Hortonworks Data Platform

Ambari Reference Guide

(March 7, 2016)

docs.cloudera.com

Hortonworks Data Platform: Ambari Reference Guide

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

1. Installing Ambari Agents Manually ... 1
  1.1. Download the Ambari Repo ... 1
  1.2. Install the Ambari Agents Manually ... 5
2. Customizing HDP Services ... 7
  2.1. Defining Service Users and Groups for a HDP 2.x Stack ... 7
  2.2. Setting Properties That Depend on Service Usernames/Groups ... 8
3. Configuring Storm for Supervision ... 10
4. Using Custom Host Names ... 12
  4.1. How to Customize the name of a host ... 12
5. Moving the Ambari Server ... 13
  5.1. Back up Current Data ... 13
  5.2. Update all Agents ... 13
  5.3. Install the New Ambari Server ... 14
6. Configuring LZO Compression ... 16
  6.1. Configure core-site.xml for LZO ... 16
  6.2. Running Compression with Hive Queries ... 16
    6.2.1. Create LZO Files ... 17
    6.2.2. Write Custom Java to Create LZO Files ... 17
7. Using Non-Default Databases ... 18
  7.1. Using Non-Default Databases - Ambari ... 18
    7.1.1. Using Ambari with Oracle ... 18
    7.1.2. Using Ambari with MySQL ... 19
    7.1.3. Using Ambari with PostgreSQL ... 21
    7.1.4. Troubleshooting Non-Default Databases with Ambari ... 22
  7.2. Using Non-Default Databases - Hive ... 23
    7.2.1. Using Hive with Oracle ... 23
    7.2.2. Using Hive with MySQL ... 25
    7.2.3. Using Hive with PostgreSQL ... 27
    7.2.4. Troubleshooting Non-Default Databases with Hive ... 28
  7.3. Using Non-Default Databases - Oozie ... 29
    7.3.1. Using Oozie with Oracle ... 30
    7.3.2. Using Oozie with MySQL ... 30
    7.3.3. Using Oozie with PostgreSQL ... 32
    7.3.4. Troubleshooting Non-Default Databases with Oozie ... 33
8. Setting up Ambari to use an Internet Proxy Server ... 34
9. Configuring Network Port Numbers ... 36
  9.1. Default Network Port Numbers - Ambari ... 36
  9.2. Optional: Changing the Default Ambari Server Port ... 36
10. Change the JDK Version ... 38
11. Using Ambari Blueprints ... 40
12. Configuring HDP Stack Repositories for Red Hat Satellite ... 41
13. Tuning Ambari Performance ... 43
14. Tuning Ambari Metrics ... 44
  14.1. Collector Modes ... 45
  14.2. Aggregated Metrics TTL Settings ... 46
  14.3. Memory Settings ... 47
  14.4. (Optional) Enabling HBase Region and Table Metrics ... 47
  14.5. General Guidelines ... 48

15. Moving the Ambari Metrics Collector 50
16. Configuring HTTPS for the Oozie Server 51

1. Installing Ambari Agents Manually

In cases where you do not have SSH for Ambari to automatically install the Agents or you want to pre-install the Agents, you can perform a manual agent setup. This involves two steps:

1. Download the Ambari Repo [1]
2. Install the Ambari Agents Manually [5]

1.1. Download the Ambari Repo

Select the OS family running on your installation host.

RHEL/CentOS/Oracle Linux 7

On a server host that has Internet access, use a command line editor to perform the following steps:

1. Log in to your host as `root`.
2. Download the Ambari repository file to a directory on your installation host.

```
wget -nv http://public-repo-1.hortonworks.com/ambari/
centos7/2.x/updates/2.2.1.1/ambari.repo -O /etc/yum.repos.d/
ambari.repo
```

Important

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.2.2.1.1-2.x	Ambari 2.x	8
base	CentOS-7 - Base	6,518
extras	CentOS-7 - Extras	37
updates	CentOS-7 - Updates	785

4. Proceed to Install the Ambari Agents manually.

Note

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/centos7/RPM-
GPG-KEY/RPM-GPG-KEY-Jenkins
```

RHEL/CentOS/Oracle Linux 6

On a server host that has Internet access, use a command line editor to perform the following steps:

1. Log in to your host as `root`.
2. Download the Ambari repository file to a directory on your installation host.

   ```
   wget -nv http://public-repo-1.hortonworks.com/ambari/
   centos6/2.x/updates/2.2.1.1/ambari.repo -O /etc/yum.repos.d/
   ambari.repo
   ```

   Important

   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.2.2.1.1-2.x	Ambari 2.x	8
base	CentOS-6 - Base	6,518
extras	CentOS-6 - Extras	37
updates	CentOS-6 - Updates	785

4. Proceed to Install the Ambari Agents manually.

Note

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

On a server host that has Internet access, use a command line editor to perform the following steps:

1. Log in to your host as `root`.
2. Download the Ambari repository file to a directory on your installation host.

   ```
   wget -nv http://public-repo-1.hortonworks.com/ambari/suse11/2.x/
   updates/2.2.1.1/ambari.repo -O /etc/zypp/repos.d/ambari.repo
   ```

   Important

   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
AMBARI.2.2.1.1-2.x	Ambari 2.x	Yes
http-demeter.uni-regensburg.de-c997c8f9	SUSE-Linux-Enterprise-Software-Development-Kit-11-SP1 11.1.1-1.57	Yes
opensuse	OpenSuse	Yes

4. Proceed to Install the Ambari Agents manually.

Ubuntu 12

On a server host that has Internet access, use a command line editor to perform the following steps:

1. Log in to your host as `root`.
2. Download the Ambari repository file to a directory on your installation host.

   ```
   wget -nv http://public-repo-1.hortonworks.com/ambari/
   ubuntu12/2.x/updates/2.2.1.1/ambari.list -O /etc/apt/
   sources.list.d/ambari.list

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

   apt-get update
   ```

   Important

   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 showpkg ambari-server
   ```

   ```
   apt-cache showpkg ambari-agent
   ```

   ```
   apt-cache showpkg ambari-metrics-assembly
   ```

   You should see the Ambari packages in the list.

