:

```
ambari-server setup --jdbc-db=mysql --jdbc-
driver=/usr/share/java/mysql-connector-java.jar
```

2 Create a user for Hive and grant it permissions.

- Using the MySQL database admin utility:

```
# mysql -u root -p
CREATE USER '<HIVEUSER>'@'localhost' IDENTIFIED BY
'<HIVEPASSWORD>';
GRANT ALL PRIVILEGES ON *.* TO '<HIVEUSER>'@'localhost';
CREATE USER '<HIVEUSER>'@'%' IDENTIFIED BY '<HIVEPASSWORD>';
GRANT ALL PRIVILEGES ON *.* TO '<HIVEUSER>'@'%';
CREATE USER '<HIVEUSER>'@'<HIVEMETASTOREFQDN>'IDENTIFIED BY
'<HIVEPASSWORD>';
GRANT ALL PRIVILEGES ON *.* TO
'<HIVEUSER>'@'<HIVEMETASTOREFQDN>';
FLUSH PRIVILEGES;
```

- Where `<HIVEUSER>` is the Hive user name, `<HIVEPASSWORD>` is the Hive user password and `<HIVEMETASTOREFQDN>` is the Fully Qualified Domain Name of the Hive Metastore host.

3 Create the Hive database.

The Hive database must be created before loading the Hive database schema.

```
# mysql -u root -p CREATE DATABASE <HIVEDATABASE>
```

Where `<HIVEDATABASE>` is the Hive database name.

4 Load the Hive database schema.

- For a **HDP 2.2 Stack:**

Ambari sets up the Hive Metastore database schema automatically.
You do not need to pre-load the Hive Metastore database schema into your MySQL database for a HDP 2.2 Stack.

- For a **HDP 2.1 Stack:**

You must pre-load the Hive database schema into your MySQL database using the schema script, as follows.

```
mysql -u root -p <HIVEDATABASE> hive-schema-0.13.0.mysql.sql
```

Find the `hive-schema-0.13.0.mysql.sql` file in the `/var/lib/ambari-server/resources/stacks/HDP/2.1/services/HIVE/etc/` directory of the Ambari Server host after you have installed Ambari Server.

## Using Hive with PostgreSQL

To set up PostgreSQL for use with Hive:

1. On the Ambari Server host, stage the appropriate PostgreSQL connector for later deployment.
    1. Install the connector.

        **RHEL/CentOS/Oracle Linux**
        ```
        yum install postgresql-jdbc*
        ```

        **SLES**
        ```
         zypper install -y postgresql-jdbc
        ```
    2. Copy the connector.jar file to the Java share directory.
        ```
        cp /usr/share/pgsql/postgresql-*.jdbc3.jar
        /usr/share/java/postgresql-jdbc.jar
        ```
    3. Confirm that .jar is in the Java share directory.
        ```
        ls /usr/share/java/postgresql-jdbc.jar
        ```
    4. Change the access mode of the.jar file to 644.

        chmod 644 /usr/share/java/postgresql-jdbc.jar
    5. Execute the following command:
        ```
        ambari-server setup --jdbc-db=postgres --jdbc-
        driver=/usr/share/java/postgresql-connector-java.jar
        ```
2. Create a user for Hive and grant it permissions.
    - Using the PostgreSQL database admin utility:
        ```
        echo "CREATE DATABASE <HIVEDATABASE>;" | psql -U postgres
        echo "CREATE USER <HIVEUSER> WITH PASSWORD '<HIVEPASSWORD>';" |
        psql -U postgres
        echo "GRANT ALL PRIVILEGES ON DATABASE <HIVEDATABASE> TO
        <HIVEUSER>;" | psql -U postgres
        ```
    - Where `<HIVEUSER>` is the Hive user name, `<HIVEPASSWORD>` is the Hive user password and `<HIVEDATABASE>` is the Hive database name.

3 Load the Hive database schema.

- For a **HDP 2.2 Stack:**

  Ambari sets up the Hive Metastore database schema automatically.
  You do not need to pre-load the Hive Metastore database schema into your PostgreSQL database for a HDP 2.2 Stack.

- For a **HDP 2.1 Stack:**

  You must pre-load the Hive database schema into your PostgreSQL database using the schema script, as follows:

  ```
  # psql -U <HIVEUSER> -d <HIVEDATABASE>
  \connect <HIVEDATABASE>;
  \i hive-schema-0.13.0.postgres.sql;
  ```

  Find the `hive-schema-0.13.0.postgres.sql` file in the `/var/lib/ambari-server/resources/stacks/HDP/2.1/services/HIVE/etc/` directory of the Ambari Server host after you have installed Ambari Server.

- For a **HDP 2.0 Stack:**

  You must pre-load the Hive database schema into your PostgreSQL database using the schema script, as follows:

  ```
  # sudo -u postgres psql
  \connect <HIVEDATABASE>;
  \i hive-schema-0.12.0.postgres.sql;
  ```

  Find the `hive-schema-0.12.0.postgres.sql` file in the `/var/lib/ambari-server/resources/stacks/HDP/2.0.6/services/HIVE/etc/` directory of the Ambari Server host after you have installed Ambari Server.

- For a **HDP 1.3 Stack:**

  You must pre-load the Hive database schema into your PostgreSQL database using the schema script, as follows:

  ```
  # sudo -u postgres psql
  \connect <HIVEDATABASE>;
  \i hive-schema-0.10.0.postgres.sql;
  ```

  Find the `hive-schema-0.10.0.postgres.sql` file in the `/var/lib/ambari-server/resources/stacks/HDP/1.3.2/services/HIVE/etc/` directory of the Ambari Server host after you have installed Ambari Server.

## Troubleshooting Hive

Use these entries to help you troubleshoot any issues you might have installing Hive with non-default databases.

### *Problem: Hive Metastore Install Fails Using Oracle*

Check the install log:

```
cp /usr/share/java/${jdbc_jar_name} ${target}] has failures: true
```

The Oracle JDBC.jar file cannot be found.

#### Solution

Make sure the file is in the appropriate directory on the Hive Metastore server and click **Retry**.

### *Problem: Install Warning when "Hive Check Execute" Fails Using Oracle*

Check the install log:

```
java.sql.SQLSyntaxErrorException: ORA-01754:
a table may contain only one column of type LONG
```

The Hive Metastore schema was not properly loaded into the database.

#### Solution

Ignore the warning, and complete the install. Check your database to confirm the Hive Metastore schema is loaded. In the Ambari Web GUI, browse to **Services** > **Hive.** Choose **Service Actions > Service Check** to check that the schema is correctly in place.

### *Problem: Hive Check Execute may fail after completing an Ambari upgrade to version 1.4.2*

For secure and non-secure clusters, with Hive security authorization enabled, the Hive service check may fail. Hive security authorization may not be configured properly.

#### Solution

Two workarounds are possible. Using Ambari Web, in **HiveConfigsAdvanced**:

- Disable `hive.security.authorization`, by setting the `hive.security.authorization.enabled` value to false.

***or***

- Properly configure Hive security authorization. For example, set the following properties:

  For more information about configuring Hive security, see Metastore Server Security in Hive Authorization and the HCatalog document Storage Based Authorization.

*Table 9. Hive Security Authorization Settings*

| Property | Value |
|---|---|
| hive.security.authorization.manager | org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationProvider |
| hive.security.metastore.authorization.manager | org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationProvider |
| hive.security.authenticator.manager | org.apache.hadoop.hive.ql.security.ProxyUserAuthenticator |

Metastore Server Security
Hive Authorization
Storage Based Authorization

## Using Non-Default Databases - Oozie

The following sections describe how to use Oozie with an existing database, other than the Derby database instance that Ambari installs by default.

- Using Oozie with Oracle
- Using Oozie with MySQL
- Using Oozie with PostgreSQL
- Troubleshooting Non-Default Databases with Oozie

### Using Oozie with Oracle

To set up Oracle for use with Oozie:

1. On the Ambari Server host, stage the appropriate JDBC driver file for later deployment.
    1. Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.
    2. Select Oracle Database 11g Release 2 - ojdbc6.jar.
    3. Make sure the .jar file has the appropriate permissions - 644.
    4. Execute the following command, adding the path to the downloaded.jar file:

    ```
    ambari-server setup --jdbc-db=oracle --jdbc-
    driver=/path/to/downloaded/ojdbc6.jar
    ```
2. Create a user for Oozie and grant it permissions.

    Using the Oracle database admin utility, run the following commands:

    ```
    # sqlplus sys/root as sysdba
    CREATE USER <OOZIEUSER> IDENTIFIED BY <OOZIEPASSWORD>;
    GRANT ALL PRIVILEGES TO <OOZIEUSER>;
    QUIT;
    ```

    Where `<OOZIEUSER>` is the Oozie user name and `<OOZIEPASSWORD>` is the Oozie user password.

