

# Ambari 1.7.0 Security Guide



© Copyright © 2012, 2014 Hortonworks, Inc. Some rights reserved. Hortonworks, Inc.

Hortonworks Data Platform  
Ambari 1.7.0  
2014-12-02

## Copyright

This work by [Hortonworks, Inc.](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

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

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

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

## Table of Contents

Ambari Security Guide.....	4
Preparing Kerberos for Hadoop .....	4
Kerberos Overview .....	4
Installing and Configuring the KDC .....	5
Creating the Kerberos Database .....	6
Starting the KDC.....	6
Installing and Configuring the Kerberos Clients .....	6
Creating Service Principals and Keytab Files for HDP 2.x .....	7
Creating Service Principals and Keytab Files for HDP 1.x .....	12
Setting Up Hadoop Users .....	17
Overview .....	17
Creating Mappings Between Principals and UNIX User names.....	17
Setting up Ambari for Kerberos.....	19
Setting up JAAS for Ambari .....	20
Deploying JCE Policy Archives on the Ambari Server.....	20
Advanced Security Options for Ambari .....	21
Setting Up Ambari for LDAP or Active Directory Authentication .....	21
Setting Up LDAP User Authentication .....	21
Optional: Encrypt Database and LDAP Passwords.....	25
Reset Encryption .....	25
Optional: Set Up Security for Ambari .....	26
Set Up HTTPS for Ambari Server .....	27
Set Up HTTPS for Ganglia.....	28
Set Up HTTPS for Nagios.....	29
Optional: Set Up Two-Way SSL Between Ambari Server and Ambari Agents.....	30
Optional: Configure Ciphers and Protocols for Ambari Server .....	31

## Ambari Security Guide

This section describes how to set up strong authentication for Hadoop users and hosts in an Ambari-installed HDP cluster, and provides information on advanced security options for Ambari.

- Preparing Kerberos for Hadoop
- Setting Up Hadoop Users
- Setting up Ambari for Kerberos

### Preparing Kerberos for Hadoop

This section describes how to set up the Kerberos components in your Hadoop cluster.

- Kerberos Overview
- Installing and Configuring the KDC
- Creating the Kerberos Database
- Starting the KDC
- Installing and Configuring the Kerberos Clients
- Creating Service Principals and Keytab Files for Hadoop 2.x
- Creating Service Principals and Keytab Files for HDP 1.x

### Kerberos Overview

Establishing identity with strong authentication is the basis for secure access in Hadoop. Users need to be able to reliably “identify” themselves and then have that identity propagated throughout the Hadoop cluster. Once this is done those users can access resources (such as files or directories) or interact with the cluster (like running MapReduce jobs). As well, Hadoop cluster resources themselves (such as Hosts and Services) need to authenticate with each other to avoid potential malicious systems “posing as” part of the cluster to gain access to data.

To create that secure communication among its various components, Hadoop uses Kerberos. Kerberos is a third party authentication mechanism, in which users and services that users want to access rely on a third party - the Kerberos server - to authenticate each to the other. The Kerberos server itself is known as the Key Distribution Center, or KDC. At a high level, it has three parts:

- A database of the users and services (known as *principals*) that it knows about and their respective Kerberos passwords
- An *authentication server (AS)* which performs the initial authentication and issues a *Ticket Granting Ticket (TGT)*
- A *Ticket Granting Server (TGS)* that issues subsequent service tickets based on the initial TGT

A user principal requests authentication from the AS. The AS returns a TGT that is encrypted using the user principal's Kerberos password, which is known only to the user principal and the AS. The user principal decrypts the TGT locally using its Kerberos password, and from that point forward, until the ticket expires, the user principal can use the TGT to get service tickets from the TGS. Service tickets are what allow a principal to access various services.

Because cluster resources (hosts or services) cannot provide a password each time to decrypt the TGT, they use a special file, called a *keytab*, which contains the resource principal's authentication credentials.

The set of hosts, users, and services over which the Kerberos server has control is called a *realm*.

Table 1. Kerberos terminology

Term	Description
Key Distribution Center, or KDC	The trusted source for authentication in a Kerberos-enabled environment.
Kerberos KDC Server	The machine, or server, that serves as the Key Distribution Center.
Kerberos Client	Any machine in the cluster that authenticates against the KDC.
Principal	The unique name of a user or service that authenticates against the KDC.
Keytab	A file that includes one or more principals and their keys.
Realm	The Kerberos network that includes a KDC and a number of Clients.

## Installing and Configuring the KDC

To use Kerberos with Hadoop you can either use an existing KDC or install a new one just for Hadoop's use. The following gives a very high level description of the installation process. To get more information see [RHEL documentation](#), [CentOS documentation](#), or [SLES documentation](#).



Because Kerberos is a time-sensitive protocol, all hosts in the realm must be time-synchronized, for example, by using the Network Time Protocol (NTP). If the local system time of a client differs from that of the KDC by as little as 5 minutes (the default), the client will not be able to authenticate.

### 1 To install a new version of the server:

```
For RHEL/CentOS/Oracle Linux
yum install krb5-server krb5-libs krb5-auth-dialog krb5-workstation
```

```
For SLES 11
zypper install krb5 krb5-server krb5-client
```

```
For UBUNTU 12
apt-get install krb5 krb5-server krb5-client
```



The host on which you install the KDC must itself be secure.

- 2 When the server is installed use a text editor to edit the configuration file, located by default here:

```
/etc/krb5.conf
```

Change the `[realms]` section of this file by replacing the default “kerberos.example.com” setting for the `kdc` and `admin_server` properties with the Fully Qualified Domain Name of the KDC server. In the following example, “kerberos.example.com” has been replaced with “my.kdc.server”.

```
[realms]
EXAMPLE.COM = {
    kdc = my.kdc.server
    admin_server = my.kdc.server
}
```

## Creating the Kerberos Database

Use the utility `kdb5_util` to create the Kerberos database.

For RHEL/CentOS/Oracle Linux 6

```
/usr/sbin/kdb5_util create -s
```

For SLES 11

```
kdb5_util create -s
```

For UBUNTU 12

```
kdb5_util create -s
```

## Starting the KDC

Start the KDC.

For RHEL/CentOS/Oracle Linux 5 (DEPRECATED)

```
/etc/rc.d/init.d/krb5kdc start
/etc/rc.d/init.d/kadmin start
```

For SLES 11

```
rckrb5kdc start
rckadmind start
```

For UBUNTU 12

```
rckrb5kdc start
rckadmind start
```

## Installing and Configuring the Kerberos Clients

- 1 To install the Kerberos clients, on every server in the cluster:

For RHEL/CentOS/Oracle Linux

```
yum install krb5-workstation
```

```
For SLES 11
zypper install krb5-client
```

```
For UBUNTU 12
apt-get install krb5-client
```

- 2 Copy the `krb5.conf` file you modified in Installing and Configuring the KDC to all the servers in the cluster.