4. Proceed to Install the Ambari Agents manually.

Ubuntu 14

On a server host that has Internet access, use a command line editor to perform the following steps:

1. Log in to your host as `root`.

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

   ```
   wget -nv http://public-repo-1.hortonworks.com/ambari/
   ubuntu14/2.x/updates/2.2.1.1/ambari.list -O /etc/apt/
   sources.list.d/ambari.list
   ```

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

   ```
   apt-get update
   ```

   Important

   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 showpkg ambari-server
   ```

   ```
   apt-cache showpkg ambari-agent
   ```

   ```
   apt-cache showpkg ambari-metrics-assembly
   ```

   You should see the Ambari packages in the list.

4. Proceed to Install the Ambari Agents manually.

Debian 7

On a server host that has Internet access, use a command line editor to perform the following steps:

1. Log in to your host as `root`.

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

   ```
   wget -nv http://public-repo-1.hortonworks.com/ambari/
   debian7/2.x/updates/2.2.1.1/ambari.list -O /etc/apt/
   sources.list.d/ambari.list
   ```

   ```
   apt-key adv --recv-keys --keyserver keyserver.debian.com
   B9733A7A07513CAD
   ```

   ```
   apt-get update
   ```

   Important

   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 showpkg ambari-server
   ```

   ```
   apt-cache showpkg ambari-agent
   ```

   ```
   apt-cache showpkg ambari-metrics-assembly
   ```

   You should see the Ambari packages in the list.

4. Proceed to Install the Ambari Agents manually.

1.2. Install the Ambari Agents Manually

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

RHEL/CentOS/Oracle Linux

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.

   ```
   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 on every host in your cluster.

   ```
   ambari-agent start
   ```

   The agent registers with the Server on start.

Debian/Ubuntu

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

   ```
   apt-get 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 on every host in your cluster.

   ```
   ambari-agent start
   ```

   The agent registers with the Server on start.

2. Customizing HDP Services

- Defining Service Users and Groups for a HDP 2.x Stack [7]
- Setting Properties That Depend on Service Usernames/Groups [8]

2.1. Defining Service Users and Groups for a HDP 2.x Stack

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.

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.

Service Users

Service*	Component	Default User Account
Accumulo	Accumulo Tracer, Accumulo Monitor, Accumulo GC, Accumulo Master	accumulo (HDP 2.2 or later)
Ambari Metrics	Metrics Collector, Metrics Monitor	ams
Atlas	Atlas Metadata Server	atlas (HDP 2.3 or later)
Falcon	Falcon Server	falcon
Flume	Flume Agents	flume
HBase	MasterServer RegionServer	hbase
HDFS	NameNode SecondaryNameNode DataNode	hdfs
Hive	Hive Metastore, HiveServer2	hive
Kafka	Kafka Broker	kafka
Knox	Knox Gateway	knox

Service*	Component	Default User Account
Mahout	Mahout clients	mahout (HDP 2.2 or later)
MapReduce2	HistoryServer	mapred
Oozie	Oozie Server	oozie
PostgreSQL	PostgreSQL (with Ambari Server)	postgres (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.)
Ranger	Ranger Admin, Ranger Usersync	ranger (HDP 2.2 or later)
Ranger KMS	Ranger KMS Server	kms (HDP 2.3 or later)
Spark	Spark History Server	spark (HDP 2.2 or later)
Sqoop	Sqoop	sqoop
Storm	Masters (Nimbus, DRPC Server, Storm REST API, Server, Storm UI Server) Slaves (Supervisors, Logviewers)	storm
Tez	Tez clients	tez
WebHCat	WebHCat Server	hcat
YARN	NodeManager ResourceManager	yarn
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.

Service Groups

Service	Components	Default Group Account
All	All	hadoop
Atlas	Atlas Metadata Server	atlas
Knox	Knox Gateway	knox
Ranger	Ranger Admin, Ranger Usersync	ranger
Ranger KMS	Ranger KMS Server	kms
Spark	Spark History Server	spark

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`:

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.

Property Name	Value
dfs.block.local-path-access.user	The HBase username. The default is "hbase".

MapReduce Settings: Advanced

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

3. Configuring Storm for Supervision

If you have installed a cluster with HDP 2.2 Stack that includes the Storm service, you can configure the Storm components to operate under supervision. This section describes those steps:

1. Stop all Storm components.

   Using Ambari Web, browse to `Services > Storm > Service Actions`, choose Stop. Wait until the Storm service 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/common-services/STORM/0.9.1.2.1/metainfo.xml

   sed -ir "s/scripts\/nimbus.py/scripts\/nimbus_prod.py/g" /var/lib/ambari-
   server/resources/common-services/STORM/0.9.1.2.1/metainfo.xml
   ```

4. Install supervisord on all Nimbus and Supervisor hosts.

   - Install EPEL repository.

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

   - Install supervisor package for supervisord.

     ```
     yum install supervisor -y
     ```

   - Enable supervisord on autostart.

     ```
     chkconfig supervisord on
     ```

   - Change supervisord configuration file permissions.

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

5. Configure `supervisord` to supervise Nimbus Server and Supervisors by appending 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_67/bin/ JAVA_HOME=
   /usr/jdk64/jdk1.7.0_67 /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_67/bin/ JAVA_HOME=
/usr/jdk64/jdk1.7.0_67 /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
```

Note

Change `/usr/jdk64/jdk1.7.0_67` accordingly to the location of the JDK being used by Ambari in your environment.