### Using Oozie with MySQL

To set up MySQL for use with Oozie:

1. On the Ambari Server host, stage the appropriate MySQL connector for later deployment.
    1. Install the connector.

        **RHEL/CentOS/Oracle Linux**
        ```
        yum install mysql-connector-java*
        ```

**SLES**
```
zypper install mysql-connector-java*
```

**UBUNTU**
```
apt-get install mysql-connector-java*
```

2. Confirm that `mysql-connector-java.jar` is in the Java share directory.

```
ls /usr/share/java/mysql-connector-java.jar
```

3. Make sure the .jar file has the appropriate permissions - 644.
4. Execute the following command:

```
ambari-server setup --jdbc-db=mysql --jdbc-
driver=/usr/share/java/mysql-connector-java.jar
```

2. Create a user for Oozie and grant it permissions.
   - Using the MySQL database admin utility:

   ```
   # mysql -u root -p
   CREATE USER '<OOZIEUSER>'@'%' IDENTIFIED BY '<OOZIEPASSWORD>';
   GRANT ALL PRIVILEGES ON *.* TO '<OOZIEUSER>'@'%';
   FLUSH PRIVILEGES;
   ```

   - Where `<OOZIEUSER>` is the Oozie user name and `<OOZIEPASSWORD>` is the Oozie user password.
3. Create the Oozie database.
   - The Oozie database must be created prior.

   ```
   # mysql -u root -p
   CREATE DATABASE <OOZIEDATABASE>
   ```

   - Where `<OOZIEDATABASE>` is the Oozie database name.

## Using Oozie with PostgreSQL

To set up PostgreSQL for use with Oozie:

1. On the Ambari Server host, stage the appropriate PostgreSQL connector for later deployment.
   1. Install the connector.

**RHEL/CentOS/Oracle Linux**
```
yum install postgresql-jdbc
```

**SLES**
```
zypper install -y postgresql-jdbc
```

**UBUNTU**

```
apt-get install -y postgresql-jdbc
```

2 Copy the connector.jar file to the Java share directory.

```
cp /usr/share/pgsql/postgresql-*.jdbc3.jar
/usr/share/java/postgresql-jdbc.jar
```

3 Confirm that .jar is in the Java share directory.

```
ls /usr/share/java/postgresql-jdbc.jar
```

4 Change the access mode of the .jar file to 644.

```
chmod 644 /usr/share/java/postgresql-jdbc.jar
```

5 Execute the following command:

```
ambari-server setup --jdbc-db=postgres --jdbc-
driver=/usr/share/java/postgresql-connector-java.jar
```

2 Create a user for Oozie and grant it permissions.

- Using the PostgreSQL database admin utility:

```
echo "CREATE DATABASE <OOZIEDATABASE>;" | psql -U postgres
echo "CREATE USER <OOZIEUSER> WITH PASSWORD '<OOZIEPASSWORD>';" |
psql -U postgres
echo "GRANT ALL PRIVILEGES ON DATABASE <OOZIEDATABASE> TO
<OOZIEUSER>;" | psql -U postgres
```

- Where `<OOZIEUSER>` is the Oozie user name, `<OOZIEPASSWORD>` is the Oozie user password and `<OOZIEDATABASE>` is the Oozieozie database name.

## Troubleshooting Oozie

Use these entries to help you troubleshoot any issues you might have installing Oozie with non-default databases.

### *Problem: Oozie Server Install Fails Using MySQL*

Check the install log:

```
cp /usr/share/java/mysql-connector-java.jar
usr/lib/oozie/libext/mysql-connector-java.jar
has failures: true
```

The MySQL JDBC.jar file cannot be found.

#### Solution

Make sure the file is in the appropriate directory on the Oozie server and click **Retry.**

### *Problem: Oozie Server Install Fails Using Oracle or MySQL*

Check the install log:

```
Exec[exec cd /var/tmp/oozie &&
/usr/lib/oozie/bin/ooziedb.sh create -sqlfile oozie.sql -run ]
has failures: true
```

Oozie was unable to connect to the database or was unable to successfully setup the schema for Oozie.

#### Solution

Check the database connection settings provided during the **Customize Services** step in the install wizard by browsing back to **Customize Services > Oozie.** After confirming and adjusting your database settings, proceed forward with the install wizard.

If the Install Oozie Server wizard continues to fail, get more information by connecting directly to the Oozie server and executing the following command as `<OOZIEUSER>`:

```
su oozie /usr/lib/oozie/bin/ooziedb.sh create -sqlfile oozie.sql -run
```

# Setting up an Internet Proxy Server for Ambari

If you plan to use the public repositories for installing the Stack, Ambari Server must have Internet access to confirm access to the repositories and validate the repositories. If your machine requires use of a proxy server for Internet access, you must configure Ambari Server to use the proxy server.

How To Set Up an Internet Proxy Server for Ambari

## How To Set Up an Internet Proxy Server for Ambari

1. On the Ambari Server host, add proxy settings to the following script: `/var/lib/ambari-server/ambari-env.sh.`

   ```
   -Dhttp.proxyHost=<yourProxyHost> -Dhttp.proxyPort=<yourProxyPort>
   ```

2. Optionally, to prevent some host names from accessing the proxy server, define the list of excluded hosts, as follows:

   ```
   -Dhttp.nonProxyHosts=<pipe|separated|list|of|hosts>
   ```

3. If your proxy server requires authentication, add the user name and password, as follows:

   ```
   -Dhttp.proxyUser=<username> -Dhttp.proxyPassword=<password>
   ```

4. Restart the Ambari Server to pick up this change.

If you plan to use local repositories, see Optional: Configure Local Repositories. Configuring Ambari to use a proxy server and have Internet access is not required. The Ambari Server must have access to your local repositories.

# Configuring Network Port Numbers

This chapter lists port number assignments required to maintain communication between Ambari Server, Ambari Agents, Ambari Web UI, Ganglia, and Nagios components.

- Default Network Port Numbers - Ambari
- Configuring Ganglia Ports
- Configuring Nagios Ports
- Optional: Changing the Default Ambari Server Port

For more information about configuring port numbers for Stack components, see Configuring Ports in the HDP Stack documentation.

## Default Network Port Numbers - Ambari

The following table lists the default ports used by Ambari Server and Ambari Agent services.

| Service | Servers | Default Ports Used | Protocol | Description | Need End User Access? | Configuration Parameters |
|---|---|---|---|---|---|---|
| Ambari Server | Ambari Server host | 8080 See Optional: Change the Ambari Server Port for instructions on changing the default port. | http See Optional: Set Up HTTPS for Ambari Server for instructions on enabling HTTPS. | Interface to Ambari Web and Ambari REST API | No | |
| Ambari Server | Ambari Server host | 8440 | https | Handshake Port for Ambari Agents to Ambari Server | No | |
| Ambari Server | Ambari Server host | 8441 | https | Registration and Heartbeat Port for Ambari Agents to Ambari Server | No | |

| | | | | | | |
|---|---|---|---|---|---|---|
| Ambari Agent | All hosts running Ambari Agents | 8670 You can change the Ambari Agent ping port in the Ambari Agent configuration. If you change the port, you must restart Nagios after making the change. | tcp | Ping port used for Nagios Server to check the health of the Ambari Agent | No | |

## Ganglia Ports

The following table lists the default ports used by the various Ganglia services.

| Service | Servers | Default Ports Used | Protocol | Description | Need End User Access? | Configuration Parameters |
|---|---|---|---|---|---|---|
| Ganglia Server | Ganglia server host | 8660/61/62/63 | | For metric (gmond) collectors | No | |
| Ganglia Monitor | All Slave Node hosts | 8660 | | For monitoring (gmond) agents | No | |
| Ganglia Server | Ganglia server host | 8651 | | For ganglia gmetad | | |
| Ganglia Web | Ganglia server host | | httpSee Optional: Set Up HTTPS for Ganglia for instructions on enabling HTTPS. | | | |

## Nagios Ports

The following table lists the default port used by the Nagios server.

| Service | Servers | Default Ports Used | Protocol | Description | Need End User Access? | Configuration Parameters |
|---|---|---|---|---|---|---|
| Nagios Server | Nagios server host | 80 | httpSee Optional: Set Up HTTPS for Nagios for instructions on enabling HTTPS. | Nagios Web UI | No | |

## Optional: Changing the Default Ambari Server Port

By default, Ambari Server uses port 8080 to access the Ambari Web UI and the REST API. To change the port number, you must edit the Ambari properties file.

Ambari Server should not be running when you change port numbers. Edit `ambari.properties` before you start Ambari Server the first time or stop Ambari Server before editing properties.

1. On the Ambari Server host, open `/etc/ambari-server/conf/ambari.properties` with a text editor.
2. Add the client API port property and set it to your desired port value:

   ```
   client.api.port=<port_number>
   ```
3. Start or re-start the Ambari Server. Ambari Server now accesses Ambari Web via the newly configured port:

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

# Changing the JDK Version on an Existing Cluster

During your initial Ambari Server Setup, you selected the JDK to use or provided a path to a custom JDK already installed on your hosts. After setting up your cluster, you may change the JDK version using the following procedure.