## Creating Service Principals and Keytab Files for HDP 2.x

Each service and sub-service in Hadoop must have its own principal. A principal name in a given realm consists of a primary name and an instance name, which in this case is the FQDN of the host that runs that service. As services do not log in with a password to acquire their tickets, their principal's authentication credentials are stored in a keytab file, which is extracted from the Kerberos database and stored locally with the service principal on the service component host.

First you must create the principal, using mandatory naming conventions.

Then you must create the keytab file with that principal's information and copy the file to the keytab directory on the appropriate service host.



Principals can be created either on the KDC machine itself or through the network, using an “admin” principal. The following instructions assume you are using the KDC machine and using the `kadmin.local` command line administration utility. Using `kadmin.local` on the KDC machine allows you to create principals without needing to create a separate “admin” principal before you start.

- 1 Open the `kadmin.local` utility on the KDC machine

```
/usr/sbin/kadmin.local
```

- 2 Create the service principals:

```
$kadmin.local
addprinc -randkey $primary_name/$fully.qualified.domain.name
@EXAMPLE.COM
```

The `-randkey` option is used to generate the password.

Notice in the example that each service principal's primary name has appended to it the instance name, the FQDN of the host on which it runs. This provides a unique principal name for services that run on multiple hosts, like DataNodes and NodeManagers. The addition of the host name serves to distinguish, for example, a request from DataNode A from a request from DataNode B. This is important for the following reasons:

- If the Kerberos credentials for one DataNode are compromised, it does not automatically lead to all DataNodes being compromised

- If multiple DataNodes have exactly the same principal and are simultaneously connecting to the NameNode, and if the Kerberos authenticator being sent happens to have same timestamps, then the authentication would be rejected as a replay request.

The principal name must match the values in the table below:

Table 2. Service Principals

Service	Component	Mandatory Principal Name
HDFS	NameNode	nn/\$FQDN
HDFS	NameNode HTTP	HTTP/\$FQDN
HDFS	SecondaryNameNode	nn/\$FQDN
HDFS	SecondaryNameNode HTTP	HTTP/\$FQDN
HDFS	DataNode	dn/\$FQDN
MR2	History Server	jhs/\$FQDN
MR2	History Server HTTP	HTTP/\$FQDN
YARN	ResourceManager	rm/\$FQDN
YARN	NodeManager	nm/\$FQDN
Oozie	Oozie Server	oozie/\$FQDN
Oozie	Oozie HTTP	HTTP/\$FQDN
Hive	Hive Metastore HiveServer2	hive/\$FQDN
Hive	WebHCat	HTTP/\$FQDN
HBase	MasterServer	hbase/\$FQDN
HBase	RegionServer	hbase/\$FQDN
ZooKeeper	ZooKeeper	zookeeper/\$FQDN
Nagios Server	Nagios	nagios/\$FQDN
JournalNode	JournalNode	jn/\$FQDN
ServerOnly required if you are setting up NameNode HA.		

For example, to create the principal for a DataNode service, issue the following command:

```
$kadmin.local
addprinc -randkey dn/ $DataNode-Host @EXAMPLE.COM
```

- 3 In addition you must create four special principals for Ambari's own use.



The names in the following table can be customized in the Customize Services step of the Ambari Install Wizard. If this is the case in your installation, the principal names should match the customized names. For example, if the HDFS Service User has been set to `hdfs1`, the respective principal for the Ambari HDFS User should also be created as `hdfs1`.

These principals do not have the FQDN appended to the primary name:

Table 3. Ambari Principals

User	Mandatory Principal Name
Ambari User This principal is used with the JAAS configuration. See Setting up JAAS for Ambari for more information.	ambari
Ambari Smoke Test User	ambari-qa
Ambari HDFS User	hdfs
Ambari HBase User	hbase

- After the principals are created in the database, you can extract the related keytab files for transfer to the appropriate host:

```
$kadmin.local
xst -norandkey -k $keytab_file_name $primary_name
/fully.qualified.domain.name@EXAMPLE.COM
```

You must use the mandatory names for the `<$keytab_file_name>` variable shown in the preceding table.



Some older versions of Kerberos do not support the `xst -norandkey` option. You can use the command without the `-norandkey` flag, except in cases where you need to copy a principal from one keytab file to another keytab file on a host. This might be a requirement if the Hadoop configurations on a host have keytab path properties that point to different keytab locations but have corresponding principal name properties that have the same values.

In situations like this, you can use the two step `kadmin / kutil` procedure. This description assumes MIT Kerberos. If you are using another version, please check the documentation for that version.

- Extract the keytab file information:

```
$kadmin
xst -k $keytab_file_name-temp1$primary_name
/fully.qualified.domain.name@EXAMPLE.COM
xst -k $keytab_file_name-temp2$primary_name
/fully.qualified.domain.name@EXAMPLE.COM
```

- Write the keytab to a file.

```
$kutil
kutil: rkt $keytab_file_name-temp1
kutil: rkt $keytab_file_name-temp2
kutil: wkt $keytab_file_name
kutil: clear
```

Table 4. Service Keytab File Names

Component	Principal Name	Mandatory Keytab File Name
NameNode	nn/\$FQDN	nn.service.keytab

NameNode HTTP	HTTP/\$FQDN	spnego.service.keytab
SecondaryNameNode	nn/\$FQDN	nn.service.keytab
SecondaryNameNode HTTP	HTTP/\$FQDN	spnego.service.keytab
DataNode	dn/\$FQDN	dn.service.keytab
MR2 History Server	jhs/\$FQDN	jhs.service.keytab
MR2 History Server HTTP	HTTP/\$FQDN	spnego.service.keytab
YARN	rm/\$FQDN	rm.service.keytab
YARN	nm/\$FQDN	nm.service.keytab
Oozie Server	oozie/\$FQDN	oozie.service.keytab
Oozie HTTP	HTTP/\$FQDN	spnego.service.keytab
Hive Metastore HiveServer2	hive/\$FQDN	hive.service.keytab
WebHCat	HTTP/\$FQDN	spnego.service.keytab
HBase Master Server	hbase/\$FQDN	hbase.service.keytab
HBase RegionServer	hbase/\$FQDN	hbase.service.keytab
ZooKeeper	zookeeper/\$FQDN	zk.service.keytab
Nagios Server	nagios/\$FQDN	nagios.service.keytab
Journal Server	jn/\$FQDN	jn.service.keytab
Only required if you are setting up NameNode HA.		
Ambari User	ambari	ambari.keytab
This principal is used with the JAAS configuration. See <a href="#">Setting up JAAS for Ambari</a> for more information.		
Ambari Smoke Test User	ambari-qa	smokeuser.headless.keytab
Ambari HDFS User	hdfs	hdfs.headless.keytab
Ambari HBase User	hbase	hbase.headless.keytab

For example, to create the keytab files for NameNode HTTP, issue the following command:

```
$kadmin.local
xst -norandkey -k spnego.service.keytab HTTP/<namenode-host>
```



If you have a large cluster, you may want to create a script to automate creating your principals and keytabs. To help with that, you can download a CSV-formatted file of all the required principal names and keytab files from the Ambari Web GUI. Select **Admin > Security > Enable Security**, then run the **Enable Security Wizard**, using the default values. At the bottom of the third page, in **Create Principals and Keytabs**, click [Download CSV](#).

