Hortonworks Data Platform

Ambari Reference Topics

(Jul 15, 2014)

docs.hortonworks.com

Hortonworks Data Platform : Ambari Reference Topics

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

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source. Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

List of Tables

1. Installing Ambari Agents Manually

In some situations you may decide you do not want to have the Ambari Install Wizard install and configure the Agent software on your cluster hosts automatically. In this case you can install the software manually.

Before you begin: On every host in your cluster, download the Ambari repository as described in Set Up the Bits.

1.1. RHEL/CentOS/Oracle Linux 5.x and 6.x

1. Install the Ambari Agent

```
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.

```
ambari-agent start
```

The agent registers with the Server on start.

1.2. SLES

1. Install the Ambari Agent.

```
zypper install ambari-agent
```

2. 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.

```
ambari-agent start
```

The agent registers with the Server on start.

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

2.1. 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.

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

The individual services in Hadoop are each run under the ownership of a corresponding Unix account. These accounts are known as service users. These service users belong to a special Unix group. In addition there is a special service user for running smoke tests on components during installation and on-demand using the **Services** View of the Ambari Web GUI. Any of these users and groups can be customized using the **Misc** tab during the **Customize Services** installation step.

Note

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.

Note

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

Table 2.1. Service Users

Service	Component	Default User Account
HDFS	NameNode SecondaryNameNode DataNode	hdfs
MapReduce	JobTracker HistoryServer TaskTracker	mapred

Service	Component	Default User Account
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[a]
Smoke Test[b]	All	ambari-qa

[a]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".

[b]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 2.2. Service Group

Service	Components	Default Group Account
All	All	hadoop

2.1.2. 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 2.3. 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 2.4. 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.

2.1.3. 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 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`

2.2. 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.

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

The individual services in Hadoop are each run under the ownership of a corresponding Unix account. These accounts are known as service users. These service users belong to a special Unix group. In addition there is a special service user for running smoke tests on components during installation and on-demand using the **Services** View of the Ambari Web GUI. Any of these users and groups can be customized using the **Misc** tab during the **Customize Services** installation step.

Note

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.

Note

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

Table 2.5. Service Users

Service	Component	Default User Account
HDFS	NameNode SecondaryNameNode DataNode	hdfs
YARN	NodeManager ResourceManager	yarn
MapReduce2	HistoryServer	mapred
Tez	Tez clients	tez[a]
HBase	MasterServer RegionServer	hbase
Hive	Hive Metastore HiveServer2	hive
HCat	HCatalog Server	hcat
WebHCat	WebHCat Server	hcat
Falcon	Falcon Server	falcon[b]
Storm	Masters (Nimbus, DRPC Server, Storm REST API, Server, Storm UI Server) Slaves (Supervisors, Logviewers)	storm[c]
Oozie	Oozie Server	oozie
Ganglia	Ganglia Server Ganglia Monitors	nobody
Ganglia	RRDTool (with Ganglia Server)	rrdcahed[d]
Ganglia	Apache HTTP Server	apache[e]
PostgreSQL	PostgreSQL (with Ambari Server)	postgres[f]
Nagios	Nagios Server	nagios[g]
ZooKeeper	ZooKeeper	zookeeper
Smoke Test[h]	All	ambari-qa

[a]Tez is available with HDP 2.1 Stack. Not applicable in HDP 2.0 Stack or earlier.

[b]Falcon is available with HDP 2.1 Stack. Not applicable in HDP 2.0 Stack or earlier.

[c]Storm is available with HDP 2.1 Stack. Not applicable in HDP 2.0 Stack or earlier.

[d]Created as part of installing RRDTool, which is used to store metrics data collected by Ganglia.

[e]Created as part of installing Apache HTTP Server, which is used to serve the Ganglia web UI.

[f]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.

[g]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.

[h]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 2.6. Service Group

Service	Components	Default Group Account
All	All	hadoop
Nagios	Nagios Server	nagios
Ganglia	Ganglia Server Ganglia Monitor	nobody

2.2.2. 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 2.7. 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 2.8. MapReduce Settings: Advanced

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

3. Using Custom Host Names

You can customize the agent registration host name and the public host name used for each host in Ambari. Use this capability when "hostname" does not return the public network host name for your machines.

To customize the name of each host in your cluster:

1. On the **Install Options** screen, select **Perform Manual Registration** for Ambari Agents.
2. Install the Agents manually, as described in Installing Ambari Agents Manually.
3. To echo the customized name of the host to which the Ambari agent registers, for every host, create a script like the following example, named `/var/lib/ambari-agent/hostname.sh`. Be sure to `chmod` the script so it is executable by the Agent.

   ```
   #!/bin/sh