How to change the JDK Version for an Existing Cluster

## How to change the JDK Version for an Existing Cluster

1. Re-run Ambari Server Setup.

   ```
   ambari-server setup
   ```

2. At the prompt to change the JDK, Enter **y**.

   ```
   Do you want to change Oracle JDK [y/n] (n)? y
   ```

3. At the prompt to choose a JDK, Enter `1` to change the JDK to v1.7.

   ```
   [1] - Oracle JDK 1.7
   [2] - Oracle JDK 1.6
   [3] - Custom JDK Enter choice: 3
   ```

   If you choose Oracle JDK 1.7 or Oracle JDK 1.6, the JDK you choose downloads and installs automatically.

4. If you choose **Custom JDK**, verify or add the custom JDK path on all hosts in the cluster.
5. After setup completes, you must restart each component for the new JDK to be used by the Hadoop services.

   Using the Ambari Web UI, do the following tasks:

   - Restart each component
   - Restart each host
   - Restart all services

For more information about managing services in your cluster, see Monitoring and Managing Services.

# Configuring NameNode High Availability

Ambari sets up active and standby NameNode hosts on a new cluster, by default. Configuring NameNode High Availability (HA) sets the standby NameNode to handle the active NameNode workload in the event that the active NameNode fails.

NameNode HA is available with HDP Stack v2.0 or later.

Following topics describe:

- How to Set Up NameNode HA
- How to Roll Back NameNode HA

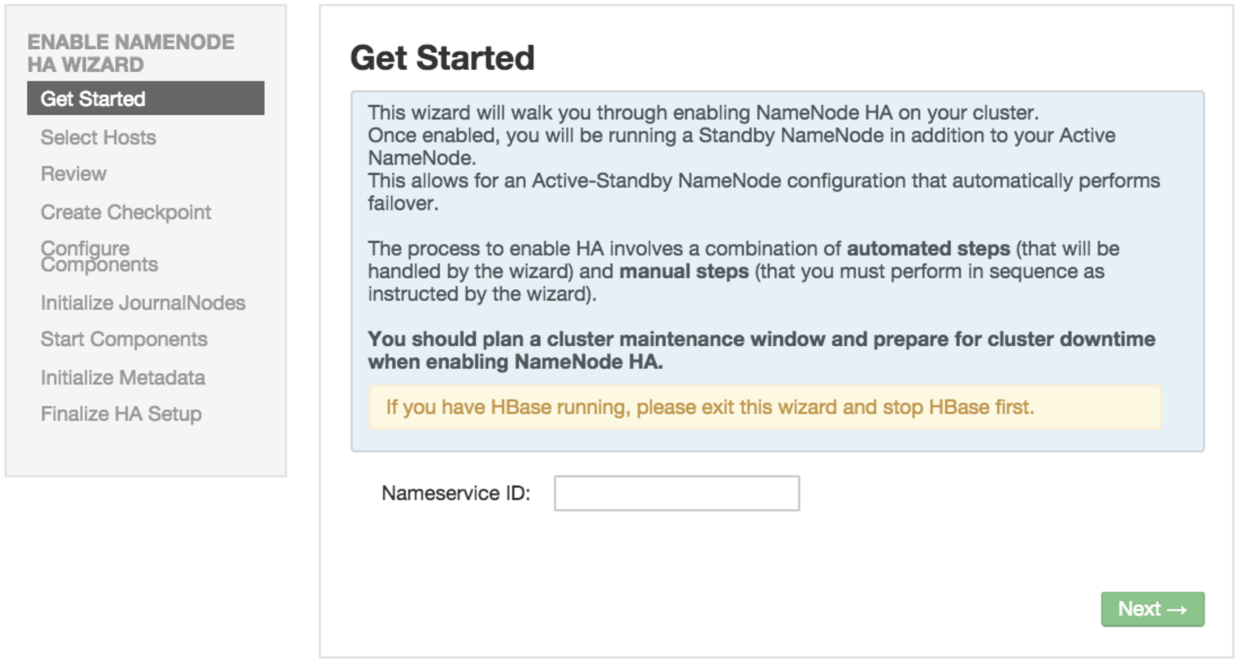
## How To Set Up NameNode High Availability

1 Check to make sure you have at least three hosts in your cluster and are running at least three ZooKeeper servers.

2 In Ambari Web, select **Services > HDFS > Summary. Select Service Actions and choose Enable NameNode HA.**

3 The Enable HA Wizard launches. This wizard describes the set of automated and manual steps you must take to set up NameNode high availability.

4 **Get Started** : This step gives you an overview of the process and allows you to select a Nameservice ID. You use this Nameservice ID instead of the NameNode FQDN once HA has been set up. Click **Next** to proceed.



Get Started

This wizard will walk you through enabling NameNode HA on your cluster.
Once enabled, you will be running a Standby NameNode in addition to your Active NameNode.
This allows for an Active-Standby NameNode configuration that automatically performs failover.

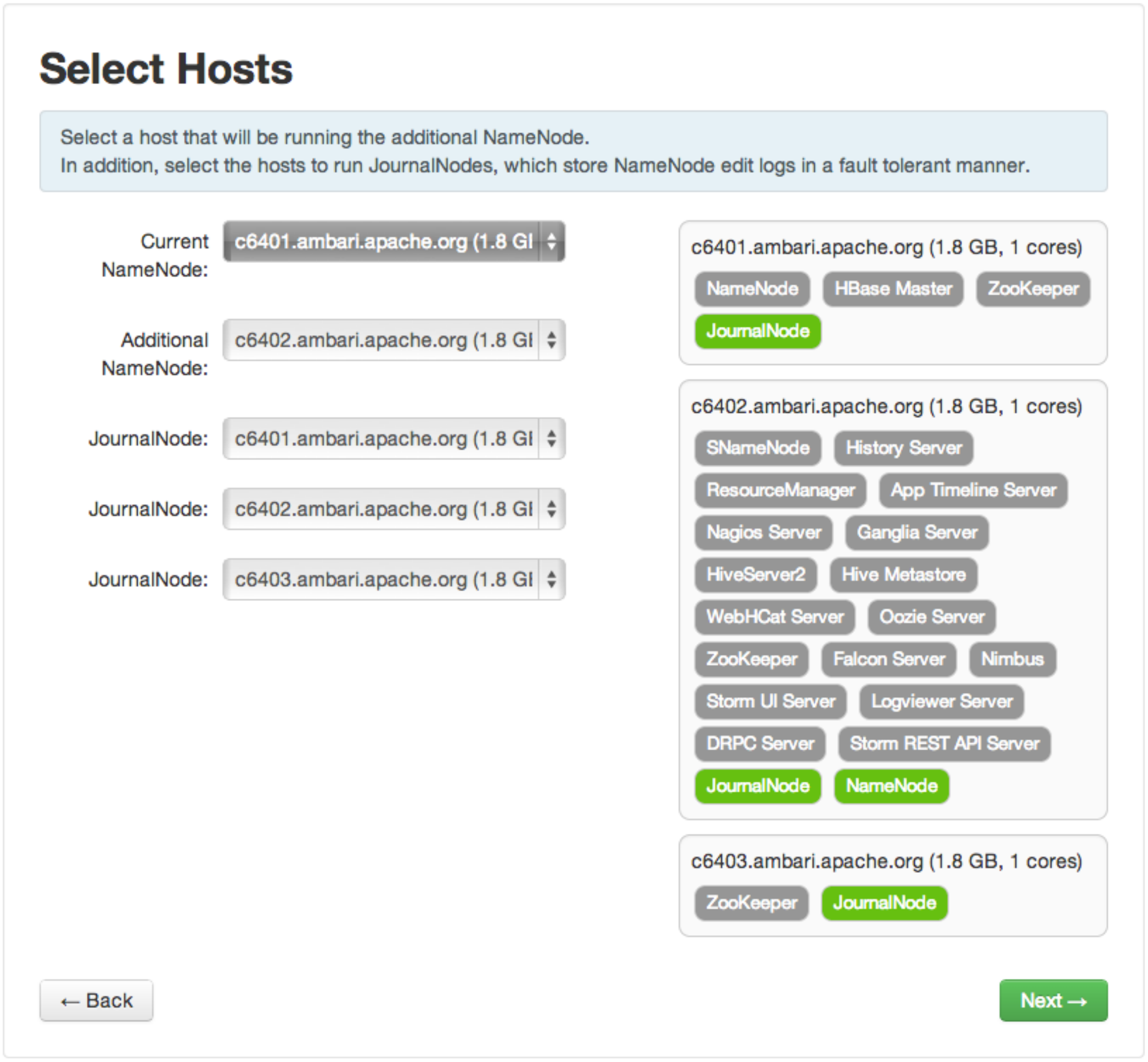
The process to enable HA involves a combination of **automated steps** (that will be handled by the wizard) and **manual steps** (that you must perform in sequence as instructed by the wizard).