- 5 When the keytab files have been created, on each host create a directory for them and set appropriate permissions.

```
mkdir -p /etc/security/keytabs/
chown root:hadoop /etc/security/keytabs
chmod 750 /etc/security/keytabs
```

- 6 Copy the appropriate keytab file to each host. If a host runs more than one component (for example, both NodeManager and DataNode), copy keytabs for both components. The Ambari Smoke Test User, the Ambari HDFS User, and the Ambari HBase User keytabs should be copied to the all hosts on the cluster.

If you have customized service user names, replace the default values below with your appropriate service user, group, and keytab names.

## 7 Set appropriate permissions for the keytabs.

If you have customized service user names, replace the default values below with your appropriate service user, group, and keytab names.

### 1 Optionally, if you have Set up JAAS on the Ambari server host:

```
chown ambari:ambari /etc/security/keytabs/ambari.keytab
chmod 400 /etc/security/keytabs/ambari.keytab
```

### 2 On the HDFS NameNode and SecondaryNameNode hosts:

```
chown hdfs:hadoop /etc/security/keytabs/nn.service.keytab
chmod 400 /etc/security/keytabs/nn.service.keytab
chown root:hadoop /etc/security/keytabs/spnego.service.keytab
chmod 440 /etc/security/keytabs/spnego.service.keytab
```

### 3 On the HDFS NameNode host, for the Ambari Test Users:

```
chown ambari-qa:hadoop
/etc/security/keytabs/smokeuser.headless.keytab
chmod 440 /etc/security/keytabs/smokeuser.headless.keytab
chown hdfs:hadoop /etc/security/keytabs/hdfs.headless.keytab
chmod 440 /etc/security/keytabs/hdfs.headless.keytab
chown hbase:hadoop /etc/security/keytabs/hbase.headless.keytab
chmod 440 /etc/security/keytabs/hbase.headless.keytab
```

### 4 On each host that runs an HDFS DataNode:

```
chown hdfs:hadoop /etc/security/keytabs/dn.service.keytab
chmod 400 /etc/security/keytabs/dn.service.keytab
```

### 5 On the host that runs the MR2 History Server:

```
chown mapred:hadoop /etc/security/keytabs/jhs.service.keytab
chmod 400 /etc/security/keytabs/jhs.service.keytab
chown root:hadoop /etc/security/keytabs/spnego.service.keytab
chmod 440 /etc/security/keytabs/spnego.service.keytab
```

### 6 On the host that runs the YARN ResourceManager:

```
chown yarn:hadoop /etc/security/keytabs/rm.service.keytab
chmod 400 /etc/security/keytabs/rm.service.keytab
```

### 7 On each host that runs a YARN NodeManager:

```
chown yarn:hadoop /etc/security/keytabs/nm.service.keytab
chmod 400 /etc/security/keytabs/nm.service.keytab
```

### 8 On the host that runs the Oozie Server:

```
chown oozie:hadoop /etc/security/keytabs/oozie.service.keytab
chmod 400 /etc/security/keytabs/oozie.service.keytab
chown root:hadoop /etc/security/keytabs/spnego.service.keytab
chmod 440 /etc/security/keytabs/spnego.service.keytab
```

**9 On the host that runs the Hive Metastore, HiveServer2 and WebHCat:**

```
chown hive:hadoop /etc/security/keytabs/hive.service.keytab
chmod 400 /etc/security/keytabs/hive.service.keytab
chown root:hadoop /etc/security/keytabs/spnego.service.keytab
chmod 440 /etc/security/keytabs/spnego.service.keytab
```

**10 On hosts that run the HBase MasterServer, RegionServer and ZooKeeper:**

```
chown hbase:hadoop /etc/security/keytabs/hbase.service.keytab
chmod 400 /etc/security/keytabs/hbase.service.keytab
chown zookeeper:hadoop /etc/security/keytabs/zk.service.keytab
chmod 400 /etc/security/keytabs/zk.service.keytab
```

**11 On the host that runs the Nagios server:**

```
chown nagios:nagios /etc/security/keytabs/nagios.service.keytab
chmod 400 /etc/security/keytabs/nagios.service.keytab
```

**12 On each host that runs a JournalNode, if you are setting up NameNode HA:**

```
chown hdfs:hadoop /etc/security/keytabs/jn.service.keytab
chmod 400 /etc/security/keytabs/jn.service.keytab
```

- 8 Verify that the correct keytab files and principals are associated with the correct service using the `klist` command. For example, on the NameNode: `klist -k -t /etc/security/keytabs/nn.service.keytab` Do this on each respective service in your cluster.**

## Creating Service Principals and Keytab Files for HDP 1.x

Each service and sub-service in Hadoop must have its own principal. A principal name in a given realm consists of a primary name and an instance name, which in this case is the FQDN of the host that runs that service. As services do not log in with a password to acquire their tickets, their principal's authentication credentials are stored in a keytab file, which is extracted from the Kerberos database and stored locally with the service principal on the service component host.

First, you must create the principal, using mandatory naming conventions. Then you must create the keytab file with that principal's information and copy the file to the keytab directory on the appropriate service host.



Principals can be created either on the KDC machine itself or through the network, using an “admin” principal. The following instructions assume you are using the KDC machine and using the `kadmin.local` command line administration utility. Using `kadmin.local` on the KDC machine allows you to create principals without needing to create a separate “admin” principal before you start.

1 Open the `kadmin.local` utility on the KDC machine

```
/usr/sbin/kadmin.local
```

2 Create the service principals:

```
$kadmin.local
addprinc -randkey $primary_name / $fully.qualified.domain.name
@EXAMPLE.COM
```

The `-randkey` option is used to generate the password.

Notice in the example that each service principal's primary name has appended to it the instance name, the FQDN of the host on which it runs. This provides a unique principal name for services that run on multiple hosts, like DataNodes and TaskTrackers. The addition of the host name serves to distinguish, for example, a request from DataNode A from a request from DataNode B. This is important for the following reasons:

- If the Kerberos credentials for one DataNode are compromised, it does not automatically lead to all DataNodes being compromised
- If multiple DataNodes have exactly the same principal and are simultaneously connecting to the NameNode, and if the Kerberos authenticator being sent happens to have same time stamp, then the authentication would be rejected as a replay request.