6. Start Supervisord service on all Supervisor and Nimbus hosts.

   ```
   service supervisord start
   ```

7. Start Ambari Server.

   ```
   ambari-server start
   ```

8. Start all the other Storm components.

   Using Ambari Web, browse to `Services > Storm > Service Actions`, choose `Start`.

4. 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.

How to Customize the name of a host [12]

4.1. How to Customize the name of a host

1. At the `Install Options` step in the Cluster Installer wizard, select `Perform Manual Registration for Ambari Agents`.
2. Install the Ambari Agents manually on each host, as described in Install the 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 host name 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`

   where <hostname> is the host name to use for Agent registration.
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 on every host for these changes to take effect.

    ```
    ambari-agent restart
    ```

5. Moving the Ambari Server

To transfer an Ambari Server that uses the default, embedded, PostgreSQL database from one host to a new host, use the following instructions:

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

Note

If your Ambari Server is using one of the non-default databases (such as MySQL, Oracle, or an existing PostgreSQL instance) then be sure to follow backup, restore, and stop/start procedures that match that database type.

5.1. Back up Current Data

1. On the Ambari Server host, 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.db.username} -f ambari.sql
   ```

   ```
   Password: {ambari.db.password}
   ```

   where the following:

Variable	Description	Default
ambari.db.username	The database username.	ambari
ambari.db.password	The database password.	bigdata

4. Create a backup of the Ambari Server meta info.

   ```
   ambari-server backup
   ```

5.2. Update all Agents

1. On each agent host, stop the agent.

```
ambari-agent stop
```

2. Remove old agent certificates (if any exist).

   ```
   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.ambari.server.fqdn}

   url_port=8440

   secured_url_port=8441
   ```

5.3. Install the New Ambari Server

1. Install the new Ambari Server on the new host.

   ```
   yum install ambari-server
   ```

2. Run setup the Ambari Server and setup similar to how the original Ambari Server is configured.

   ```
   ambari-server setup
   ```

3. Restart the PostgreSQL instance.

   ```
   service postgresql restart
   ```

4. Open the PostgreSQL interactive terminal.

   ```
   su - postgres

   psql
   ```

5. Using the interactive terminal, drop the "ambari" database created by the new ambari setup and install.

   ```
   drop database ambari;
   ```

6. Check to make sure the databases have been dropped. The "ambari" databases should not be listed.

   ```
   \l
   ```

7. Create new "ambari" database to hold the transferred data.

   ```
   create database ambari;
   ```

8. Exit the PostgreSQL interactive terminal.

```
\q
```

9. Copy the saved data (/tmp/dbdumps/ambari.sql) from Back up Current Data to the new Ambari Server host.

10. Load the saved data into the new database.

    ```
    psql -d ambari -f /tmp/dbdumps/ambari.sql
    ```

11. Start the new Server.

    ```
    ambari-server start
    ```

12. On each Agent host, start the Ambari Agent.

    ```
    ambari-agent start
    ```

13. Open Ambari Web. Point your browser to:

    <new.Ambari.Server>`:8080`

The new Ambari Server is ready to use.

6. 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.

6.1. 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`
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.

Note

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.

6.2. 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 [17]

- Write Custom Java to Create LZO Files [17]

6.2.1. 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.Lz
   hive.exec.compress.output=true;SET
   mapreduce.output.fileoutputformat.compress=true;"
   ```

6.2.2. 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

   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>"
   ```

7. Using Non-Default Databases

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

- Using Non-Default Databases - Ambari [18]
- Using Non-Default Databases - Hive [23]
- Using Non-Default Databases - Oozie [29]

Important

Using the **Microsoft SQL Server** or **SQL Anywhere** database options are not supported.

7.1. 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 [18]
- Using Ambari with MySQL [19]
- Using Ambari with PostgreSQL [21]
- Troubleshooting Non-Default Databases with Ambari [22]

Important

Using the **Microsoft SQL Server** or **SQL Anywhere** database options are not supported.

Important

For High Availability (HA) purposes, it is **required** that the relational database used with Ambari is also made highly available following best practices for the given database type.

7.1.1. 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. For **Oracle Database 11g**: select `Oracle Database 11g Release 2 drivers > ojdbc6.jar`.

   c. For **Oracle Database 12c**: select `Oracle Database 12c Release 1 drivers > ojdbc7.jar`.

   d. Copy the .jar file to the Java share directory. For example:

      ```
      cp ojdbc7.jar /usr/share/java/
      ```

   e. Make sure the .jar file has the appropriate permissions. For example:

      ```
      chmod 644 /usr/share/java/ojdbc7.jar
      ```

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.

   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 respond to the prompts using the username/password credentials you created in step 2.

7.1.2. 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
   ```

   Ubuntu

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

   Debian

   ```
   apt-get 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.

   - 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.

7.1.3. 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.

7.1.4. Troubleshooting Non-Default Databases with Ambari

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

7.1.4.1. 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.

7.1.4.1.1. 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.

7.1.4.2. 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.

7.1.4.2.1. 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
```

7.1.4.3. 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.