**You should plan a cluster maintenance window and prepare for cluster downtime when enabling NameNode HA.**

If you have HBase running, please exit this wizard and stop HBase first.

Nameservice ID:

5 **Select Hosts** : Select a host for the additional NameNode and the JournalNodes. The wizard suggest options that you can adjust using the drop-down lists. Click **Next** to proceed.



**Select Hosts**

Select a host that will be running the additional NameNode.
In addition, select the hosts to run JournalNodes, which store NameNode edit logs in a fault tolerant manner.

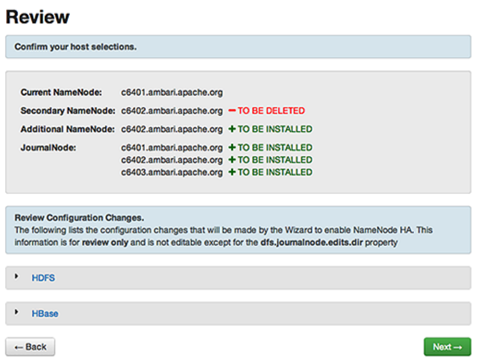
| | |
|---|---|
| Current NameNode: | c6401.ambari.apache.org (1.8 GI |
| Additional NameNode: | c6402.ambari.apache.org (1.8 GI |
| JournalNode: | c6401.ambari.apache.org (1.8 GI |
| JournalNode: | c6402.ambari.apache.org (1.8 GI |
| JournalNode: | c6403.ambari.apache.org (1.8 GI |

c6401.ambari.apache.org (1.8 GB, 1 cores)
NameNode, HBase Master, ZooKeeper, JournalNode

c6402.ambari.apache.org (1.8 GB, 1 cores)
SNameNode, History Server, ResourceManager, App Timeline Server, Nagios Server, Ganglia Server, HiveServer2, Hive Metastore, WebHCat Server, Oozie Server, ZooKeeper, Falcon Server, Nimbus, Storm UI Server, Logviewer Server, DRPC Server, Storm REST API Server, JournalNode, NameNode

c6403.ambari.apache.org (1.8 GB, 1 cores)
ZooKeeper, JournalNode

6 **Review** : Confirm your host selections and click **Next**.



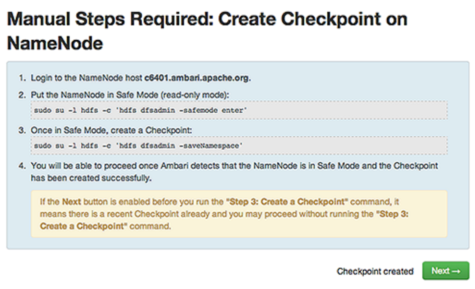
**Review**

Confirm your host selections.

| | | |
|---|---|---|
| Current NameNode: | c6401.ambari.apache.org | |
| Secondary NameNode: | c6402.ambari.apache.org | − TO BE DELETED |
| Additional NameNode: | c6402.ambari.apache.org | + TO BE INSTALLED |
| JournalNode: | c6401.ambari.apache.org | + TO BE INSTALLED |
| | c6402.ambari.apache.org | + TO BE INSTALLED |
| | c6403.ambari.apache.org | + TO BE INSTALLED |

**Review Configuration Changes.**
The following lists the configuration changes that will be made by the Wizard to enable NameNode HA. This information is for **review only** and is not editable except for the **dfs.journalnode.edits.dir** property

7 **Create Checkpoints** : Follow the instructions in the step. You need to log in to your **current** NameNode host to run the commands to put your NameNode into safe mode and create a checkpoint. When Ambari detects success, the message on the bottom of the window changes. Click **Next.**



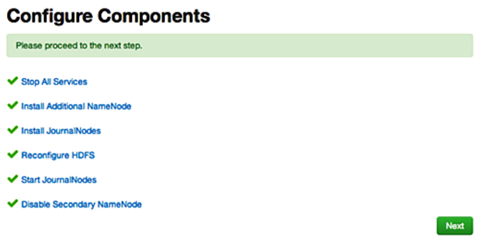
**Manual Steps Required: Create Checkpoint on NameNode**

1. Login to the NameNode host **c6401.ambari.apache.org**.
2. Put the NameNode in Safe Mode (read-only mode):
```
sudo su -l hdfs -c 'hdfs dfsadmin -safemode enter'
```
3. Once in Safe Mode, create a Checkpoint:
```
sudo su -l hdfs -c 'hdfs dfsadmin -saveNamespace'
```
4. You will be able to proceed once Ambari detects that the NameNode is in Safe Mode and the Checkpoint has been created successfully.

If the **Next** button is enabled before you run the **"Step 3: Create a Checkpoint"** command, it means there is a recent Checkpoint already and you may proceed without running the **"Step 3: Create a Checkpoint"** command.

Checkpoint created

8 **Configure Components** : The wizard configures your components, displaying progress bars to let you track the steps. Click **Next** to continue.



**Configure Components**

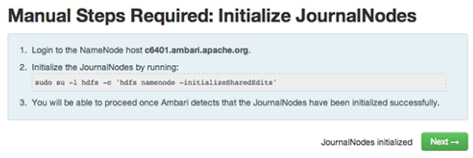
Please proceed to the next step.

✓ Stop All Services
✓ Install Additional NameNode
✓ Install JournalNodes
✓ Reconfigure HDFS
✓ Start JournalNodes
✓ Disable Secondary NameNode

9 **Initialize JournalNodes** : Follow the instructions in the step. You need to login to your **current** NameNode host to run the command to initialize the JournalNodes. When Ambari detects success, the message on the bottom of the window changes. Click **Next.**

***After upgrading to Ambari 1.6.1, or using a Blueprint to install your cluster,*** initializing JournalNodes may fail. For information about how to work around this issue, see the the following topic in the Ambari Troubleshooting Guide:

Enabling NameNode HA wizard fails at the Initialize JournalNode step

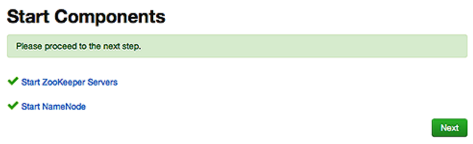


**Manual Steps Required: Initialize JournalNodes**

1. Login to the NameNode host **c6401.ambari.apache.org**.
2. Initialize the JournalNodes by running:
```
sudo su -l hdfs -c 'hdfs namenode -initializeSharedEdits'
```
3. You will be able to proceed once Ambari detects that the JournalNodes have been initialized successfully.

JournalNodes initialized

10 **Start Components** : The wizard starts the ZooKeeper servers and the NameNode, displaying progress bars to let you track the steps. Click **Next** to continue.

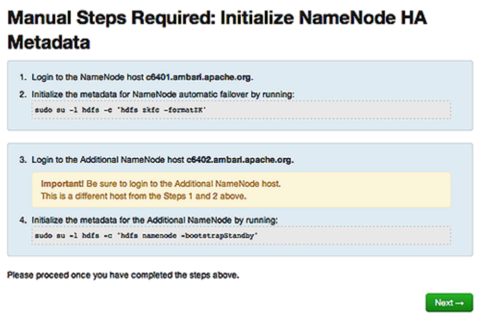


**Start Components**

Please proceed to the next step.

✓ Start ZooKeeper Servers
✓ Start NameNode

11 **Initialize Metadata** : Follow the instructions in the step. For this step you must log in to both the **current** NameNode and the **additional** NameNode. Make sure you are logged in to the correct host for each command. Click **Next** when you have completed the two commands. A **Confirmation** pop-up window displays, reminding you to do both steps. Click **OK** to confirm.



Manual Steps Required: Initialize NameNode HA Metadata

1. Login to the NameNode host **c6401.ambari.apache.org.**
2. Initialize the metadata for NameNode automatic failover by running:

```
sudo su -l hdfs -c 'hdfs zkfc -formatZK'
```

3. Login to the Additional NameNode host **c6402.ambari.apache.org.**

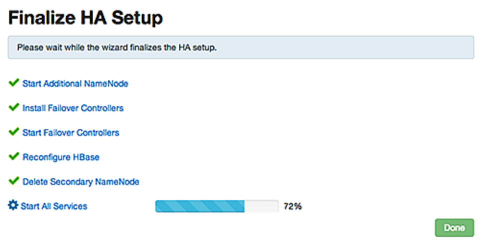
**Important!** Be sure to login to the Additional NameNode host.
This is a different host from the Steps 1 and 2 above.

4. Initialize the metadata for the Additional NameNode by running:

```
sudo su -l hdfs -c 'hdfs namenode -bootstrapStandby'
```

Please proceed once you have completed the steps above.