The principal name must match the following values:

Table 5. Service Principals

Service	Component	Mandatory Principal Name
HDFS	NameNode	nn/\$FQDN
HDFS	NameNode HTTP	HTTP/\$FQDN
HDFS	SecondaryNameNode	nn/\$FQDN
HDFS	SecondaryNameNode HTTP	HTTP/\$FQDN
HDFS	DataNode	dn/\$FQDN
MapReduce	JobTracker	jt/\$FQDN
MapReduce	TaskTracker	tt/\$FQDN
Oozie	Oozie Server	oozie/\$FQDN
Oozie	Oozie HTTP	HTTP/\$FQDN
Hive	Hive Metastore HiveServer2	hive/\$FQDN
Hive	WebHCat	HTTP/\$FQDN
HBase	MasterServer	hbase/\$FQDN
HBase	RegionServer	hbase/\$FQDN
ZooKeeper	ZooKeeper	zookeeper/\$FQDN
Nagios Server	Nagios	nagios/\$FQDN

**For example :** To create the principal for a DataNode service, issue the following command:

```
$kadmin.local
addprinc -randkey dn/$DataNode-Host@EXAMPLE.COM
```

3 In addition you must create four special principals for Ambari's own use.



The names in the following table can be customized in the Customize Services step of the Ambari Install Wizard. If this is the case in your installation, the principal names should match the customized names. For example, if the HDFS Service User has been set to `hdfs1`, the respective principal for the Ambari HDFS User should also be created as `hdfs1`.

You do NOT need to append the FQDN to the primary name:

Table 6. Ambari Principals

User	Default Principal Name
Ambari User This principal is used with the JAAS configuration. See Setting Up JAAS for Ambari for more information.	ambari
Ambari Smoke Test User	ambari-qa
Ambari HDFS User	hdfs
Ambari HBase User	hbase

- 4 After the principals are created in the database, you can extract the related keytab files for transfer to the appropriate host:

```
$kadmin.local
xst -norandkey -k $keytab_file_name
$primary_name/fully.qualified.domain.name@EXAMPLE.COM
```



Some older versions of Kerberos do not support the `xst -norandkey` option. You can use the command without the `-norandkey` flag, except in cases where you need to merge two principals with the same name into a single keytab file for a single host. In this case, you can use the two step `kadmin / ktutil` procedure. This description assumes MIT Kerberos. If you are using another version, please check the documentation for that version.

- 1 Extract the keytab file information.

```
$kadmin
xst -k $keytab_file_name-temp1$primary_name
/fully.qualified.domain.name@EXAMPLE.COM
xst -k $keytab_file_name-temp2$primary_name
/fully.qualified.domain.name@EXAMPLE.COM
```

- 2 Merge the keytabs into a single file.

```
$kutil
rkt $keytab_file_name-temp1
rkt $keytab_file_name-temp2
wkt $keytab_file_name
clear
```

You must use the mandatory names for the `<keytab_file_name>` variable shown in the following table. Adjust the principal names if necessary.

Table 7. Service Keytab File Names

Component	Principal Name	Mandatory Keytab File Name
NameNode	nn/\$FQDN	nn.service.keytab
NameNode HTTP	HTTP/\$FQDN	spnego.service.keytab
SecondaryNameNode	nn/\$FQDN	nn.service.keytab
SecondaryNameNode HTTP	HTTP/\$FQDN	spnego.service.keytab
DataNode	dn/\$FQDN	dn.service.keytab
JobTracker	jt/\$FQDN	jt.service.keytab
TaskTracker	tt/\$FQDN	tt.service.keytab
Oozie Server	oozie/\$FQDN	oozie.service.keytab
Oozie HTTP	HTTP/\$FQDN	spnego.service.keytab
Hive Metastore HiveServer2	hive/\$FQDN	hive.service.keytab
WebHCat	HTTP/\$FQDN	spnego.service.keytab
HBase Master Server	hbase/\$FQDN	hbase.service.keytab
HBase RegionServer	hbase/\$FQDN	hbase.service.keytab
ZooKeeper	zookeeper/\$FQDN	zk.service.keytab
Nagios Server	nagios/\$FQDN	nagios.service.keytab
Ambari User This principal is used with the JAAS configuration. See Setting Up JAAS for Ambari for more information.	ambari	ambari.keytab
Ambari Smoke Test User	ambari-qa	smokeuser.headless.keytab
Ambari HDFS User	hdfs	hdfs.headless.keytab
Ambari HBase User	hbase	hbase.headless.keytab

**For example :** To create the keytab files for NameNode HTTP, issue this command:

```
xst -norandkey -k spnego.service.keytab HTTP/<namenode-host>
```



If you have a large cluster, you may want to create a script to automate creating your principals and keytabs. The Ambari Web GUI can help. For further information see [Creating Service Principals and Keytab Files for HDP 2.x](#) or [Creating Service Principals and Keytab Files for HDP 1.x](#).

- When the keytab files have been created, on each host create a directory for them and set appropriate permissions.

```
mkdir -p /etc/security/keytabs/  
chown root:hadoop /etc/security/keytabs  
chmod 750 /etc/security/keytabs
```

- Copy the appropriate keytab file to each host. If a host runs more than one component (for example, both TaskTracker and DataNode), copy keytabs for both components. The Ambari Smoke Test User, the Ambari HDFS User, and the Ambari HBase User keytabs should be copied to all hosts.
- Set appropriate permissions for the keytabs.

If you have customized service user names, replace the default values below with your appropriate service user, group, and keytab names.

- 1 **Optionally, if you have Set up JAAS for Ambari on the Ambari server host:**

```
chown ambari:ambari /etc/security/keytabs/ambari.keytab
chmod 400 /etc/security/keytabs/ambari.keytab
```

- 2 **On the HDFS NameNode and SecondaryNameNode hosts:**

```
chown hdfs:hadoop /etc/security/keytabs/nn.service.keytab
chmod 400 /etc/security/keytabs/nn.service.keytab
chown root:hadoop /etc/security/keytabs/spnego.service.keytab
chmod 440 /etc/security/keytabs/spnego.service.keytab
```

- 3 **On the HDFS NameNode host, for the Ambari Test Users:**

```
chown ambari-qa:hadoop
/etc/security/keytabs/smokeuser.headless.keytab
chmod 440 /etc/security/keytabs/smokeuser.headless.keytab
chown hdfs:hadoop /etc/security/keytabs/hdfs.headless.keytab
chmod 440 /etc/security/keytabs/hdfs.headless.keytab
chown hbase:hadoop /etc/security/keytabs/hbase.headless.keytab
chmod 440 /etc/security/keytabs/hbase.headless.keytab
```