7.1.4.3.1. Solution

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

7.1.4.4. 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.

7.1.4.4.1. 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.

7.2. 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 [23]
- Using Hive with MySQL [25]
- Using Hive with PostgreSQL [27]
- Troubleshooting Non-Default Databases with Hive [28]

Important

Using the **Microsoft SQL Server** or **SQL Anywhere** database options are not supported.

7.2.1. 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. For **Oracle Database 11g**: select `Oracle Database 11g Release 2 drivers > ojdbc6.jar`.

   c. For **Oracle Database 12c**: select `Oracle Database 12c Release 1 drivers > ojdbc7.jar`.

   d. Make sure the .jar file has the appropriate permissions. For example:

      ```
      chmod 644 ojdbc7.jar
      ```

   e. Execute the following command, adding the path to the downloaded .jar file:

      ```
      ambari-server setup --jdbc-db=oracle --jdbc-driver=/path/to/
      downloaded/ojdbc7.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 **HDP 2.2 or later Stacks**

     Important

     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.

7.2.2. 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*
      ```

      Ubuntu

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

      Debian

      ```
      apt-get 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.

   - For a **HDP 2.2 or later Stacks**:

     Important

     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 **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.

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

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

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

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

   d. Execute the following command:

      ```
      ambari-server setup --jdbc-db=postgres --jdbc-driver=/usr/
      share/java/postgresql-jdbc.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 or later Stacks:**

Important

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 **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 **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.

7.2.4. Troubleshooting Non-Default Databases with Hive

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

7.2.4.1. 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.

7.2.4.1.1. Solution

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

7.2.4.2. 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.

7.2.4.2.1. 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.

7.2.4.3. 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.

7.2.4.3.1. 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].

  Hive Security Authorization Settings

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

  Metastore Server SecurityHive Authorization and the HCatalog document [Storage Based Authorization].

7.3. 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 [30]
- Using Oozie with MySQL [30]
- Using Oozie with PostgreSQL [32]

• Troubleshooting Non-Default Databases with Oozie [33]

Important

Using the **Microsoft SQL Server** or **SQL Anywhere** database options are not supported.

7.3.1. Using Oozie with Oracle

To set up Oracle for use with Oozie:

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

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

   b. For **Oracle Database 11g**: select `Oracle Database 11g Release 2 drivers > ojdbc6.jar`.

   c. For **Oracle Database 12c**: select `Oracle Database 12c Release 1 drivers > ojdbc7.jar`.

   d. Make sure the .jar file has the appropriate permissions. For example:

   ```
   chmod 644 ojdbc7.jar
   ```

   e. Execute the following command, adding the path to the downloaded .jar file:

   ```
   ambari-server setup --jdbc-db=oracle --jdbc-driver=/path/to/
   downloaded/ojdbc7.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>;

   GRANT CONNECT, RESOURCE TO <OOZIEUSER>;

   QUIT;
   ```

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

7.3.2. 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.

a. 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*
   ```

   DEBIAN

   ```
   apt-get 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 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.

7.3.3. 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
      ```

      UBUNTU

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

      DEBIAN

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

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

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

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

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

   d. Execute the following command:

      ```
      ambari-server setup --jdbc-db=postgres --jdbc-driver=/usr/
      share/java/postgresql-jdbc.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 Oozie database name.

7.3.4. Troubleshooting Non-Default Databases with Oozie

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

7.3.4.1. 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.

7.3.4.1.1. Solution

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

7.3.4.2. 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.

7.3.4.2.1. 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
```

8. Setting up Ambari to use an Internet Proxy Server

If you plan to use the **public repositories** (i.e. available on the Internet) for installing the cluster software, you need to make sure Ambari and the hosts in the cluster have Internet access to obtain the software from those repositories. Specifically:

- **Ambari Server:** uses Internet access to validate the repositories.
- **yum** (or equivalent package manager depending on your operating system): performs the software installation from the repositories.

Therefore, if your environment requires use of an Internet proxy server for access, you must configure Ambari Server component and "yum" on all the hosts to use the proxy server.

Note

Ambari can install software if you have no Internet access. If you have no Internet access (via a proxy server or otherwise), you can use local repositories for installing the cluster software. In that case, configuring Ambari to use a proxy server is not required. However, Ambari and the hosts in the cluster must have access to your local repositories. See Using a Local Repository for more information on setting up and using local repositories.

Configure Internet Proxy Settings for Ambari Server

1. On the Ambari Server host, stop Ambari Server:

   ```
   ambari-server stop
   ```

2. Add proxy settings to the following script: `/var/lib/ambari-server/ambari-env.sh`.

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

3. 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>
   ```

4. If your proxy server requires authentication, add the username and password, as follows:

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

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

Configure yum for Internet Proxy Settings for All Hosts

Setting up yum to use a proxy server depends a lot on your environment and operating system. The instructions below provide some guidance but we **strongly recommend** you consult with your System Administrators and Operating System documentation for assistance & specific instructions.

1. On each host in the cluster, specify the proxy settings in /etc/yum.conf by adding the following entry:

   ```
   proxy=http://<yourProxyHost>:<yourProxyPort>
   ```

2. If your proxy server requires authentication, add the username and password, as follows:

   ```
   enableProxyAuth=1

   proxy_username=<username>

   proxy_password=<password>
   ```

3. Save the yum configuration file.

It is important to highlight that defining a proxy server, username and password in /etc/yum.conf means **all users of yum connect to the proxy server with those details**. Please consult your System Administrators and refer to your Operating System documentation for more details on this configuration and possible alternatives.