   echo ambari_hostname
   ```

   where ambari_hostname is the hostname to use for Agent registration
4. Open `/etc/ambari-agent/conf/ambari-agent.ini` on every host, using a text editor.
5. Add to the `[agent]` section the following line:

   ```
   hostname_script=/var/lib/ambari-agent/hostname.sh
   ```

   where`/var/lib/ambari-agent/hostname.sh` is the name of your custom echo script.
6. To generate a public host name for every host, create a script like the following example, named `/var/lib/ambari-agent/public_hostname.sh` to show the name for that host in the UI. Be sure to `chmod` the script so it is executable by the Agent.

   ```
   #!/bin/sh
   hostname -f
   ```
7. Open `/etc/ambari-agent/conf/ambari-agent.ini` on every host, using a text editor.
8. Add to the `[agent]` section the following line:

   ```
   public_hostname_script=/var/lib/ambari-agent/public_hostname.sh
   ```
9. If applicable, add the host names to `/etc/hosts` on every host.
10. Restart the Agent for these changes to take effect.

    ```
    ambari-agent restart
    ```

4. Moving the Ambari Server

Use the following instructions to transfer the Ambari Server to a new host.

Note

These steps describe moving the Ambari Server when it uses the default PostgreSQL database. If you are using a non-default database such as Oracle for Ambari, adjust the database backup, restore, and stop/start procedures to match that database.

1. Back up all current data from the original Ambari Server and MapReduce databases.
2. Update all Agents to point to the new Ambari Server.
3. Install the Server on a new host and populate databases with information from original Server.

4.1. Back up Current Data

1. Stop the original Ambari Server.

```
ambari-server stop
```

2. Create a directory to hold the database backups.

```
cd /tmp
mkdir dbdumps
cd dbdumps/
```

3. Create the database backups.

```
pg_dump -U $AMBARI-SERVER-USERNAME ambari > ambari.sql Password: $AMBARI-SERVER-PASSWORD
pg_dump -U $MAPRED-USERNAME ambarirca > ambarirca.sql Password: $MAPRED-PASSWORD
```

Substitute the values you set up at installation for *$AMBARI-SERVER-USERNAME*, *$MAPRED-USERNAME*, *$AMBARI-SERVER-PASSWORD*, and *$MAPRED-PASSWORD*. Defaults are `ambari-server/bigdata` and `mapred/mapred`.

4.2. Update Agents

1. On each agent host, stop the agent.

```
ambari-agent stop
```

2. Remove old agent certificates.

```
rm /var/lib/ambari-agent/keys/*
```

3. Using a text editor, edit `/etc/ambari-agent/conf/ambari-agent.ini` to point to the new host.

```
[server]
hostname=$NEW FULLY.QUALIFIED.DOMAIN.NAME
url_port=8440
secured_url_port=8441
```

4.3. Install the New Server and Populate the Databases

1. Install the Server on the new host.
2. Stop the Server so that you can copy the old database data to the new Server.

   ```
   ambari-server stop
   ```

3. Restart the PostgreSQL instance.

   ```
   service postgresql restart
   ```

4. Open the PostgreSQL interactive terminal.

   ```
   su - postgres
   psql
   ```

5. Using the interactive terminal, drop the databases created by the fresh install.

   ```
   drop database ambari;
   drop database ambarirca;
   ```

6. Check to make sure the databases have been dropped.

   ```
   /list
   ```

   The databases should not be listed.

7. Create new databases to hold the transferred data.

   ```
   create database ambari;
   create database ambarirca;
   ```

8. Exit the interactive terminal.

   ```
   ^d
   ```

9. Copy the saved data from Back up Current Data to the new Server.

   ```
   cd /tmp
   scp -i <ssh-key> root@<original-server>/tmp/dbdumps/*.sql/tmp
   (Note: compress/transfer/uncompress as needed from source to dest)
   psql -d ambari -f /tmp/ambari.sql
   psql -d ambarirca -f /tmp/ambarirca.sql
   ```

10. Start the new Server.

    ```
    <exit to root>
    ambari-server start
    ```

11. On each Agent host, start the Agent.

```
ambari-agent start
```

12. Open Ambari Web. Point your compatible browser to:

```
<new_Ambari_Server>:8080
```

13. Go to **Services** -> **MapReduce** and use the Management Header to **Stop** and **Start** the MapReduce service.

14. Start other services as necessary.

The new Server is ready to use.

5. 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.

5.1. Configure core-site.xml for LZO

Using Ambari Web, browse to Services > HDFS > Configs, perform the following steps to configure LZO:

1. Find the `io.compression.codecs` property key.
2. Append to the `io.compression.codecs` property key, the following value:

```
com.hadoop.compression.lzo.LzoCodec
```

3. Expand the `Custom core-site.xml` section.
4. Select **Add Property**.
5. Add to `Custom core-site.xml` the following properties:

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

6. Choose Save.
7. Stop, then Start the HDFS service.

Note

You **must** Stop, then Start the HDFS service for Ambari to install the necessary LZO packages. Performing a Restart or a Restart All will not start the required package install.

5.2. Running Compression with Hive Queries

5.2.1. Create LZO Files

1. Create LZO files as the output of the Hive query.
2. Use lzo 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.LzoCodec
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.LzoCodec;SET hive.exec.compress.output=true;SET mapreduce.
output.fileoutputformat.compress=true;"
```