- 4 **On each host that runs an HDFS DataNode:**

```
chown hdfs:hadoop /etc/security/keytabs/dn.service.keytab
chmod 400 /etc/security/keytabs/dn.service.keytab
```

- 5 **On the host that runs the MapReduce JobTracker:**

```
chown mapred:hadoop /etc/security/keytabs/jt.service.keytab
chmod 400 /etc/security/keytabs/jt.service.keytab
```

- 6 **On each host that runs a MapReduce TaskTracker:**

```
chown mapred:hadoop /etc/security/keytabs/tt.service.keytab
chmod 400 /etc/security/keytabs/tt.service.keytab
```

- 7 **On the host that runs the Oozie Server:**

```
chown oozie:hadoop /etc/security/keytabs/oozie.service.keytab
chmod 400 /etc/security/keytabs/oozie.service.keytab
chown root:hadoop /etc/security/keytabs/spnego.service.keytab
chmod 440 /etc/security/keytabs/spnego.service.keytab
```

- 8 **On the host that runs the Hive Metastore, HiveServer2 and WebHCat:**

```
chown hive:hadoop /etc/security/keytabs/hive.service.keytab
chmod 400 /etc/security/keytabs/hive.service.keytab
chown root:hadoop /etc/security/keytabs/spnego.service.keytab
chmod 440 /etc/security/keytabs/spnego.service.keytab
```

**9 On hosts that run the HBase MasterServer, RegionServer and ZooKeeper:**

```
chown hbase:hadoop /etc/security/keytabs/hbase.service.keytab
chmod 400 /etc/security/keytabs/hbase.service.keytab
chown zookeeper:hadoop /etc/security/keytabs/zk.service.keytab
chmod 400 /etc/security/keytabs/zk.service.keytab
```

**10 On the host that runs the Nagios server:**

```
chown nagios:nagios /etc/security/keytabs/nagios.service.keytab
chmod 400 /etc/security/keytabs/nagios.service.keytab
```

**8 Verify that the correct keytab files and principals are associated with the correct service using the `klist` command. For example, on the NameNode:**

```
klist -k -t /etc/security/keytabs/nn.service.keytab .
```

Do this on each respective service in your cluster.

## Setting Up Hadoop Users

This section provides information on setting up Hadoop users for Kerberos.

- Overview
- Creating Mappings Between Principals and UNIX Usernames

### Overview

Hadoop determines group file ownership and access control based on each user's group memberships. To configure Hadoop for use with Kerberos and Ambari you must create a mapping between service principals and these UNIX user names.

A user is mapped to the groups in which it belongs using an implementation of the `GroupMappingServiceProvider` interface. The implementation is pluggable and is configured in `core-site.xml`.

By default Hadoop uses `ShellBasedUnixGroupsMapping`, which is an implementation of `GroupMappingServiceProvider`. It fetches the group membership for a user name by executing a UNIX shell command. In secure clusters, since the user names are actually Kerberos principals, `ShellBasedUnixGroupsMapping` will work only if the Kerberos principals map to valid UNIX user names. Hadoop provides a feature that lets administrators specify mapping rules to map a Kerberos principal to a local UNIX user name.

### Creating Mappings Between Principals and UNIX User names

Hadoop uses a rule-based system to create mappings between service principals and their related UNIX user names. The rules are specified in the `core-site.xml` configuration file as the value to the optional key `hadoop.security.auth_to_local`.

The default rule is simply named `DEFAULT`. It translates all principals in your default domain to their first component. For example, `myusername@APACHE.ORG` and `myusername/admin@APACHE.ORG` both become `myusername`, assuming your default domain is `APACHE.ORG`.

Use the following instructions to configure the mappings between principals and UNIX user names:

- **Creating Rules**
- **Examples**

### Creating Rules

- **Simple Rules**

To make a simple map between principal names and UNIX users, you create a straightforward substitution rule. For example, to map the `ResourceManager(rm)` and `NodeManager(nm)` principals in the `EXAMPLE.COM` realm to the UNIX `$YARN_USER` user and the `NameNode(nn)` and `DataNode(dn)` principals to the UNIX `$HDFS_USER` user, you would make this the value for the `hadoop.security.auth_to_local` key in `core-site.xml`.

```
RULE:[2:$1@$0] ([jt]t@.*EXAMPLE.COM) s/./ / $YARN_USER /
RULE:[2:$1@$0] ([nd]n@.*EXAMPLE.COM) s/./ / $HDFS_USER /
DEFAULT
```

- **Complex Rules**

To accommodate more complex translations, you create a hierarchical set of rules to add to the default. Each rule is divided into three parts: base, filter, and substitution.

- **The Base:**

The base begins with the number of components in the principal name (excluding the realm), followed by a colon, and the pattern for building the user name from the sections of the principal name. In the pattern section `$0` translates to the realm, `$1` translates to the first component and `$2` to the second component.

For example:

```
[1:$1@$0] translates myusername@APACHE.ORG to myusername@APACHE.ORG
```

```
[2:$1] translates myusername/admin@APACHE.ORG to myusername
```

```
[2:$1%$2] translates myusername/admin@APACHE.ORG to myusername%admin
```

- **The Filter:**

The filter consists of a regex in a parentheses that must match the generated string for the rule to apply.

For example:

`(.*%admin)` matches any string that ends in `%admin`

`(.*@SOME.DOMAIN)` matches any string that ends in `@SOME.DOMAIN`

- **The Substitution:**

The substitution is a sed rule that translates a regex into a fixed string.

For example:

`s/@ACME\ .COM//` removes the first instance of `@SOME.DOMAIN`.

`s/@ [A-Z]*\ .COM//` removes the first instance of `@` followed by a name followed by `COM`.

`s/X/Y/g` replaces all of the `X` in the name with `Y`

### Examples

- If your default realm was `APACHE.ORG`, but you also wanted to take all principals from `ACME.COM` that had a single component `joe@ACME.COM`, you would create this rule:

```
RULE:[1:$1@$0] (. *@ACME\ .COM) s/@.*//
DEFAULT
```

- To also translate names with a second component, you would use these rules:

```
RULE:[1:$1@$0] (. *@ACME\ .COM) s/@.*//
RULE:[2:$1@$0] (. *@ACME\ .COM) s/@.*//
DEFAULT
```

- To treat all principals from `APACHE.ORG` with the extension `/admin` as `admin`, your rules would look like this:

```
RULE[2:$1%$2@$0] (. *%admin@APACHE\ .ORG) s/.*/admin/
DEFAULT
```

## Setting up Ambari for Kerberos

To turn on Kerberos-based security in the Ambari Web GUI you must:

- 1 Have already set up Kerberos for your cluster.  
For more information, see [Preparing Kerberos for Hadoop](#).
- 2 Go to the **Admin** tab.
- 3 Select **Security**.
- 4 Click **Enable Security** and follow the steps in the **Enable Security Wizard**.