12 **Finalize HA Setup** : The wizard the setup, displaying progress bars to let you track the steps. Click **Done** to finish the wizard. After the Ambari Web GUI reloads, you may see some alert notifications. Wait a few minutes until the services come back up. If necessary, restart any components using Ambari Web.



Finalize HA Setup

Please wait while the wizard finalizes the HA setup.

✓ Start Additional NameNode
✓ Install Failover Controllers
✓ Start Failover Controllers
✓ Reconfigure HBase
✓ Delete Secondary NameNode
Start All Services 72%

Choose Services, then start Nagios, after completing all steps in the HA wizard.

13 If you are using Hive, you must manually change the Hive Metastore FS root to point to the Nameservice URI instead of the NameNode URI. You created the Nameservice ID in the Get Started step.

1 Check the current FS root. On the Hive host:

```
hive --config /etc/hive/conf.server --service metatool -
listFSRoot
```

The output looks similar to the following:

```
Listing FS Roots...
hdfs://<namenode-host>/apps/hive/warehouse
```

2 Use this command to change the FS root:

```
$ hive --config /etc/hive/conf.server --service metatool  -
updateLocation <new-location><old-location>
```

For example, where the Nameservice ID is mycluster:

```
 $ hive --config /etc/hive/conf.server --service metatool -
updateLocation hdfs://mycluster/apps/hive/warehouse
hdfs://c6401.ambari.apache.org/apps/hive/warehouse
```

The output looks similar to the following:

```
Successfully updated the following locations...
Updated X records in SDS table
```

14 If you are using Oozie, you must use the Nameservice ID instead of the NameNode URI in your workflow files. For example, where the Nameservice ID is `mycluster`:

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
 <start to="mr-node"/>
 <action name="mr-node">
   <map-reduce>
     <job-tracker><jobTracker></job-tracker>
     <name-node>hdfs://mycluster</name-node>
```

15 If you are using Hue, to enable NameNode HighAvailability, you must use httpfs instead of webhdfs to communicate with name nodes inside the cluster. After successfully setting up NameNode High Availability:

1 Install an httpfs server on any node in the cluster:

```
yum install hadoop-httpfs
```

2 Ensure that Hue hosts and groups use the httpfs server.

For example, on the httpfs server host, add to httpfs-site.xml the following lines:

```
<property>
<name>httpfs.proxyuser.hue.hosts</name> <value>*</value>
</property>
<property> <name>httpfs.proxyuser.hue.groups</name>
<value>*</value>
</property>
```

3 Ensure that groups and hosts in the cluster use the httpfs server. For example, use **Services > HDFS > Configs** to add the following properties and values to `core-site.xml`.

| Property | Value |
|---|---|
| hadoop.proxyuser.httpfs.groups | * |
| hadoop.proxyuser.httpfs.hosts | * |

4 Using Ambari, in **Services >HDFS** restart the HDFS service in your cluster.

5 On the Hue host, configure Hue to use the httpfs server by editing hue.ini to include the following line:

`webhdfs_url=http://<fqdn.of.httpfs.server>:14000/webhdfs/v1/`

6 Restart the Hue service.

16 Adjust the ZooKeeper Failover Controller retries setting for your environment.

- Browse to **Services > HDFS > Configs >** `core-site`.
- Set `ha.failover-controller.active-standby-elector.zk.op.retries=120`

## How to Roll Back NameNode HA

To roll back NameNode HA to the previous non-HA state use the following step-by-step manual process, depending on your installation.

1. Stop HBase
2. Checkpoint the ActiveNameNode
3. Stop All Services
4. Prepare the AmbariHost for Rollback
5. Restore the HBase Configuration
6. Delete ZooKeeper Failover Controllers
7. Modify HDFS Configurations
8. Recreate thestandby NameNode
9. Re-enable the standbyNameNode
10. Delete AllJournalNodes
11. Delete theAdditional NameNode
12. Verify the HDFS Components
13. Start HDFS

### Stop HBase

1. From Ambari Web, go to the Services view and select HBase.
2. Choose **Service Actions > Stop.**
3. Wait until HBase has stopped completely before continuing.

### Checkpoint the Active NameNode

If HDFS has been in use **after** you enabled NameNode HA, but you wish to revert back to a non-HA state, you must checkpoint the HDFS state before proceeding with the rollback.

If the **Enable NameNode HA** wizard failed and you need to revert back, you can skip this step and move on to Stop All Services.

- If Kerberos security has **not** been enabled on the cluster:

  On the Active NameNode host, execute the following commands to save the namespace. You must be the HDFS service user to do this.

  ```
  sudo su -l <HDFS_USER> -c 'hdfs dfsadmin -safemode enter'
  sudo su -l <HDFS_USER> -c 'hdfs dfsadmin -saveNamespace'
  ```

- If Kerberos security **has** been enabled on the cluster:

  ```
  sudo su -l <HDFS_USER> -c 'kinit -kt
  /etc/security/keytabs/nn.service.keytab nn/<HDFS_USER>@<HDFS_USER>;hdfs
  dfsadmin -safemode enter'
  sudo su -l <HDFS_USER> -c 'kinit -kt
  /etc/security/keytabs/nn.service.keytab nn/<HDFS_USER>@<HDFS_USER>;hdfs
  dfsadmin -saveNamespace'
  ```

  Where `<HDFS_USER>` is the HDFS service user; for example hdfs, `<HOSTNAME>` is the Active NameNode hostname, and `<REALM>` is your Kerberos realm.

## Stop All Services

Browse to **Ambari Web > Services**, then choose **Stop All** in the Services navigation panel. You must wait until all the services are completely stopped.

## Prepare the Ambari Server Host for Rollback

Log into the Ambari server host and set the following environment variables to prepare for the rollback procedure:

| Variable | Value |
|---|---|
| export AMBARI_USER=AMBARI_USERNAME | Substitute the value of the administrative user for Ambari Web. The default value is admin. |
| export AMBARI_PW=AMBARI_PASSWORD | Substitute the value of the administrative password for Ambari Web. The default value is admin. |
| export AMBARI_PORT=AMBARI_PORT | Substitute the Ambari Web port. The default value is 8080. |
| export AMBARI_PROTO=AMBARI_PROTOCOL | Substitute the value of the protocol for connecting to Ambari Web. Options are http or https. The default value is http. |
| export CLUSTER_NAME=CLUSTER_NAME | Substitute the name of your cluster, set during the Ambari Install Wizard process. For example: mycluster. |

| | |
|---|---|
| export NAMENODE_HOSTNAME=NN_HOSTNAME | Substitute the FQDN of the host for the non-HA NameNode. For example: nn01.mycompany.com. |
| export ADDITIONAL_NAMENODE_HOSTNAME=ANN_HOSTNAME | Substitute the FQDN of the host for the additional NameNode in your HA setup. |
| export SECONDARY_NAMENODE_HOSTNAME=SNN_HOSTNAME | Substitute the FQDN of the host for the standby NameNode for the non-HA setup. |
| export JOURNALNODE1_HOSTNAME=JOUR1_HOSTNAME | Substitute the FQDN of the host for the first Journal Node. |
| export JOURNALNODE2_HOSTNAME=JOUR2_HOSTNAME | Substitute the FQDN of the host for the second Journal Node. |
| export JOURNALNODE3_HOSTNAME=JOUR3_HOSTNAME | Substitute the FQDN of the host for the third Journal Node. |

Double check that these environment variables are set correctly.

## Restore the HBase Configuration

If you have installed HBase, you may need to restore a configuration to its pre-HA state.

1 To check if your current HBase configuration needs to be restored, on the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> hbase-site
```

Where the environment variables you set up in Prepare the Ambari Server Host for Rollback substitute for the variable names.

Look for the configuration property `hbase.rootdir`. If the value is set to the NameService ID you set up using the **Enable NameNode HA** wizard, you need to revert the `hbase-site` configuration set up back to non-HA values. If it points instead to a specific NameNode host, it does not need to be rolled back and you can go on to Delete ZooKeeper Failover Controllers.

For example:

```
"hbase.rootdir":"hdfs://<name-service-id>:8020/apps/hbase/data"
```

The hbase.rootdir property points to the NameService ID and the value needs to be rolled back

```
"hbase.rootdir":"hdfs://<nn01.mycompany.com>:8020/apps/hbase/data"
```

The hbase.rootdir property points to a specific NameNode host and not a NameService ID. This does not need to be rolled back.

2 If you need to roll back the `hbase.rootdir` value, on the Ambari Server host, use the `config.sh` script to make the necessary change:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> set localhost <CLUSTER_NAME> hbase-site
hbase.rootdir hdfs://<NAMENODE_HOSTNAME>:8020/apps/hbase/data
```

Where the environment variables you set up in Prepare the Ambari Server Host for Rollback substitute for the variable names.

3 Verify that the `hbase.rootdir` property has been restored properly. On the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> hbase-site
```

The `hbase.rootdir` property should now be set to the NameNode hostname, not the NameService ID.