5.2.2. 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 generated above

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>"
```

6. 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

6.1. 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

6.1.1. Using Ambari with Oracle

To set up Oracle for use with Ambari:

1. On the Ambari Server host, install the appropriate JDBC .jar file.
   a. Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.
   b. Select `Oracle Database 11g Release 2 - ojdbc6.jar`.
   c. Copy the .jar file to the Java share directory.

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

   d. Make sure the .jar file has the appropriate permissions - 644.
2. Create a user for Ambari and grant it permissions.
   - Using the Oracle database admin utility:

     ```
     # 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;
     QUIT;
     ```

- Where $AMBARIUSER is the Ambari user name and $AMBARIPASSWORD is the Ambari user password.

3. Load the Ambari Server database schema.

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

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

   b. 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 enter the information use the username/password credentials you created in step 2.

6.1.2. Using Ambari with MySQL

To set up MySQL for use with Ambari:

1. On the Ambari Server host, install the connector.

   a. Install the connector

   RHEL/CentOS/Oracle Linux

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

   SLES

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

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

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

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

2. Create a user for Ambari and grant it permissions.

   - 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.

6.1.3. 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.

6.1.4. Troubleshooting Ambari

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

6.1.4.1. 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.

6.1.4.1.1. Solution

Make sure the file is in the appropriate directory on the Ambari server and re-run `ambari-server setup`. See Step 1 above.

6.1.4.2. 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.

6.1.4.2.1. Solution

Confirm 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/ambaridb
```

6.1.4.3. 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: ORA01017:
```

```
invalid username/password; logon denied
```

You are using an invalid username/password.

6.1.4.3.1. Solution

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

6.1.4.4. 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: ORA00942:
table or view does not exist
```

The schema has not been loaded.

6.1.4.4.1. Solution

Confirm you have loaded the database schema. See Step 4 above.

6.2. 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

6.2.1. 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.
   a. Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.
   b. Select `Oracle Database 11g Release 2 - ojdbc6.jar` and download the file.
   c. Make sure the .jar file has the appropriate permissions - 644.
   d. 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.

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

   - When using **HDP 2.1 Stack**:

     ```
     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.

   - When using **HDP 2.0 Stack**:

     ```
     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.

   - When using **HDP 1.3 Stack**:

     ```
     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.

6.2.2. 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.

   a. Install the connector.

   RHEL/CentOS/Oracle Linux

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

   SLES

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

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

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

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

d. 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.

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

   - When using **HDP 2.1 Stack**:

```
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.

6.2.3. 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.

   a. Install the connector.

      RHEL/CentOS/Oracle Linux

      ```
      yum install postgresql-jdbc*
      ```

      SLES

      ```
      zypper install -y postgresql-jdbc
      ```

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

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

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

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

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

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

   e. 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.

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

   - When using **HDP 2.1 Stack**:

     ```
     # 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.

- When using **HDP 2.0 Stack**:

```
# 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.

- When using **HDP 1.3 Stack**:

```
# 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.

6.2.4. Troubleshooting Hive

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

6.2.4.1. 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.

6.2.4.1.1. Solution

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

6.2.4.2. 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.

6.2.4.2.1. 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.

6.2.4.3. 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.

6.2.4.3.1. Solution

Two workarounds are possible. Using Ambari Web, in **Hive Configs Advanced**:

- 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:

Table 6.1. Hive Security Authorization Settings

Property	Value
hive.security.authorization.manager	org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationP
hive.security.metastore.authorization.manager	org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationP
hive.security.authenticator.manager	org.apache.hadoop.hive.ql.security.ProxyUserAuthenticator

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

6.3. 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

6.3.1. 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.
   a. Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.
   b. Select `Oracle Database 11g Release 2 - ojdbc6.jar`.
   c. Make sure the .jar file has the appropriate permissions - 644.
   d. 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:

     ```
     # 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.

6.3.2. 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.

   - Install the connector.

     RHEL/CentOS/Oracle Linux

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

     SLES

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

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

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

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

   - 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.

6.3.3. 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.

   a. Install the connector.

   RHEL/CentOS/Oracle Linux

   ```
   yum install postgresql-jdbc*
   ```

   SLES

   ```
   zypper install -y postgresql-jdbc
   ```

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

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

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

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

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

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

   e. 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 Hive database name.

6.3.4. Troubleshooting Oozie

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

6.3.4.1. 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.

6.3.4.1.1. Solution

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

6.3.4.2. 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.

6.3.4.2.1. 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
```

7. 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.

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.

Note

If you plan to use local repositories, see Optional: Configure Ambari for 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.

