Hortonworks Data Platform

Ambari Security Guide

(March 7, 2016)

docs.cloudera.com

Hortonworks Data Platform: Ambari Security 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. Ambari Security Guide

Ambari and Hadoop have many advanced security options. This guide provides information on configuring Ambari and Hadoop for strong authentication with Kerberos, as well as other security options.

2. Configuring Ambari and Hadoop for Kerberos

This chapter describes how to configure Kerberos for strong authentication for Hadoop users and hosts in an Ambari-managed cluster.

2.1. Kerberos Overview

Strongly authenticating and establishing a user's identity 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). Besides users, Hadoop cluster resources themselves (such as Hosts and Services) need to authenticate with each other to avoid potential malicious systems or daemon's "posing as" trusted components of the cluster to gain access to data.

Hadoop uses Kerberos as the basis for strong authentication and identity propagation for both user and services. Kerberos is a third party authentication mechanism, in which users and services 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.

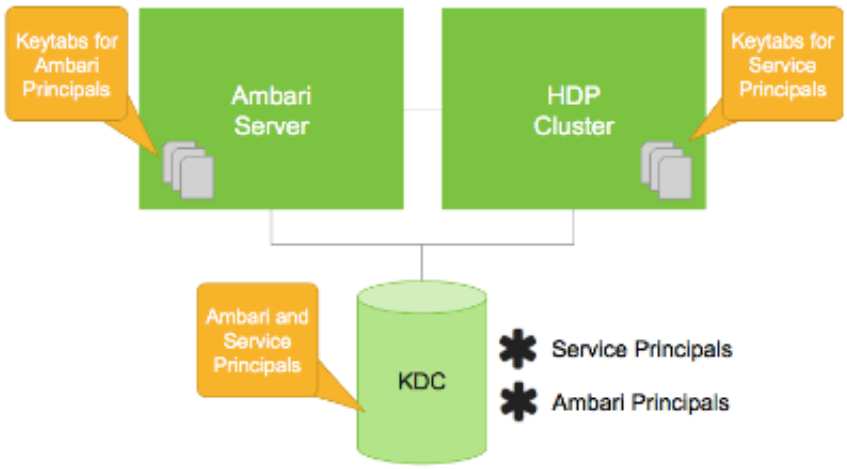
Because cluster resources (hosts or services) cannot provide a password each time to decrypt the TGT, they use a special file, called a **keytab**, which contains the resource principal's authentication credentials. The set of hosts, users, and services over which the Kerberos server has control is called a **realm**.

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 (KDC).
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.
KDC Admin Account	An administrative account used by Ambari to create principals and generate keytabs in the KDC.

2.2. Hadoop and Kerberos Principals

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, in this case the instance name 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 in a secured directory with the service principal on the service component host.



Principals and Keytabs

Principal and Keytab Naming Conventions

Asset	Convention	Example
Principals	$service_component_name/ $FQDN@EXAMPLE.COM	nn/ c6401.ambari.apache.org@EXAMPLE.COM
Keytabs	$service_component_abbreviation.service.keytab	/etc/security/keytabs/nn.service.keytab

Note

In addition to the Hadoop **Service Principals**, Ambari itself also requires a set of **Ambari Principals** to perform service "smoke" checks and alert health checks.

Keytab files for the Ambari, or headless, principals reside on each cluster host, just as keytab files for the service principals.

Notice in the preceding example the primary name for each service principal. These primary names, such as nn or hive for example, represent the NameNode or Hive service, respectively. Each primary name has appended to it the instance name, the FQDN of the host on which it runs. This convention provides a unique principal name for services that run on multiple hosts, like DataNodes and NodeManagers. Adding 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:

- Compromised Kerberos credentials for one DataNode do not automatically lead to compromised Kerberos credentials for all DataNodes.
- 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 is rejected as a replay request.

2.3. Installing and Configuring the KDC

Ambari is able to configure Kerberos in the cluster to work with an existing MIT KDC, or existing Active Directory installation. This section describes the steps necessary to prepare for this integration.

Note

If you do not have an existing KDC (MIT or Active Directory), Install a new MIT KDC . Please be aware that installing a KDC on a cluster host ***after*** installing the Kerberos client may overwrite the krb5.conf file generated by Ambari.