## Delete ZooKeeper Failover Controllers

You may need to delete ZooKeeper (ZK) Failover Controllers.

1 To check if you need to delete ZK Failover Controllers, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=ZKFC
```

If this returns an empty `items` array, you may proceed to Modify HDFS Configuration. Otherwise you must use the following DELETE commands:

2 To delete all ZK Failover Controllers, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
DELETE
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/hosts/<NAMENODE_HOSTNAME>/host_components/ZKFC

curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
DELETE
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/hosts/<ADDITIONAL_NAMENODE_HOSTNAME>/host_components/ZKFC
```

3 Verify that the ZK Failover Controllers have been deleted. On the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=ZKFC
```

This command should return an empty `items` array.

## Modify HDFS Configurations

You may need to modify your `hdfs-site` configuration and/or your `core-site` configuration.

1 To check if you need to modify your `hdfs-site` configuration, on the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> hdfs-site
```

If you see **any** of the following properties, you must delete them from your configuration.

- `dfs.nameservices`
- `dfs.client.failover.proxy.provider.<NAMESERVICE_ID>`
- `dfs.ha.namenodes.<NAMESERVICE_ID>`
- `dfs.ha.fencing.methods`
- `dfs.ha.automatic-failover.enabled`
- `dfs.namenode.http-address.<NAMESERVICE_ID>.nn1`
- `dfs.namenode.http-address.<NAMESERVICE_ID>.nn2`
- `dfs.namenode.rpc-address.<NAMESERVICE_ID>.nn1`
- `dfs.namenode.rpc-address.<NAMESERVICE_ID>.nn2`
- `dfs.namenode.shared.edits.dir`
- `dfs.journalnode.edits.dir`
- `dfs.journalnode.http-address`
- `dfs.journalnode.kerberos.internal.spnego.principal`
- `dfs.journalnode.kerberos.principal`
- `dfs.journalnode.keytab.file`

Where `<NAMESERVICE_ID>` is the NameService ID you created when you ran the **Enable NameNode HA** wizard.

2 To delete these properties, execute the following **for each property** you found. On the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> delete localhost <CLUSTER_NAME> hdfs-
site property_name
```

Where you replace `property_name` with the name of **each** of the properties to be deleted.

3 Verify that all of the properties have been deleted. On the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> hdfs-site
```

None of the properties listed above should be present.

4 To check if you need to modify your `core-site` configuration, on the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> core-site
```

5 If you see the property `ha.zookeeper.quorum`, it must be deleted. On the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> delete  localhost <CLUSTER_NAME> core-
site ha.zookeeper.quorum
```

6 If the property `fs.defaultFS` is set to the NameService ID, it must be reverted back to its non-HA value. For example:

```
"fs.defaultFS":"hdfs://<name-service-id>"
The property fs.defaultFS needs to be modified as it points to a
NameService ID
"fs.defaultFS":"hdfs://<nn01.mycompany.com>"
```

The property `fs.defaultFS` does not need to be changed as it points to a specific NameNode, not to a NameService ID

7 To revert the property `fs.defaultFS` to the NameNode host value, on the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> set localhost <CLUSTER_NAME> core-site
fs.defaultFS hdfs://<NAMENODE_HOSTNAME>
```

8 Verify that the `core-site` properties are now properly set. On the Ambari Server host:

```
/var/lib/ambari-server/resources/scripts/configs.sh -u <AMBARI_USER> -p
<AMBARI_PW> -port <AMBARI_PORT> get localhost <CLUSTER_NAME> core-site
```

The property `fs.defaultFS` should be set to point to the NameNode host and the property `ha.zookeeper.quorum` should not be there.

## Recreate the Standby NameNode

You may need to recreate your standby NameNode.

1 To check to see if you need to recreate the standby NameNode, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X GET
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=SECONDARY_NAMENODE
```

If this returns an empty `items` array, you must recreate your standby NameNode. Otherwise you can go on to Re-enable Standby NameNode.

2 Recreate your standby NameNode. On the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
POST -d '{"host_components" :
[{"HostRoles":{"component_name":"SECONDARY_NAMENODE"}]    }'
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/hosts?Hosts/host_name=<SECONDARY_NAMENODE_HOSTNAME>
```

3 Verify that the standby NameNode now exists. On the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X GET
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=SECONDARY_NAMENODE
```

This should return a non-empty `items` array containing the standby NameNode.

## Re-enable the Standby NameNode

To re-enable the standby NameNode, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
'{"RequestInfo":{"context":"Enable Secondary
NameNode"},"Body":{"HostRoles":{"state":"INSTALLED"}}}'<AMBARI_PROTO>://local
host:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>/hosts/<SECONDARY_NAMENODE_H
OSTNAME}/host_components/SECONDARY_NAMENODE
```

- If this returns 200, go to Delete All JournalNodes.
- If this returns 202, wait a few minutes and run the following on the Ambari Server host:

```
curl -u <AMBARI_USER>:${AMBARI_PW -H "X-Requested-By: ambari" -i -X
"<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME
>/host_components?HostRoles/component_name=SECONDARY_NAMENODE&fields=Ho
stRoles/state"
```

When `"state" : "INSTALLED"` is in the response, go on to the next step.

## Delete All JournalNodes

You may need to delete any JournalNodes.

1 To check to see if you need to delete JournalNodes, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X GET
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=JOURNALNODE
```

If this returns an empty `items` array, you can go on to Delete the Additional NameNode. Otherwise you must delete the JournalNodes.

2 To delete the JournalNodes, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
DELETE
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/hosts/<JOURNALNODE1_HOSTNAME>/host_components/JOURNALNODE

curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
DELETE
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/hosts/<JOURNALNODE2_HOSTNAME>/host_components/JOURNALNODE

curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
DELETE
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/hosts/<JOURNALNODE3_HOSTNAME>/host_components/JOURNALNODE
```

3 Verify that all the JournalNodes have been deleted. On the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X GET
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=JOURNALNODE
```

This should return an empty `items` array.

## Delete the Additional NameNode

You may need to delete your Additional NameNode.

1 To check to see if you need to delete your Additional NameNode, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X GET
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=NAMENODE
```

If the `items` array contains two NameNodes, the Additional NameNode must be deleted.

2 To delete the Additional NameNode that was set up for HA, on the Ambari Server host:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X
DELETE
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/hosts/<ADDITIONAL_NAMENODE_HOSTNAME>/host_components/NAMENODE
```

3 Verify that the Additional NameNode has been deleted:

```
curl -u <AMBARI_USER>:<AMBARI_PW> -H "X-Requested-By: ambari" -i -X GET
<AMBARI_PROTO>://localhost:<AMBARI_PORT>/api/v1/clusters/<CLUSTER_NAME>
/host_components?HostRoles/component_name=NAMENODE
```

This should return an `items` array that shows only one NameNode.

## Verify the HDFS Components

Make sure you have the correct components showing in HDFS.

1 Go to **Ambari Web UI > Services**, then select **HDFS.**

2 Check the Summary panel and make sure that the first three lines look like this:

- NameNode
- SNameNode
- DataNodes

You should **not** see any line for JournalNodes.

## Start HDFS

1 In the **Ambari Web UI**, select **Service Actions**, then choose **Start.**

Wait until the progress bar shows that the service has completely started and has passed the service checks.

If HDFS does not start, you may need to repeat the previous step.

2 To start all of the other services, select **Actions > Start All** in the **Services** navigation panel.

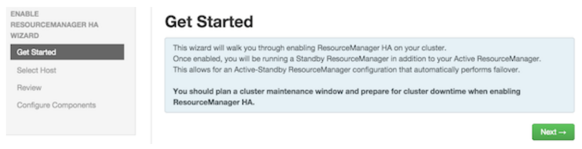
# Configuring ResourceManager High Availability

This feature is available with HDP Stack 2.2 or later.

The following topic explains How to set up ResourceManager High Availability.

## How to Set Up ResourceManager High Availability

1. Check to make sure you have at least three hosts in your cluster and are running at least three ZooKeeper servers.
2. In Ambari Web, browse to **Services > YARN > Summary.** Select **Service Actions** and choose **Enable ResourceManager HA.**
3. The Enable ResourceManager HA Wizard launches. The wizard describes a set of automated and manual steps you must take to set up ResourceManager High Availability.
4. **Get Started**: This step gives you an overview of enabling ResourceManager HA. Click **Next** to proceed.