8. 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 the following topics in the HDP Stack documentation:

- HDP 2.1 Stack - Configuring Ports
- HDP 2.0 Stack - Configuring Ports
- HDP 1.3 Stack - Configuring Ports

8.1. Default Network Port Numbers - Ambari

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

Table 8.1. Ambari Ports

Service	Servers	Default Ports Used
Ambari Server	Ambari Server host	8080[a]
Ambari Server	Ambari Server host	8440
Ambari Server	Ambari Server host	8441
Ambari Agent	All hosts running Ambari Agents	8670[c]

[a]See Optional: Change the Ambari Server Port for instructions on changing the default port.

[b]See Optional: Set Up HTTPS for Ambari Server for instructions on enabling HTTPS.

[c] 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.

8.2. Ganglia Ports

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

Table 8.2. Ganglia Ports

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	

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
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		http[a]			

[a]See Optional: Set Up HTTPS for Ganglia for instructions on enabling HTTPS.

8.3. Nagios Ports

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

Table 8.3. Nagios Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
Nagios Server	Nagios server host	80	http[a]	Nagios Web UI	No	

[a]See Optional: Set Up HTTPS for Nagios for instructions on enabling HTTPS.

8.4. 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.

Important

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>
```

9. 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.

Before changing the JDK version on a secure cluster, follow the instructions for Deploying JCE Policy Archives on the Abari Server.

Use the following procedure to change the JDK:

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: 1
   ```

   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 one of the following:

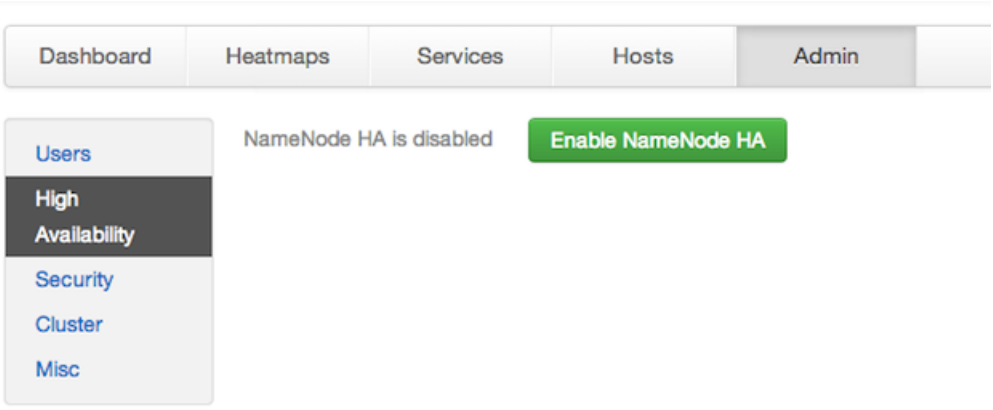
   - Restart each component
   - Restart each host
   - Restart all services

10. Configuring NameNode High Availability

Use these instructions to set up NameNode HA using Ambari Web.

10.1. Setting Up NameNode High Availability

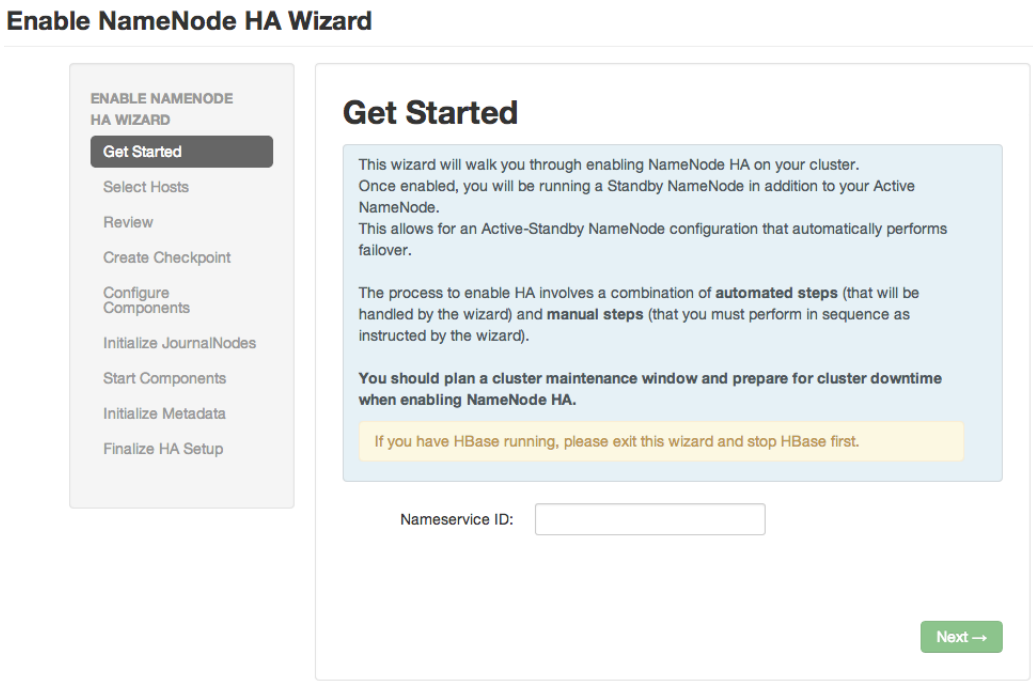
On Ambari Web, go to the Admin view. Select **High Availability** in the left navigation bar.