## Setting up JAAS for Ambari

If you want to set up Java Authentication and Authorization Services (JAAS) configurations for Ambari to provide independent, secure access to native Hadoop GUIs such as the NameNode UI, use the Apache community documentation topic [Create the JAAS Configuration Files](#) to set up your configurations. Then, do the following:

- 1 Log into the Ambari server host.

Ambari Server should not be running when you do this. Edit configuration files before you start Ambari Server the first time or, stop the Ambari Server, edit the files, then re-start Ambari Server.

- 2 Run the following, specific setup-security command and answer the prompts:

```
ambari-server setup-security
```

- 1 Select 5 for **Setup Ambari kerberos JAAS configuration**.
- 2 Enter the Kerberos principal name for the Ambari server you set up earlier.
- 3 Enter the path to the keytab for the Ambari principal.
- 4 Restart Ambari Server:

```
ambari-server restart
```

## Deploying JCE Policy Archives on the Ambari Server

On a secure cluster having no internet access, you must deploy the Java Cryptography Extension (JCE) security policy.jar files on the Ambari Server, before setting up your Ambari server with a custom JDK.

When you enable security, Ambari distributes the JCE.jar files to all appropriate hosts in your cluster.

To obtain and deploy the JCE.jar files appropriate for the JDK version in your cluster on your Ambari-server host,

- 1 Download the archive from one of the following locations  
**For JDK 1.6 :** <http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html>  
**For JDK 1.7 :** <http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html>
- 2 Save the archive in a temporary location.
- 3 Copy the archive to `/var/lib/ambari-server/resources` on the Ambari server.

## Advanced Security Options for Ambari

This section describes several security options for an Ambari-monitored-and-managed Hadoop cluster.

- Setting Up LDAP or Active Directory Authentication
- Encrypt Database and LDAP Passwords
- Set Up Security for Ambari
- Set Up Two-Way SSL Between Ambari Server and Ambari Agents

### Setting Up Ambari for LDAP or Active Directory Authentication

By default Ambari uses an internal database as the user store for authentication and authorization. If you want to configure LDAP or Active Directory (AD) external authentication, you need to collect the following information and run a setup command.

Also, you must synchronize your LDAP users and groups into the Ambari DB to be able to manage authorization and permissions against those users and groups.

#### Setting Up LDAP User Authentication

The following table details the properties and values you need to know to set up LDAP authentication.



If you are going to set `bindAnonymously` to false (the default), you need to make sure you have an LDAP Manager name and password set up. If you are going to use SSL, you need to make sure you have already set up your certificate and keys.

Table 8. Ambari Server LDAP Properties

Property	Values	Description
<code>authentication.ldap.primaryUrl</code>	<code>server:port</code>	The hostname and port for the LDAP or AD server. Example: <code>my.ldap.server:389</code>
<code>authentication.ldap.secondaryUrl</code>	<code>server:port</code>	The hostname and port for the secondary LDAP or AD server. Example: <code>my.secondary.ldap.server:389</code> This is an optional value.
<code>authentication.ldap.useSSL</code>	true or false	If true, use SSL when connecting to the LDAP or AD server.
<code>authentication.ldap.usernameAttribute</code>	[LDAP attribute]	The attribute for username. Example: <code>uid</code>
<code>authentication.ldap.baseDn</code>	[Distinguished Name]	The root Distinguished Name to search in the directory for users. Example: <code>ou=people,dc=hadoop,dc=apache,dc=org</code>

authentication.ldap.bindAnonymously	true or false	If true, bind to the LDAP or AD server anonymously
authentication.ldap.managerDn	[Full Distinguished Name]	If Bind anonymous is set to false, the Distinguished Name ("DN") for the manager. Example: uid=hdfs,ou=people,dc=hadoop,dc=apache,dc=org
authentication.ldap.managerPassword	[password]	If Bind anonymous is set to false, the password for the manager
authentication.ldap.userObjectClass	[LDAP Object Class]	The object class that is used for users. Example: organizationalPerson
authentication.ldap.groupObjectClasses	[LDAP Object Class]	The object class that is used for groups. Example: groupOfUniqueNames
authentication.ldap.groupMembershipAttr	[LDAP attribute]	The attribute for group membership. Example: uniqueMember
authentication.ldap.groupNamingAttr	[LDAP attribute]	The attribute for group name.

### Configure Ambari to use LDAP Server

If the LDAPS server certificate is signed by a trusted Certificate Authority, there is no need to import the certificate into Ambari so this section does not apply to you. If the LDAPS server certificate is self-signed, or is signed by an unrecognized certificate authority such as an internal certificate authority, you must import the certificate and create a keystore file. The following example creates a keystore file at `/keys/ldaps-keystore.jks`, but you can create it anywhere in the file system:

Run the LDAP setup command on the Ambari server and answer the prompts, using the information you collected above:

- 1 `mkdir /keys`
- 2 `$JAVA_HOME/bin/keytool -import -trustcacerts -alias root -file $PATH_TO_YOUR_LDAPS_CERT -keystore /keys/ldaps-keystore.jks`
- 3 Set a password when prompted. You will use this during `ambari-server setup-ldap`.

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

- 1 At the **Primary URL\*** prompt, enter the server URL and port you collected above. Prompts marked with an asterisk are required values.
- 2 At the **Secondary URL** prompt, enter the secondary server URL and port. This value is optional.
- 3 At the **Use SSL\*** prompt, enter your selection. *If using LDAPS*, enter `true`.
- 4 At the **User name attribute\*** prompt, enter your selection. The default value is `uid`.
- 5 At the **Base DN\*** prompt, enter your selection.
- 6 At the **Bind anonymously\*** prompt, enter your selection.

- 7 At the **Manager DN\*** prompt, enter your selection if you have set `bind.Anonymously` to `false`.
- 8 At the **Enter the Manager Password\*** prompt, enter the password for your LDAP manager.
- 9 At the Enter the `userObjectClass*` prompt, enter the object class that is used for users.
- 10 At the Enter the `groupObjectClass*` prompt, enter the object class that is used for groups.
- 11 At the Enter the `groupMembershipAttr*` prompt, enter the attribute for group membership.
- 12 At the Enter the `groupNamingAttr*` prompt, enter the attribute for group name.
- 13 If you set **Use SSL\*** = true in step 3, the following prompt appears: **Do you want to provide custom TrustStore for Ambari?**

Consider the following options and respond as appropriate.

- **More secure option:** If using a self-signed certificate that you do not want imported to the existing JDK keystore, enter `y`.