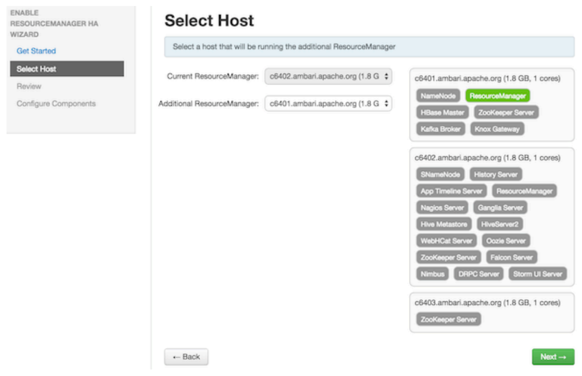


Get Started

This wizard will walk you through enabling ResourceManager HA on your cluster.
Once enabled, you will be running a Standby ResourceManager in addition to your Active ResourceManager.
This allows for an Active-Standby ResourceManager configuration that automatically performs failover.

**You should plan a cluster maintenance window and prepare for cluster downtime when enabling ResourceManager HA.**

5. **Select Host**: The wizard shows you the host on which the current ResourceManager is installed and suggests a default host on which to install an additional ResourceManager. Accept the default selection, or choose an available host. Click **Next** to proceed.



Select Host

Select a host that will be running the additional ResourceManager

Current ResourceManager: c6402.ambari.apache.org (1.8 G

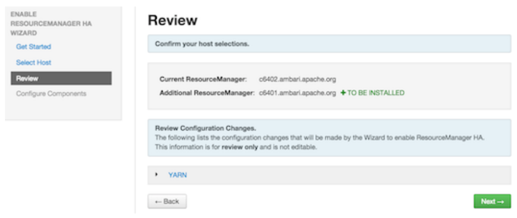
Additional ResourceManager: c6401.ambari.apache.org (1.8 G

c6401.ambari.apache.org (1.8 GB, 1 cores)
NameNode, ResourceManager, HBase Master, ZooKeeper Server, Kafka Broker, Knox Gateway

c6402.ambari.apache.org (1.8 GB, 1 cores)
SNameNode, History Server, App Timeline Server, ResourceManager, Nagios Server, Ganglia Server, Hive Metastore, HiveServer2, WebHCat Server, Oozie Server, ZooKeeper Server, Falcon Server, Nimbus, DRPC Server, Storm UI Server

c6403.ambari.apache.org (1.8 GB, 1 cores)
ZooKeeper Server

6. **Review Selections:** The wizard shows you the host selections and configuration changes that will occur to enable ResourceManager HA. Expand YARN, if necessary, to review all the YARN configuration changes. Click **Next** to approve the changes and start automatically configuring ResourceManager HA.



Review

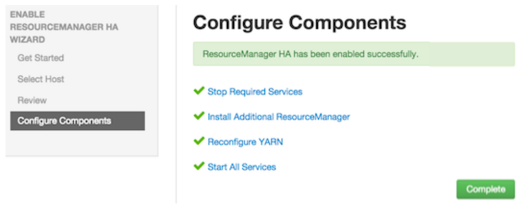
Confirm your host selections.

Current ResourceManager: c6402.ambari.apache.org
Additional ResourceManager: c6401.ambari.apache.org + TO BE INSTALLED

Review Configuration Changes.
The following lists the configuration changes that will be made by the Wizard to enable ResourceManager HA.
This information is for **review only** and is not editable.

YARN

7 **Configure Components:** The wizard configures your components automatically, displaying progress bars to let you track the steps. After all progress bars complete, click **Complete** to finish the wizard.



Configure Components

ResourceManager HA has been enabled successfully.

✓ Stop Required Services
✓ Install Additional ResourceManager
✓ Reconfigure YARN
✓ Start All Services

# Configuring RHEL HA for Hadoop 1.x

Ambari supports High Availability of components such as NameNode or JobTracker in a HDP 1.x cluster running RHEL HA. After installing NameNode monitoring components on hosts in an HA cluster, as described in HDP System Administration, configure Ambari to reassign any component on a failover host in the cluster, using the host_relocate_component.py script.

For example, if the host for the primary NameNode or JobTracker component fails, Ambari reassigns the primary NameNode or JobTracker component to the configured failover host, when you start or restart Ambari server.

To configure RHEL HA for an Hadoop 1.x, do the following tasks:

- Deploy the scripts
- Configure Ambari properties across the HA cluster
- Troublshoot RHEL HA, if necessary

## Deploy the scripts

While the Ambari server and ambari agents are running on each host:

1. Download relocate_host_component.py from `/var/lib/ambari-server/resources/scripts` on the Ambari server to `/usr/bin/` on each failover host.
2. Download **hadoop.sh** from `/var/lib/ambari-server/resources/scripts` on the Ambari server and replace hadoop.sh in /usr/share/cluster/ on each failover host.

## Configure Ambari properties across the HA cluster

To enable Ambari to run `relocate_host_component.py`, use a text editor to edit the cluster configuration file on each failover host in the HA cluster.

In `/etc/cluster/cluster.conf`, set values for each of the following properties:

- `<server>=<ambari-hostname / ip>`
- `<port>=<8080>`
- `<protocol>=<http / https>`
- `<user>=<admin>`
- `<password>=<admin>`
- `<cluster>=<cluster-name>`
- `<output>=</var/log/ambari_relocate.log>`

For example, the Hadoop daemon section of `cluster.conf` on the NameNode localhost in an HA cluster will look like:

```
<hadoop__independent_subtree="1" __max_restarts="10"
__restart_expire_time="600" name="NameNode Process"
 daemon="namenode" boottime="10000" probetime="10000" stoptime="10000"
url="http://10.0.0.30:50070/dfshealth.jsp"
 pid="/var/run/hadoop/hdfs/hadoop-hdfs-namenode.pid" path="/"

ambariproperties="server=localhost,port=8080,protocol=http,user=admin,passwor
d=admin,cluster=c1,output=/var/log/ambari_relocate.log"/>
```

The `relocate_host_component.py` script reassigns components on failover of any host in the HA cluster, when you start or restart Ambari server.

## Troubleshooting RHEL HA

1. Review errors in `/var/log/messages/`.
2. If the following error message appears:

   ```
   abrtd: Executable '/usr/bin/relocate_resources.py' doesn't belong to
   any package and ProcessUnpackaged is set to 'no'
   ```

   Set the following property, in `/etc/abrt/abrt-action-save-package-data.conf`,

   ```
   set ProcessUnpackaged=Yes
   ```
3. If the scripts return Error status=exit code 3, make sure the following are true:
   - The ambari agent on the failover host is running.
   - Failover did not result from STOP HDFS or STOP NN/JT, using Ambari.

The following table lists and describes parameters for `relocate_host_components.py`.

| Parameter | Value | Example | Description |
|---|---|---|---|
| -h, | na | --help | Display all parameter options. |
| -v, | na | --verbose | Increases output verbosity. |
| -s, | SERVER_HOSTNAME, | --host=SERVER_HOSTNAME | Ambari server host name (FQDN) |
| -p, | SERVER_PORT, | --port=SERVER_PORT | Ambari server port. [default: 8080] |
| -r, | PROTOCOL, | --protocol=PROTOCOL | Protocol for communicating with Ambari server (http/https) [default: http] |
| -c, | CLUSTER_NAME, | --cluster-name=CLUSTER_NAME | Ambari cluster to operate on. |
| -e, | SERVICE_NAME, | --service-name=SERVICE_NAME | Ambari Service to which the component belongs. |

| | | | |
|---|---|---|---|
| -m, | COMPONENT_NAME, | --component-name=COMPONENT_NAME | Ambari Service Component to operate on. |
| -n, | NEW_HOSTNAME, | --new-host=NEW_HOSTNAME | New host to relocate the component to. |
| -a, | ACTION, | --action=ACTION | Script action. [default: relocate] |
| -o, | FILE, | --output-file=FILE | Output file. [default: /temp/ambari_reinstall_probe.out ] |
| -u, | USERNAME, | --username=USERNAME | Ambari server admin user. [default: admin] |
| -w, | PASSWORD, | --password=PASSWORD | Ambari server admin password. |
| -d, | COMPONENT_NAME, | --start-component | Start the component after reassignment. |

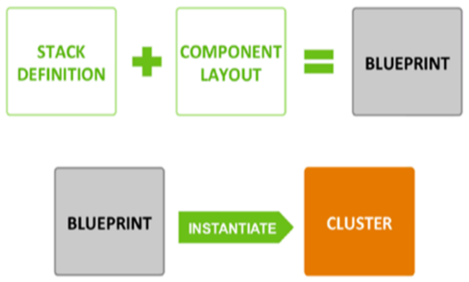
# Using Ambari Blueprints

Ambari Blueprints provide an API to perform cluster installations. You can build a reusable “blueprint” that defines which Stack to use, how Service Components should be laid-out across a cluster, and what configurations to set.

Overview: Ambari Blueprints

## Overview: Ambari Blueprints

Ambari Blueprints provide an API to perform cluster installations. You can build a reusable “blueprint” that defines which Stack to use, how Service Components should be laid-out across a cluster and what configurations to set.