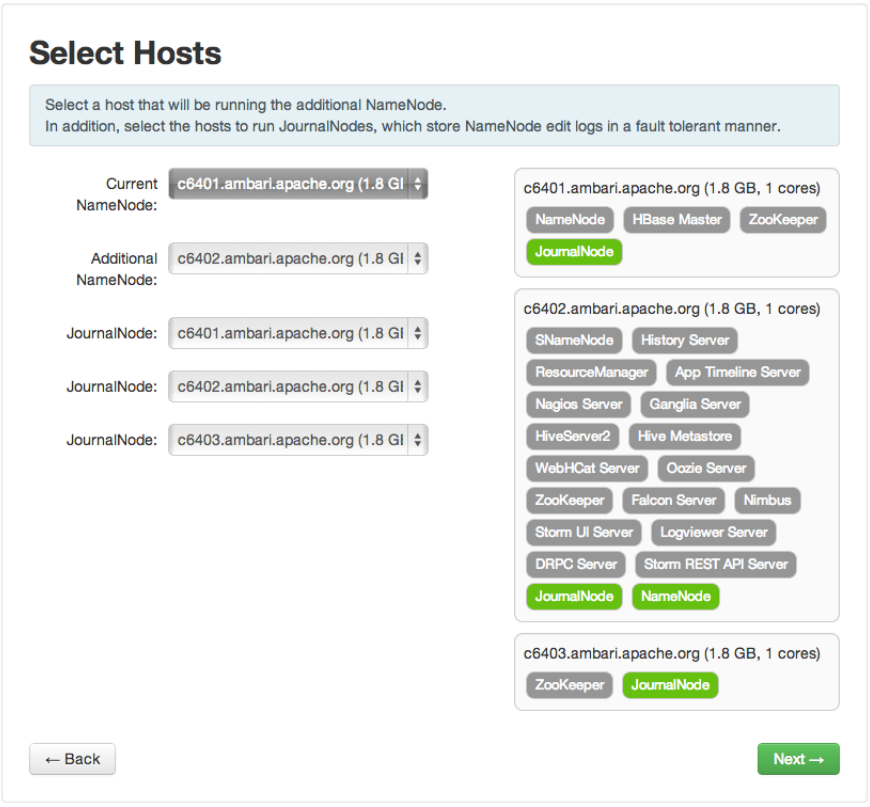
1. Check to make sure you have at least three hosts in your cluster and are running at least three ZooKeeper servers.

2. Click **Enable NameNode HA** and follow the **Enable NameNode HA Wizard**. The wizard describes a set of automated and manual steps you must take to set up NameNode high availability.

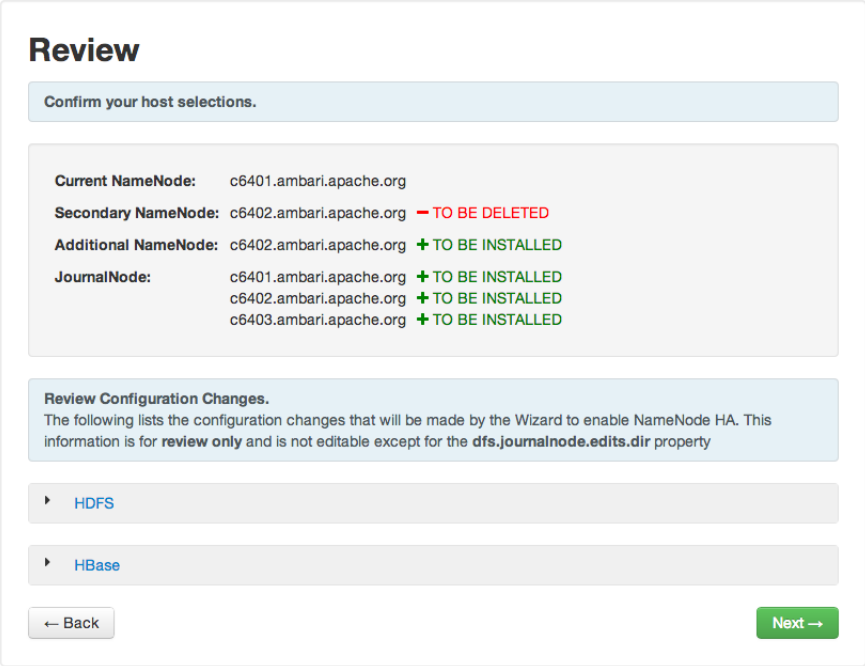
3. **Get Started**: This step gives you an overview of the process and allows you to select a Nameservice ID. You use this Nameservice ID instead of the NameNode FQDN once HA has been set up. Click **Next** to proceed.