Operating System	Reference
CentOS / Red Hat	https://www.centos.org/docs/5/html/yum/sn-yum-proxy-server.html
Oracle Linux	https://docs.oracle.com/cd/E37670_01/E37355/html/ol_proxy_config.html
Ubuntu / Debian	https://help.ubuntu.com/community/AptGet/Howto

9. Configuring Network Port Numbers

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

- Default Network Port Numbers - Ambari [36]
- Optional: Changing the Default Ambari Server Port [36]

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

9.1. 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 Configure Ambari Server for Authenticatd HTTP for instructions.	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.	tcp	Ping port used for alerts to check the health of the Ambari Agent	No	

9.2. 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>
   ```

10. Change the JDK Version

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.

The choice of JDK is dependent on which HDP Stack you plan to install in your cluster. The following table indicates which JDKs work with which Stacks.

Stack	JDKs
HDP 2.3 or 2.4	JDK 1.7 or JDK 1.8
HDP 2.0, 2.1 or 2.2	JDK 1.7

Important

If you plan to upgrade between Stacks (for example, go from HDP 2.2 -> HDP 2.3 or 2.4) **do not change the JDK** until you have successfully upgraded the Stack and are running the cluster on the target Stack. For example, the high-level process should follow:

1. Running HDP 2.2 with JDK 1.7.
2. Perform Stack upgrade to HDP 2.3 or 2.4.
3. Change JDK from 1.7 to 1.8 (using the following procedure).

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 v1.8.

   ```
   [1] - Oracle JDK 1.8

   [2] - Oracle JDK 1.7

   [3] - Custom JDK
   ```

4. If you choose Oracle JDK 1.8 or Oracle JDK 1.7, the JDK you choose downloads and installs automatically on the Ambari Server host. This option requires that you have an internet connection. You must install this JDK on all hosts in the cluster to this same path.
5. If you choose `Custom JDK`, verify or add the custom JDK path on all hosts in the cluster. Use this option if you want to use OpenJDK or do not have an internet connection (and have pre-installed the JDK on all hosts).

6. After setup completes, you must restart each component for the new JDK to be used by the Hadoop services.

7. Using the Ambari Web UI, do the following tasks:

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

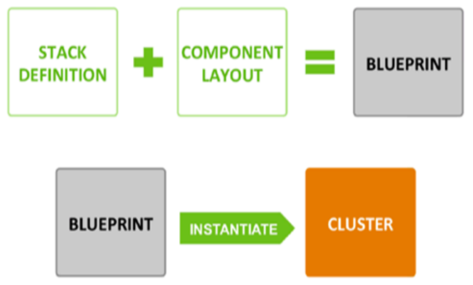
For more information about managing services in your cluster, see Managing Services.

Important

You **must** also update your JCE security policy files on the Ambari Server and all hosts in the cluster **to match the new JDK version**. If you are running Kerberos and do not update the JCE to match the JDK, you will have issues starting services. Refer to the Ambari Security Guide for more information on Installing the JCE.

11. 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.



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.

12. 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 created and distributed by Ambari in the .repo files:

1. **Before installing your cluster**, on the Ambari Server host, browse to the following:

   ```
   /var/lib/ambari-server/resources/stacks/HDP/2.0.6/configuration
   ```

2. Modify the cluster-env.xml file

   ```
   vi cluster-env.xml
   ```

3. Search for the property name **repo_suse_rhel_template** (for RHEL/ CentOS /SLES), **repo_ubuntu_template** (for Ubuntu), or **repo_debian_template** (for Debian). For example:

   ```
   <name>repo_suse_rhel_template</name>
   ```

4. Modify the <value> of the property to set the repository as disabled.

   ```
   <value>[{{repo_id}}-DISABLED]
   name={{repo_id}}-DISABLED
   {% if mirror_list %}mirrorlist={{mirror_list}}{% else %}baseurl={{base_url}}
   {% endif %}
   path=/
   enabled=0
   gpgcheck=0</value>
   ```

5. Save and exit.

6. Restart the Ambari Server and proceed with your cluster install.

Important

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

Important

When using RedHat Satellite or Spacewalk for repository management, it's very important to note that the name of the repository must match our HDP naming convention. When adding repositories, please ensure their names match those found in the .repo files. Example: HDP-2.2.4.2

Note

If you need to modify the template after cluster install, you can modify the property on cluster-env configuration using the Ambari REST API or the `/var/lib/ambari-server/resources/scripts/configs.sh` script. The example below shows setting the `repo_suse_rhel_template` property to "enabled=0" using the script.