For example, you want this certificate used only by Ambari, not by any other applications run by JDK on the same host.

If you choose this option, additional prompts appear. Respond to the additional prompts as follows:

- 1 At the **TrustStore type** prompt, enter `jks`.
  - 2 At the **Path to TrustStore file** prompt, enter `/keys/ldaps-keystore.jks` (or the actual path to your keystore file).
  - 3 At the **Password for TrustStore** prompt, enter the password that you defined for the keystore.
- **Less secure option:** If using a self-signed certificate that you want to import and store in the existing, default JDK keystore, enter `n`.

- 1 Convert the SSL certificate to X.509 format, if necessary, by executing the following command:

```
openssl x509 -in slapd.pem -out <slapd.crt>
```

Where `<slapd.crt>` is the path to the X.509 certificate.

- 2 Import the SSL certificate to the existing keystore, for example the default jre certificates storage, using the following instruction:

```
/usr/jdk64/jdk1.7.0_45/bin/keytool -import -trustcacerts -file slapd.crt -keystore /usr/jdk64/jdk1.7.0_45/jre/lib/security/cacerts
```

Where Ambari is set up to use JDK 1.7. Therefore, the certificate must be imported in the JDK 7 keystore.

- 14 Review your settings and if they are correct, select `y`.
- 15 Start or restart the Server

```
ambari-server restart
```

Initially the users you have enabled all have Ambari User privileges. Ambari Users can read metrics, view service status and configuration, and browse job information. For these new users to be able to start or stop services, modify configurations, and run smoke tests, they need to be Admins. To make this change, as an Ambari Admin, use **Manage Ambari > Users > Edit**. For instructions, see [Managing Users and Groups](#).

### *Synchronizing LDAP Users and Groups*

Run the LDAP synchronize command and answer the prompts to initiate the sync:

```
ambari-server sync-ldap [option]
```



To perform this operation, your Ambari Server must be running and you must be an Ambari Admin.

The utility provides three options for synchronization:

- Specific set of users and groups, or
- Synchronize the existing users and groups in Ambari with LDAP, or
- All users and groups

### *Specific Set of Users and Groups*

```
ambari-server sync-ldap --users users.txt --groups groups.txt
```

Use this option to synchronize a specific set of users and groups from LDAP into Ambari. Provide the command a text file of comma-separated users and groups, and those LDAP entities will be imported and synchronized with Ambari.



Group membership is determined using the Group Membership Attribute specified during setup-ldap.

### *Existing Users and Groups*

```
ambari-server sync-ldap --existing
```

After you have performed a synchronization of a specific set of users and groups, you use this option to synchronize only those entities that are in Ambari with LDAP. Users will be removed from Ambari if they no longer exist in LDAP, and group membership in Ambari will be updated to match LDAP.



Group membership is determined using the Group Membership Attribute specified during setup-ldap.

## All Users and Groups



Only use this option if you are sure you want to synchronize all users and groups from LDAP into Ambari. If you only want to synchronize a subset of users and groups, use a specific set of users and groups option.

```
ambari-server sync-ldap --all
```

This will import all entities with matching LDAP user and group object classes into Ambari.

## Optional: Encrypt Database and LDAP Passwords

By default the passwords to access the Ambari database and the LDAP server are stored in a plain text configuration file. To have those passwords encrypted, you need to run a special setup command.

Ambari Server should not be running when you do this: either make the edits before you start Ambari Server the first time or bring the server down to make the edits.

- 1 On the Ambari Server, run the special setup command and answer the prompts:

```
ambari-server setup-security
```

- 1 Select **4** for **Encrypt passwords stored in ambari.properties file.**
- 2 Provide a master key for encrypting the passwords. You are prompted to enter the key twice for accuracy.  
  
If your passwords are encrypted, you need access to the master key to start Ambari Server.
- 3 You have three options for maintaining the master key:
  - At the **Persist** prompt, select **y**. This stores the key in a file on the server.
  - Create an environment variable `AMBARI_SECURITY_MASTER_KEY` and set it to the key.
  - Provide the key manually at the prompt on server start up.
- 4 Start or restart the Server

```
ambari-server restart
```

## Reset Encryption

There may be situations in which you want to:

- Remove encryption entirely
- Change the current master key, either because the key has been forgotten or because you want to change the current key as a part of a security routine.

Ambari Server should not be running when you do this.

### Remove Encryption Entirely

To reset Ambari database and LDAP passwords to a completely unencrypted state:

- 1 On the Ambari host, open `/etc/ambari-server/conf/ambari.properties` with a text editor and set this property

```
security.passwords.encryption.enabled=false
```

- 2 Delete `/var/lib/ambari-server/keys/credentials.jceks`
- 3 Delete `/var/lib/ambari-server/keys/master`
- 4 You must now reset the database password and, if necessary, the LDAP password. Run `ambari-server setup` and `ambari-server setup-ldap` again.

### Change the Current Master Key

To change the master key:

- If you know the current master key or if the current master key has been persisted:

- 1 Re-run the encryption setup command and follow the prompts.

```
ambari-server setup-security
```

- 1 Select **4** for **Encrypt passwords stored in ambari.properties file**.
- 2 Enter the current master key when prompted if necessary (if it is not persisted or set as an environment variable).
- 3 At the **Do you want to reset Master Key** prompt, enter `yes`.
- 4 At the prompt, enter the new master key and confirm.

- If you do **not** know the current master key:

- 1 Remove encryption entirely, as described here.
- 2 Re-run `ambari-server setup-security` as described here.
- 3 Start or restart the Ambari Server.

```
ambari-server restart
```

## Optional: Set Up Security for Ambari

There are four ways you can increase the security settings for your Ambari server installation.



If you plan to configure your cluster for Kerberos, you may use the **Setup Ambari kerberos JAAS configuration** option, which is described in [Setting up Ambari for Kerberos](#).

- Set Up HTTPS for Ambari Server
- Set Up HTTPS for Ganglia
- Set Up HTTPS for Nagios
- Encrypt Database and LDAP Passwords

## Set Up HTTPS for Ambari Server

If you want to limit access to the Ambari Server to HTTPS connections, you need to provide a certificate. While it is possible to use a self-signed certificate for initial trials, they are not suitable for production environments. After your certificate is in place, you must run a special setup command.

Ambari Server should not be running when you do this. Either make these changes before you start Ambari the first time, or bring the server down before running the setup command.

- 1 Log into the Ambari Server host.
- 2 Locate your certificate. If you want to create a temporary self-signed certificate, use this as an example:

```
openssl genrsa -out $wserver.key 2048
openssl req -new -key $wserver.key -out $wserver.csr
openssl x509 -req -days 365 -in $wserver.csr -signkey $wserver.key -out
$wserver.crt
```

Where `$wserver` is the Ambari Server host name.