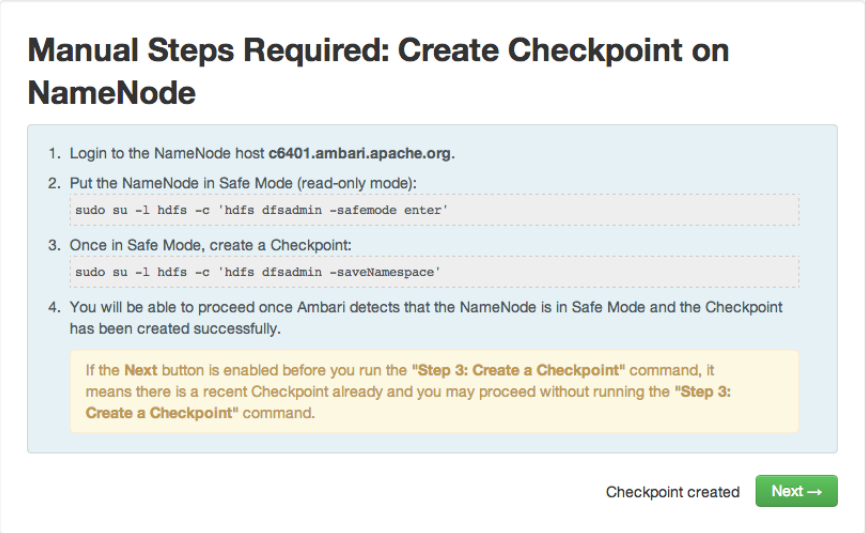
4. **Select Hosts**: Select a host for the additional NameNode and the JournalNodes. The wizard suggests options that you can adjust using the drop-down lists. Click **Next** to proceed.



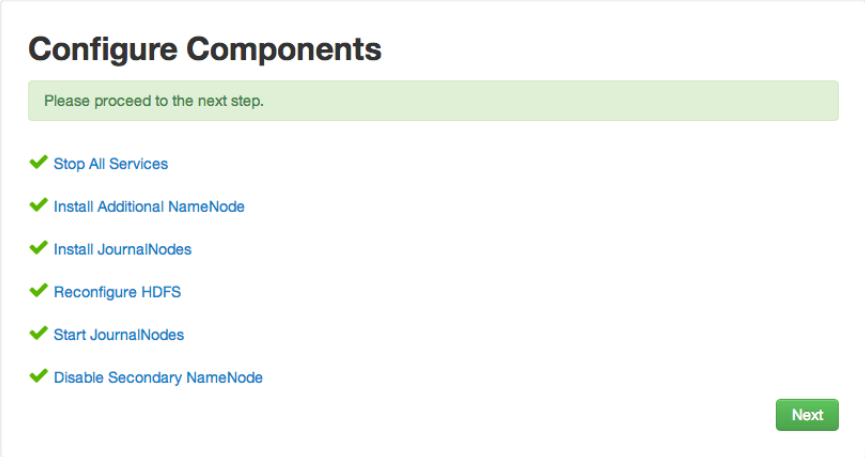
5. **Review**: Confirm your host selections and click **Next**.



6. **Create Checkpoints**: Follow the instructions in the step. You need to login 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**.



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

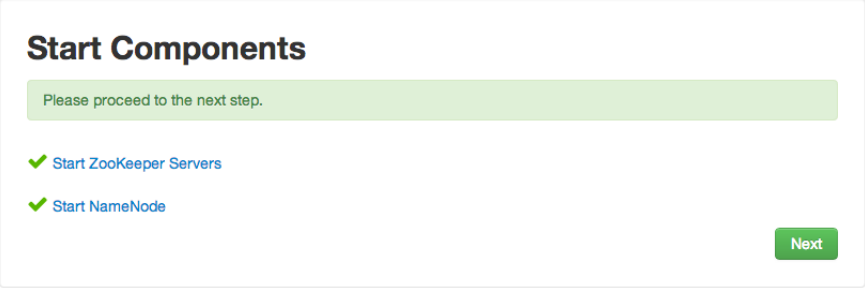


8. **Initialize JournalNodes**: Follow the instructions in the step. You need to login to your **current** NameNode host to run the command to initialize the JournalNodes. When Ambari detects success, the message on the bottom of the window changes. Click **Next**.