After setting up a blueprint, you can call the API to instantiate the cluster by providing the list of hosts to use. The Ambari Blueprint framework promotes reusability and facilitates automating cluster installations without UI interaction.

Learn more about Ambari Blueprints API on the Ambari Wiki.

# Configuring HDP Stack Repositories for Red Hat Satellite

As part of installing HDP Stack with Ambari, `HDP.repo` and `HDP-UTILS.repo` files are generated and distributed to the cluster hosts based on the Base URL user input from the Cluster Install Wizard during the Select Stack step. In cases where you are using Red Hat Satellite to manage your Linux infrastructure, you can disable the repositories defined in the HDP Stack .repo files and instead leverage Red Hat Satellite.

How To Configure HDP Stack Repositories for Red Hat Satellite

## How To Configure HDP Stack Repositories for Red Hat Satellite

To disable the repositories defined in the HDP Stack.repo files:

1 Before starting the Ambari Server and installing a cluster, on the Ambari Server browse to the Stacks definition directory.

```
cd /var/lib/ambari-server/resources/stacks/
```

2 Browse the install hook directory:

**For HDP 2.0 or HDP 2.1 Stack**

```
cd HDP/2.0.6/hooks/before-INSTALL/templates
```

**For HDP 1.3 Stack**

```
cd HDP/1.3.2/hooks/before-INSTALL/templates
```

3 Modify the.repo template file

```
vi repo_suse_rhel.j2
```

4 Set the enabled property to 0 to disable the repository.

```
enabled=0
```

5 Save and exit.

6 Start the Ambari Server and proceed with your install.

The .repo files will still be generated and distributed during cluster install but the repositories defined in the .repo files will not be enabled.

⚠ You must configure Red Hat Satellite to define and enable the Stack repositories. Please refer to the Red Hat Satellite documentation for more information.

## Configuring Storm for Supervision

Ambari administrators should install and configure a process controller to monitor and run Apache Storm under supervision. Storm is fail-fast application, meaning that it is designed to fail under certain circumstances, such as a runtime exception or a break in network connectivity. Without a watchdog process, these events can quickly take down an entire Storm cluster in production. A watchdog process prevents this by monitoring for failed Storm processes and restarting them when necessary. This topic describes how to configure `supervisord` to manage the Storm processes, but you may choose to use another process controller, such as `monit` or `daemontools`.

> Running Storm under supervision is only supported for Nimbus Server and Supervisors.

How To Configure Storm for Supervision

### How To Configure Storm for Supervision

To configure Storm for operating under supervision:

1 Stop all Storm components.

   Using **Ambari Web Services > Storm > Service Actions**, choose **Stop**, then wait until stop completes.

2 Stop Ambari Server.

   ```
   ambari-server stop
   ```

3 Change Supervisor and Nimbus command scripts in the Stack definition.

   On Ambari Server host, run:

   ```
   sed -ir
   "s/scripts\/supervisor.py/scripts\/supervisor_prod.py/g"/var/lib/ambari
   -server/resources/stacks/HDP/2.1/services/STORM/metainfo.xml
   sed -ir "s/scripts\/nimbus.py/scripts\/nimbus_prod.py/g"
   /var/lib/ambari-
   server/resources/stacks/HDP/2.1/services/STORM/metainfo.xml
   ```

4 Install `supervisord` on all Nimbus and Supervisor hosts.

   1 Install EPEL repository.

      ```
      yum install epel-release -y
      ```

   2 Install supervisor package for `supervisord`.

      ```
      yum install supervisor -y
      ```

3. **Enable `supervisord` on autostart.**

   ```
   chkconfig supervisord on
   ```

4. **Change `supervisord` configuration file permissions.**

   ```
   chmod 600 /etc/supervisord.conf
   ```

5. **Configure `supervisord` to supervise Nimbus Server and Supervisors.**

   **Append the following to `/etc/supervisord.conf` on all Supervisor host and Nimbus hosts accordingly.**

   ```
   [program:storm-nimbus]
   command=env PATH=$PATH:/bin:/usr/bin/:/usr/jdk64/jdk1.7.0_45/bin/
   JAVA_HOME=/usr/jdk64/jdk1.7.0_45 /usr/hdp/current/storm-
   nimbus/bin/storm nimbus
   user=storm
   autostart=true
   autorestart=true
   startsecs=10
   startretries=999
   log_stdout=true
   log_stderr=true
   logfile=/var/log/storm/nimbus.out
   logfile_maxbytes=20MB
   logfile_backups=10


   [program:storm-supervisor]
   command=env PATH=$PATH:/bin:/usr/bin/:/usr/jdk64/jdk1.7.0_45/bin/
   JAVA_HOME=/usr/jdk64/jdk1.7.0_45 /usr/hdp/current/storm-
   supervisor/bin/storm supervisor
   user=storm
   autostart=true
   autorestart=true
   startsecs=10
   startretries=999
   log_stdout=true
   log_stderr=true
   logfile=/var/log/storm/supervisor.out
   logfile_maxbytes=20MB
   logfile_backups=10
   ```

   Change /usr/jdk64/jdk1.7.0_45 accordingly to the location of the jdk for Ambari in your environment

6. **Start Ambari Server.**

```
ambari-server start
```

# Tuning Ambari Performance

For clusters larger than 200 nodes, calculate and set a larger task cache size on the Ambari server. Also, enable Nagios macros appropriate for the HDP Stack version.

## How To Tune Ambari Performance

For clusters larger than 200 nodes:

1. Calculate the new, larger cache size, using the following relationship:

   ```
   ecCacheSizeValue=60*<cluster_size>
   ```
   where `<cluster_size>` is the number of nodes in the cluster.

2. On the Ambari Server host, in `etc/ambari-server/conf/ambari-properties`, add the following property and value:

   ```
   server.ecCacheSize=<ecCacheSizeValue>
   ```
   where `<ecCacheSizeValue>` is the value calculated previously, based on the number of nodes in the cluster.

3. On Ambari Server host, make the following changes:

   ```
   -enable_environment_macros=1
   +enable_environment_macros=0
   ```

   - For HDP2, make this change in `/var/lib/ambari-server/resources/stacks/HDP/2.0.6/services/NAGIOS/package/templates/nagios.cfg.j2`
   - For HDP1, make this change in `/var/lib/ambari-server/resources/stacks/HDP/1.3.2/services/NAGIOS/package/templates/nagios.cfg.j2`

4. Restart Ambari Server.

   ```
   ambari-server restart
   ```

5. Restart Nagios.

   Using **Ambari Web > Services > Nagios > Service Actions**, choose **Restart All.**

# Refreshing YARN Capacity Scheduler

After you modify the Capacity Scheduler configuration, YARN supports refreshing the queues without requiring you to restart your ResourceManager. The “refresh” operation is valid if you have made no destructive changes to your configuration. Removing a queue is an example of a destructive change.

## How to refresh the YARN Capacity Scheduler

This topic describes how to refresh the Capacity Scheduler in cases where you have added or modified existing queues.

- In Ambari Web, browse to **Services > YARN > Summary.**
- Select **Service Actions**, then choose **Refresh YARN Capacity Scheduler.**
- Confirm you would like to perform this operation.

  The refresh operation is submitted to the YARN ResourceManager.

⚠ The Refresh operation will fail with the following message: “Failed to re-init queues” if you attempt to refresh queues in a case where you performed a destructive change, such as removing a queue. In cases where you have made destructive changes, you must perform a ResourceManager restart for the capacity scheduler change to take effect.

# Rebalancing HDFS

HDFS provides a “balancer” utility to help balance the blocks across DataNodes in the cluster.

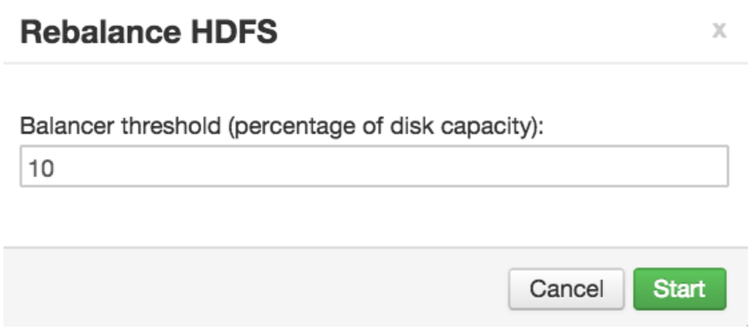
## How to rebalance HDFS

This topic describes how you can initiate an HDFS rebalance from Ambari.

1 . In Ambari Web, browse to **Services > HDFS > Summary.**

2 Select **Service Actions**, then choose **Rebalance HDFS.**

3 Enter the Balance Threshold value as a percentage of disk capacity.


Rebalance HDFS

Balancer threshold (percentage of disk capacity): 10

4 Click **Start** to begin the rebalance.

5 You can check rebalance progress or cancel a rebalance in process by opening the Background Operations dialog.