The certificate you use must be PEM-encoded, not DER-encoded. If you attempt to use a DER-encoded certificate, you see the following error:

```
unable to load certificate
140109766494024:error:0906D06C:PEM routines:PEM_read_bio:no start
line:pem_lib.c
:698:Expecting: TRUSTED CERTIFICATE
```

You can convert a DER-encoded certificate to a PEM-encoded certificate using the following command:

```
openssl x509 -in cert.crt -inform der -outform pem -out cert.pem
```

where `cert.crt` is the DER-encoded certificate and `cert.pem` is the resulting PEM-encoded certificate.

- 3 Run the special setup command and answer the prompts

```
ambari-server setup-security
```

- 1 Select `1` for **Enable HTTPS for Ambari server.**
- 2 Respond `y` to `Do you want to configure HTTPS ?`
- 3 Select the port you want to use for SSL. The default port number is 8443.
- 4 Provide the path to your certificate and your private key. For example, put your certificate and private key in `/etc/ambari-server/certs` with `root` as the owner or the non-root user you designated during Ambari Server setup for the `ambari-server` daemon.

- 5 Provide the password for the private key.
- 6 Start or restart the Server

```
ambari-server restart
```

## Set Up HTTPS for Ganglia

If you want Ganglia to use HTTPS instead of the default HTTP to communicate with Ambari Server, use the following instructions.

The servers should not be running when you do this: either make the edits before you start Ambari Server the first time or bring the servers down to make the edits.

- 1 Set up the Ganglia server.
  - 1 Log into the Ganglia server host.
  - 2 Create a self-signed certificate on the Ganglia server host. For example:

```
openssl genrsa -out $gserver.key 2048
openssl req -new -key $gserver.key -out $gserver.csr
openssl x509 -req -days 365 -in $gserver.csr -signkey
$gserver.key -out $gserver.crt
```

Where `$gserver` is the Ganglia server host name.

- 3 Install SSL on the Ganglia server host.

```
yum install mod_ssl
```

- 4 Edit the SSL configuration file on the Ganglia server host.

- 1 Using a text editor, open:

```
/etc/httpd/conf.d/ssl.conf
```

- 2 Add lines setting the certificate and key file names to the files you created earlier in this procedure. For example:

```
SSLCertificateFile $gserver.crt
SSLCertificateKeyFile $gserver.key
```

Where `$gserver` is the Ganglia server host name.

- 5 Disable HTTP access (optional).

- 1 Using a text editor, open:

```
/etc/httpd/conf/httpd.conf
```

- 2 Comment out the port 80 listener:

```
# Listen 80
```

- 6 Restart the `httpd` service on the Ganglia server host.

```
service httpd restart
```

- 2 Set up and restart the Ambari Server.

- 1 Log into the Ambari Server.
- 2 Run the special setup command and answer the prompts.

```
ambari-server setup-security
```

- 1 Select **2** for **Enable HTTPS for Ganglia service.**
  - 2 Respond **y** to **Do you want to configure HTTPS for Ganglia service.**
  - 3 Enter your TrustStore type. Your options are `jks`, `jceks`, or `pks12`.
  - 4 Enter the path to your TrustStore file.
  - 5 Enter the password for your TrustStore and then re-enter to confirm. The password must be at least 6 characters long.
  - 6 Enter the path to the Ganglia server certificate file.
- 3 Start or restart the Server

```
ambari-server restart
```

## Set Up HTTPS for Nagios

If you want Nagios to use HTTPS instead of HTTP (the default), use the following instructions.

The servers should not be running when you do this: either make the edits before you start Ambari Server the first time or bring the servers down to make the edits.

- 1 Set up the Nagios server.
  - 1 Log into the Nagios server host.
  - 2 Create a self-signed certificate on the Nagios server host. For example:

```
openssl genrsa -out $nserver.key 2048
openssl req -new -key $nserver.key -out $nserver.csr
openssl x509 -req -days 365 -in $nserver.csr -signkey
$nserver.key -out $nserver.crt
```

Where `$nserver` is the Nagios server host name.

- 3 Install SSL on the Nagios server host.

```
yum install mod_ssl
```

- 4 Edit the SSL configuration file on the Nagios server host.

- 1 Using a text editor, open:

```
/etc/httpd/conf.d/ssl.conf
```

- 2 Add lines setting the certificate and key file names to the files you created previously in this procedure. For example:

```
SSLCertificateFile $nserver.crt
SSLCertificateKeyFile $nserver.key
```

Where `$nserver` is the Nagios server host name.

- 5 Disable HTTP access (optional)

- 1 Using a text editor, open:

```
/etc/httpd/conf/httpd.conf
```

- 2 Comment out the port 80 listener:

```
# Listen 80
```

- 6 Restart the `httpd` service on the Nagios server host.

```
service httpd restart
```

- 2 Set up and restart the Ambari Server.

- 1 Log into the Ambari Server.
- 2 Run the special setup command and answer the prompts.

```
ambari-server setup-security
```

- 1 Select **2** for **Enable HTTPS for Nagios service**.
  - 2 Respond **y** to **Do you want to configure HTTPS for Nagios?**
  - 3 Enter your TrustStore type. Your options are `jks`, `jceks`, or `pks12`.
  - 4 Enter the path to your TrustStore file.
  - 5 Enter the password for your TrustStore and then re-enter to confirm. The password must be at least 6 characters long.
  - 6 Enter the path to the Nagios server certificate file.
- 3 Start or restart the Server

```
ambari-server restart
```

## Optional: Set Up Two-Way SSL Between Ambari Server and Ambari Agents

Two-way SSL provides a way to encrypt communication between Ambari Server and Ambari Agents. By default Ambari ships with Two-way SSL disabled. To enable Two-way SSL:

Ambari Server should not be running when you do this: either make the edits before you start Ambari Server the first time or bring the server down to make the edits.

- 1 On the Ambari Server host, open `/etc/ambari-server/conf/ambari.properties` with a text editor.
- 2 Add the following property:

```
security.server.two_way_ssl = true
```

- 3 Start or restart the Ambari Server.

```
ambari-server restart
```

The Agent certificates are downloaded automatically during Agent Registration.

## Optional: Configure Ciphers and Protocols for Ambari Server

Ambari provides control of ciphers and protocols that are exposed via Ambari Server.

- 1 To disable specific ciphers, you can optionally add a list of the following format to `ambari.properties`. If you specify multiple ciphers, separate each cipher using a vertical bar `|`.

```
security.server.disabled.ciphers=TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
```

- 2 To disable specific protocols, you can optionally add a list of the following format to `ambari.properties`. If you specify multiple protocols, separate each protocol using a vertical bar `|`.

```
security.server.disabled.protocols=SSL|SSLv2|SSLv3
```