Important

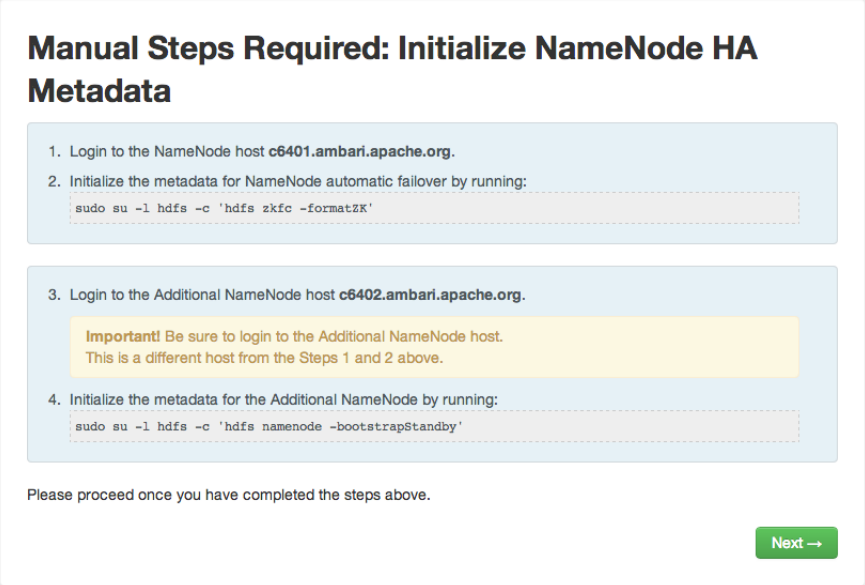
After upgrading to Ambari 1.6.1, or using a Blueprint to install your cluster, initializing JournalNodes may fail. For information about how to work around this issue, see the 1.6.1 Release Notes Known Issues or the Ambari Troubleshooting Guide, section 4.3.

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

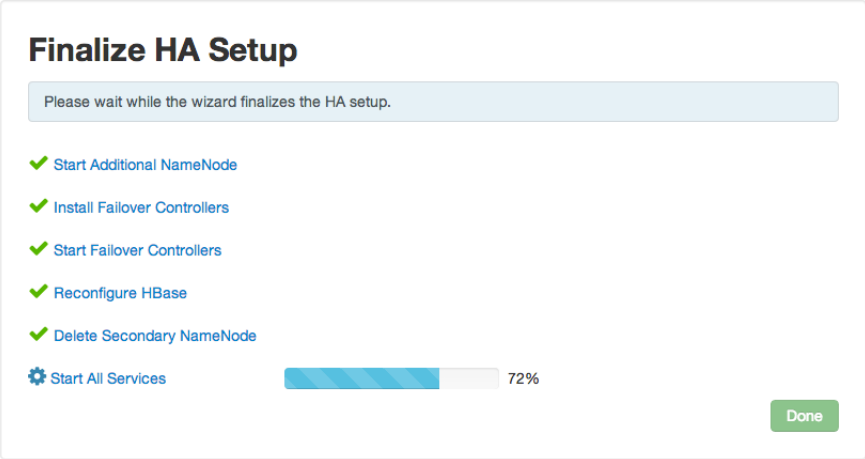


10. **Initialize Metadata**: Follow the instructions in the step. For this step you must login to both the **current** NameNode and the **additional** NameNode. Make sure you are logged into the correct host for each command. Click **Next** when you have completed the two

commands. A **Confirmation** pop-up window appears to remind you that you must do both steps. Click **OK** to confirm.



11. **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.



Note

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

12. 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.

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

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

The output might look like this:

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

b. 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 might look like this:

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

13. If you are using Oozie, you must use the Nameservice URI 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>
```

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

a. Install an httpfs server on any node in the cluster:

```
yum install hadoop-httpfs
```

b. 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>
```

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

Table 10.1. Core-site.xml properties and values for NameNode HA on a cluster using Hue

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

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

e. 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/
```

f. Restart the Hue service.

10.2. Rolling Back NameNode HA

To roll back NameNode HA to the previous non-HA state use the following step-by-step manual process. Some of the steps are optional depending on your installation.

1. Stop HBase
2. Checkpoint the Active NameNode
3. Stop All Services
4. Prepare the Ambari Server Host for Rollback
5. Restore the HBase Configuration
6. Delete ZK Failover Controllers
7. Modify HDFS Configurations
8. Recreate the Secondary NameNode
9. Re-enable the Secondary NameNode
10. Delete All JournalNodes
11. Delete the Additional NameNode
12. Verify the HDFS Components
13. Start HDFS

10.2.1. Stop HBase

1. From Ambari Web, go to the Services view and select HBase.
2. Click **Stop** on the Management Header.
3. Wait until HBase has stopped completely before continuing.

10.2.2. 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 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 (*$HDFS_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 $HDFS_USER -c 'kinit -kt /etc/security/keytabs/nn.service.keytab
 nn/$HOSTNAME@$REALM;hdfs dfsadmin -safemode enter'
sudo su -l $HDFS_USER -c 'kinit -kt /etc/security/keytabs/nn.service.keytab
 nn/$HOSTNAME@$REALM;hdfs  dfsadmin -saveNamespace'
```

  Where *$HDFS_USER* is the HDFS service user, *$HOSTNAME* is the Active NameNode hostname, and *$REALM* is your Kerberos realm.