```
configs.sh -u admin -p admin set ambari.server cluster.name
 cluster-env repo_suse_rhel_template "[{{repo_id}}-DISABLED]\nname=
{{repo_id}}-DISABLED\n{% if mirror_list %}mirrorlist={{mirror_list}}
{% else %}baseurl={{base_url}}{% endif %}\n\npath=/\nenabled=0\
ngpgcheck=0"
```

where **ambari.server** is the Ambari Server hostname and **cluster.name** is the name of your cluster.

13. Tuning Ambari Performance

For clusters larger than 200 nodes, consider the following tuning options:

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. Add the following properties to adjust the JDBC connection pool settings:

   ```
   server.jdbc.connection-pool.acquisition-size=5
   ```

   ```
   server.jdbc.connection-pool.max-age=0
   ```

   ```
   server.jdbc.connection-pool.max-idle-time=14400
   ```

   ```
   server.jdbc.connection-pool.max-idle-time-excess=0
   ```

   ```
   server.jdbc.connection-pool.idle-test-interval=7200
   ```

4. If using MySQL as the Ambari database, in your MSQL configuration, increase the wait_timeout and interacitve_timeout to 8 hours (28800) and max. connections from 32 to 128.

   Important

   It is **critical** that the Ambari configuration for `server.jdbc.connection-pool.max-idle-time` and `server.jdbc.connection-pool.idle-test-interval` must be lower than the MySQL wait_timeout and interactive_timeout set on the MySQL side. If you choose to decrease these timeout values, adjust down`server.jdbc.connection-pool.max-idle-time` and `server.jdbc.connection-pool.idle-test-interval` accordingly in the Ambari configuration so that they are less than wait_timeout and interactive_timeout.

5. Restart Ambari Server.

   ```
   ambari-server restart
   ```

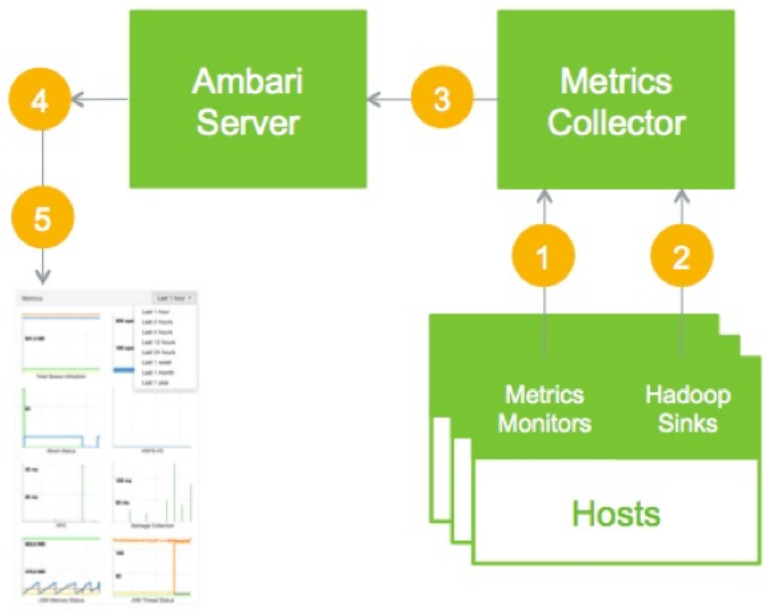
6. If you are using the Ambari Metrics service, you might want to consider switching from the default embedded mode to distributed mode, as well as other tuning options. See Configuring and Tuning Ambari Metrics for more information.

14. Tuning Ambari Metrics

Ambari Metrics System ("AMS") is a system for collecting, aggregating and serving Hadoop and system metrics in Ambari-managed clusters. AMS has three primary components: **Metrics Collector**, **Metrics Monitors** and **Hadoop Sinks**.

- The **Metrics Monitors** are installed and run on each host in the cluster to collect system-level metrics and publish to the **Metrics Collector**.
- The **Hadoop Sinks** plug into the various Hadoop components to publish Hadoop metrics to the **Metrics Collector**.
- The **Metrics Collector** is a daemon that runs on a specific host in the cluster and receives data from the registered publishers, the **Monitors** and **Sinks**.

The following diagram provides a high-level illustration of how the components of AMS work together to collect metrics and make those metrics available to Ambari.



1. **Metrics Monitors (on each host), send system-level metrics to Collector**
2. **Hadoop Sinks (on each host), send system-level metrics to Collector**
3. **Metrics Collector stores and aggregates metrics**
4. **Ambari exposes REST API for metrics retrieval**
5. **Ambari REST API feed Ambari Web UI**

To get optimal performance from the Ambari Metrics System, you should review the following Collector configuration options and the General Guidelines.

Option	Description
Collector Modes	The Collector can run in two modes: **embedded** mode and **distributed** mode. These modes impact where metrics data is stored and how the Collector process runs. See Collector Modes for more information.
Aggregated TTL Settings	The Time to Live settings for aggregated metrics. This impacts the amount of data that is stored and how long the data is retained. See Aggregated Metrics TTL for more information.
Memory Settings	Memory properties for the Collector components. These settings impact the overall performance of the Collector. See Memory Settings for more information.

14.1. Collector Modes

The **Metrics Collector** is built using Hadoop technologies such as HBase, Phoenix, and ATS. The Collector can store metrics data on the local filesystem, referred to as "**embedded** mode" or use an external HDFS, referred to as "**distributed** mode". By default, the Collector runs in **embedded mode**. In **embedded mode**, the Collector will capture and write metrics to the local file system on the host where the Collector is running. As well, all the Collector runs in a single process on that host.

Important

When running in **embedded** mode, you should confirm the "hbase.rootdir" and "hbase.tmp.dir" directory configurations in **Ambari Metrics > Configs > Advanced > ams-hbase-site** are using a sufficiently sized and not heavily utilized partition, such as:

```
file:///grid/0/var/lib/ambari-metrics-collector/hbase.
```

Refer to General Guidelines for more information on Disk Space recommendations.

Another critical factor in embedded mode is the TTL settings, which manage how much data will be stored. Refer to Aggregated Metrics TTL for more information on these settings.

When the Collector is configured for **distributed** mode, the Collector writes metrics to HDFS and the components will run in distributed processes. This mode helps manage CPU and memory consumption.

To switch the **Metrics Collector** from **embedded** mode to **distributed** mode, in **Ambari Web**, browse to **Services > Ambari Metrics > Configs**, make the following changes, then restart the Metrics Collector.

Configuration Section	Property	Description	Value
General	Metrics Service operation mode (timeline.metrics.service.operation.mode)	Designates whether to run in distributed or embedded mode.	distributed
Advanced ams-hbase-site	hbase.cluster.distributed	Indicates AMS will run in distributed mode.	true
Advanced ams-hbase-site	hbase.rootdir *see note 1*	The HDFS directory location where metrics will be stored.	hdfs://$NAMENODE_FQDN:8020/apps/ams/metrics

Note 1: If your cluster if configured for a highly-available NameNode, set the `hbase.rootdir` value to use the HDFS nameservice, instead of the NameNode hostname:

```
hdfs://hdfsnameservice/apps/ams/metrics
```

Optionally, existing data can be migrated from the local store to HDFS prior to switching to distributed mode.

1. Create HDFS directory for ams user. For example:

   ```
   su - hdfs -c 'hdfs dfs -mkdir -p /apps/ams/metrics'
   ```

2. Stop Metrics Collector.

3. Copy the metric data from the AMS local directory to an HDFS directory. This is the value of hbase.rootdir in Advanced ams-hbase-site used when running in embedded mode. For example:

   ```
   su - hdfs -c 'hdfs dfs -copyFromLocal /var/lib/ambari-metrics-
   collector/hbase/* /apps/ams/metrics'
   ```

   ```
   su - hdfs -c 'hdfs dfs -chown -R ams:hadoop /apps/ams/metrics'
   ```

4. Perform the configuration changes above to switch to distributed mode.

5. Start the Metrics Collector.

14.2. Aggregated Metrics TTL Settings

AMS provides configurable Time To Live configuration for aggregated metrics. The TTL settings are available in Ambari Metrics > Configs > Advanced ams-site and have the ".ttl" suffix. Each property name is self explanatory and controls the amount of time to keep metrics at the specified aggregation level before they are purged. The values for these TTL's are set in seconds. In an example where you are running a single-node sandbox and want to ensure that no values are stored for more than 7 days to save on local disk space, you would set any property ending in ".ttl" that has a value greater than 604800, 7 days in seconds, to 604800. That would ensure that properties such as timeline.metrics.cluster.aggregator.daily.ttl that controls the daily aggregation TTL, which by by default stores data for 2 years, will only store daily aggregations for 604800 seconds, or 7 days. Reducing the TTL values helps significantly reduce the total amount of storage used for metric storage. Those that matter most for reducing the total amount of disk space used for AMS are:

- timeline.metrics.cluster.aggregator.minute.ttl - Controls minute level aggregated metrics TTL
- timeline.metrics.host.aggregator.ttl - Controls host-based precision metrics TTL

It's important to note that these settings should be set during installation. If these settings need to be changed post-installation, they have to be set using the HBase shell. This is a current limitation imposed by Phoenix that is resolved in Ambari 2.1.2. To change these TTL settings in Ambari 2.1.1 and earlier, the HBase shell is used to connect to the embedded HBase instance that is part of AMS. Run the following command from the Collector host.

```
/usr/lib/ams-hbase/bin/hbase --config /etc/ams-hbase/conf shell
```

Once connected to HBase each of the following tables needs to be updated with the appropriate TTL values that are being changed. The table below maps a specific ".ttl" property in **Ambari Metrics > Configs > Advanced ams-site** to the actual HBase table.

Property	Table
timeline.metrics.cluster.aggregator.daily.ttl	METRIC_AGGREGATE_DAILY
timeline.metrics.cluster.aggregator.hourly.ttl	METRIC_AGGREGATE_HOURLY
timeline.metrics.cluster.aggregator.minute.ttl	METRIC_AGGREGATE
timeline.metrics.host.aggregator.daily.ttl	METRIC_RECORD_DAILY
timeline.metrics.host.aggregator.hourly.ttl	METRIC_RECORD_HOURLY
timeline.metrics.host.aggregator.minute.ttl	METRIC_RECORD_MINUTE
timeline.metrics.host.aggregator.ttl	METRIC_RECORD

For each table that needs to be updated, alter the TTL value as follows:

```
hbase(main):000:0> alter 'METRIC_RECORD_DAILY', { NAME => '0', TTL
=> 604800}
```

14.3. Memory Settings

Since AMS uses multiple components (such as HBase and Phoenix) for metrics storage and query, there are multiple tunable properties for tuning memory use. The following table lists each memory configuration.

Configuration	Property	Description
Advanced ams-env	metrics_collector_heapsize	Heap size configuration for the Collector.
Advanced ams-hbase-env	hbase_regionserver_heapsize	Heap size configuration for the single AMS HBase Region Se
Advanced ams-hbase-env	hbase_master_heapsize	Heap size configuration for the single AMS HBase Master.
Advanced ams-hbase-env	regionserver_xmn_size	Maximum value for the young generation heap size for the s AMS HBase RegionServer.
Advanced ams-hbase-env	hbase_master_xmn_size	Maximum value for the young generation heap size for the s AMS HBase Master.

14.4. (Optional) Enabling HBase Region and Table Metrics

Ambari disables HBase metrics (per region and per table) by default. HBase metrics can be numerous and can cause performance issues. HBase RegionServer metrics are available by default.

If you want HBase (per region and per table) metrics to be collected by Ambari, you can do the following. It is **highly recommended** that you test turning on this option and confirm that your AMS performance is acceptable.

1. On the Ambari Server, browse to:

   ```
   /var/lib/ambari-server/resources/common-services/
   HBASE/0.96.0.2.0/package/templates
   ```

2. Edit the following template files:

   `hadoop-metrics2-hbase.properties-GANGLIA-MASTER.j2`

   `hadoop-metrics2-hbase.properties-GANGLIA-RS.j2`

3. Comment out (or remove) the following lines:

   ```
   *.source.filter.class=org.apache.hadoop.metrics2.filter.GlobFilter
   ```

   ```
   hbase.*.source.filter.exclude=*Regions*
   ```

4. Save the template files and restart Ambari Server for the changes to take effect.

Important

If you upgrade Ambari to a newer version, you will need to re-apply this change to the template file.

14.5. General Guidelines

The operation mode, TTL, memory settings, and disk space requirements for AMS are dependent on the number of nodes in the cluster. The following table lists specific recommendations and tuning guidelines for each.

In **Ambari Web**, browse to **Ambari Metrics > Configs**, make the following changes, then restart the Collector.

Cluster Environment	Host Count	Disk Space	Collector Mode	TTL	Memory Settings
Single-Node Sandbox	1	2GB	embedded	Reduce TTLs to 7 Days	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=512 hbase_master_heapsize=512 hbase_master_xmn_size=128
PoC	1-5	5GB	embedded	Reduce TTLs to 30 Days	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=512 hbase_master_heapsize=512 hbase_master_xmn_size=128
Pre-Production	5-20	20GB	embedded	Reduce TTLs to 3 Months	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=1024 hbase_master_heapsize=512 hbase_master_xmn_size=128
Production	20-50	50GB	embedded	n.a.	metrics_collector_heap_size=1024 hbase_regionserver_heapsize=1024 hbase_master_heapsize=512 hbase_master_xmn_size=128
Production	50-200	100GB	embedded	n.a.	metrics_collector_heap_size=2048

Cluster Environment	Host Count	Disk Space	Collector Mode	TTL	Memory Settings
					hbase_regionserver_heapsize=2048 hbase_master_heapsize=2048 hbase_master_xmn_size=256
Production	200-400	200GB	embedded	n.a.	metrics_collector_heap_size=2048 hbase_regionserver_heapsize=2048 hbase_master_heapsize=2048 hbase_master_xmn_size=512
Production	400-800	200GB	distributed	n.a.	metrics_collector_heap_size=8192 hbase_regionserver_heapsize=122288 hbase_master_heapsize=1024 hbase_master_xmn_size=1024 regionserver_xmn_size=1024
Production	800+	500GB	distributed	n.a.	metrics_collector_heap_size=12288 hbase_regionserver_heapsize=16384 hbase_master_heapsize=16384 hbase_master_xmn_size=2048 regionserver_xmn_size=1024

15. Moving the Ambari Metrics Collector

Use this procedure to move the Ambari Metrics Collector to a new host. For information and guidelines on tuning the Ambari Metrics Service, refer to Tuning Ambari Metrics in the Ambari Reference Guide.

1. In **Ambari Web** , stop the **Ambari Metrics** service.

2. Execute the following API call to delete the **current** Metric Collector component.

   ```
   curl -u admin:admin -H "X-Requested-By:ambari" - i -X
   DELETE http://ambari.server:8080/api/v1/clusters/cluster.name/
   hosts/metrics.collector.hostname/host_components/METRICS_COLLECTOR
   ```

   where **ambari.server** is the Ambari Server host, **cluster.name** is your Cluster Name, and **metrics.collector.hostname** is the host running the Metrics Collector.

3. Execute the following API call to add Metrics Collector **to a new host**.

   ```
   curl -u admin:admin -H "X-Requested-By:ambari" - i -X
   POST http://ambari.server:8080/api/v1/clusters/cluster.name/
   hosts/metrics.collector.hostname/host_components/METRICS_COLLECTOR
   ```

   where **ambari.server** is the Ambari Server host, **cluster.name** is your Cluster Name, and **metrics.collector.hostname** is the host that will run the Metrics Collector.

4. In Ambari Web, go the Host page where you installed the new Metrics Collector. Click to Install the Metrics Collector component from the Host page.

5. In **Ambari Web**, start the **Ambari Metrics** service.

6. For every service, use **Ambari Web** > **Service Actions** > **Restart All** to start sending metrics to the new collector.

16. Configuring HTTPS for the Oozie Server

You might want to configure the Oozie server to connect via HTTPS, so it authenticates using SSL. This configuration is needed if you want to use the Oozie Web UI from a browser or use a digital certificate for security. You can configure HTTPS for the Oozie server using the Ambari UI.

1. Navigate to the Oozie Configs tab, click the Service Actions button, and click Stop to stop Oozie.

2. Expand the `Advanced oozie-env` section and add the following lines to the `oozie-env template` before the comment `# The port Oozie server runs`:

   ```
   export OOZIE_HTTPS_PORT=11443
   export OOZIE_HTTPS_KEYSTORE_FILE=/home/oozie/.keystore
   export OOZIE_HTTPS_KEYSTORE_PASS=password
   ```

3. Change `OOZIE_HTTP_PORT={{oozie_server_port}}` to `OOZIE_HTTP_PORT=11000`.

4. Expand the `Advanced oozie-site` section and set the `oozie.base.url` to `https://<BaseURL>:11443/oozie`.