10.2.3. Stop All Services

Use the Services view in Ambari Web and click **Stop All** in the Services navigation panel. You must wait until all the services are completely stopped.

10.2.4. 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:

Table 10.2. Set Environment Variables

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 Secondary NameNode for the non-HA setup.
export JOURNALNODE1_HOSTNAME=*JOUR1_HOSTNAME*	Substitute the FQDN of the host for the first Journal Node.

Variable	Value
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.

Important

Double check that these environment variables are set correctly.

10.2.5. 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 before 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 ZK 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 before 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.

10.2.6. Delete ZK Failover Controllers

You may need to delete 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 can go on to Modify HDFS Configuration. Otherwise you must use the DELETE commands below.

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.

10.2.7. 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 `hdfs-site` 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 and not 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.

10.2.8. Recreate the Secondary NameNode

You may need to recreate your Secondary NameNode.

1. To check to see if you need to recreate the Secondary 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 Secondary NameNode. Otherwise you can go on to Re-enable Secondary NameNode.

2. Recreate your Secondary 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 Secondary 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 Secondary NameNode.

10.2.9. Re-enable the Secondary NameNode

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

```
curl -u ${AMBARI_USER}:${AMBARI_PW} -H "X-Requested-By: ambari" -i -X PUT -d
 '{"RequestInfo":{"context":"Enable Secondary NameNode"},"Body":{"HostRoles":
{"state":"INSTALLED"}}}' ${AMBARI_PROTO}://localhost:${AMBARI_PORT}/api/v1/
clusters/${CLUSTER_NAME}/hosts/${SECONDARY_NAMENODE_HOSTNAME}/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 GET "${AMBARI_PROTO}://localhost:${AMBARI_PORT}/api/v1/clusters/
${CLUSTER_NAME}/host_components?HostRoles/component_name=SECONDARY_NAMENODE&
fields=HostRoles/state"
```

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

10.2.10. 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 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.

10.2.11. 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.

10.2.12. Verify your HDFS Components

Make sure you have the correct components showing in HDFS.

1. In Ambari Web, go to the Services view and select HDFS from the Services navigation panel.
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.

10.2.13. Start HDFS

1. On the HDFS page of the Services view, click **Start** in the Management Header to start up HDFS. Wait until the service is fully started and has passed the service check.

   If HDFS does not start, you may need to go through the previous steps again.
2. Start all the other services by using **Start All** in the Services navigation panel.

11. 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.

Note

Make sure that NameNode is installed. For successful reassignment, a component must be in one of the following states:

- Maintenance (started or stopped)
- Unknown

11.1. 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.

11.2. Configure Ambari properties across the HA cluster

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

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,
password=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.

11.3. 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.

Table 11.1. Parameter Options 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.

Parameter	Value	Example	Description
-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.

12. 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.

Then, you call the API to instantiate the cluster by providing the list of hosts to use.

This promotes reusability and facilitates automating cluster installations without UI interaction.

Learn more about Ambari Blueprints API on the Ambari Wiki.

13. 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.

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.

Important

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

14. Configuring Storm for Supervision

Ambari administrators should install and configure a process controller to monitor and run Apache Storm under supervision. Storm is a 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 section describes how to configure `supervisord` to manage the Storm processes, but adminstrators may use another process controller of their choice, such as `monit` or `daemontools` .

Note

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

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.

   a. Install EPEL repository.

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

   b. Install supervisor package for `supervisord`.

      ```
      yum install supervisor -y
      ```

   c. Enable `supervisord` on autostart.

      ```
      chkconfig supervisord on
      ```

   d. 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/lib/storm/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/lib/storm/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
```

Note

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
```

15. Tuning Ambari Performance for large (>2K-node) clusters

For an HDP cluster having more than 2000 nodes, we recommend that you tune performance in the following ways:

- Calculate and set a larger task cache size on the Ambari Server.
- Disable Nagios macros for large clusters to reduce the time Nagios spends processing macro definitions.
- Increase the maximum number of open files allowed on the Ganglia and Nagios hosts.

1. Calculate an appropriate cache size using the following formula:

   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}
   ```

3. On the Ambari Server host, disable environment macros for large clusters by making the following changes:

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

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

4. On Ganglia and Nagios hosts, in `/etc/security/limits.conf` replace the ulimit value with a higher number.

   Note

   Ganglia runs as user = nobody, by default. Nagios runs as user = nagios, by default.

5. Restart Ambari Server.

   ```
   ambari-server restart
   ```

6. Restart Nagios.

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