You can choose to have Ambari connect to the KDC and automatically create the necessary Service and Ambari principals, generate and distribute the keytabs (“Automated Kerberos Setup”). Ambari also provides an advanced option to manually configure Kerberos. If you choose this option, you must create the principals, generate and distribute the keytabs. Ambari will not do this automatically (“Manual Kerberos Setup”).

- Use an Existing MIT KDC [4]
- Use an Existing Active Directory [5]
- Use Manual Kerberos Setup [5]

For convenience, use the instructions to (Optional) Install a new MIT KDC if you do not have an existing KDC available.

2.3.1. Use an Existing MIT KDC

To use an existing MIT KDC for the cluster, you must prepare the following:

- Ambari Server and cluster hosts have network access to both the KDC and KDC admin hosts.
- KDC administrative credentials are on-hand.

Proceed with Enabling Kerberos Security in Ambari.

Note

You will be prompted to enter the KDC Admin Account credentials during the Kerberos setup so that Ambari can contact the KDC and perform the necessary principal and keytab generation. By default, Ambari will not retain the KDC credentials unless you have configured Ambari for encrypted passwords. Refer to Managing Admin Credentials for more information.

2.3.2. Use an Existing Active Directory

To use an existing Active Directory domain for the cluster with Automated Kerberos Setup, you must prepare the following:

- Ambari Server and cluster hosts have network access to, and be able to resolve the DNS names of, the Domain Controllers.
- Active Directory secure LDAP (LDAPS) connectivity has been configured.
- Active Directory User container for principals has been created and is on-hand. For example, "OU=Hadoop,OU=People,dc=apache,dc=org"
- Active Directory administrative credentials with delegated control of “Create, delete, and manage user accounts” on the previously mentioned User container are on-hand.

Proceed with Enabling Kerberos Security in Ambari.

Note

You will be prompted to enter the KDC Admin Account credentials during the Kerberos setup so that Ambari can contact the KDC and perform the necessary principal and keytab generation. By default, Ambari will not retain the KDC credentials unless you have configured Ambari for encrypted passwords. Refer to Managing Admin Credentials for more information.

2.3.3. Use Manual Kerberos Setup

To perform Manual Kerberos Setup, you must prepare the following:

- Cluster hosts have network access to the KDC.
- Kerberos client utilities (such as kinit) have been installed on every cluster host.
- The Java Cryptography Extensions (JCE) have been setup on the Ambari Server host and all hosts in the cluster.
- The Service and Ambari Principals will be manually created in the KDC before completing this wizard.
- The keytabs for the Service and Ambari Principals will be manually created and distributed to cluster hosts before completing this wizard.

Proceed with Enabling Kerberos Security in Ambari.

2.3.4. (Optional) Install a new MIT KDC

The following gives a very high level description of the KDC installation process. To get more information see specific Operating Systems documentation, such as RHEL documentation, CentOS documentation, or SLES documentation.

Note

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.

Install the KDC Server

1. Install a new version of the KDC server:

   RHEL/CentOS/Oracle Linux

   ```
   yum install krb5-server krb5-libs krb5-workstation
   ```

   SLES

   ```
   zypper install krb5 krb5-server krb5-client
   ```

   Ubuntu/Debian

   ```
   apt-get install krb5-kdc krb5-admin-server
   ```

2. Using a text editor, open the KDC server configuration file, located by default here:

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

3. 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 host. 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
   }
   ```

4. Some components such as HUE require renewable tickets. To configure MIT KDC to support them, ensure the following settings are specified in the `libdefaults` section of the `/etc/krb5.conf` file.

   ```
   renew_lifetime = 7d
   ```

Note

For Ubuntu/Debian, the setup of the default realm for the KDC and KDC Admin hostnames is performed during the KDC server install. You can re-run

setup using dpkg-reconfigure krb5-kdc. Therefore, Steps 2 and 3 above are not needed for Ubuntu/Debian.

Create the Kerberos Database

- Use the utility kdb5_util to create the Kerberos database.

  RHEL/CentOS/Oracle Linux

  ```
  kdb5_util create -s
  ```

  SLES

  ```
  kdb5_util create -s
  ```

  Ubuntu/Debian

  ```
  krb5_newrealm
  ```

Start the KDC

- Start the KDC server and the KDC admin server.

  RHEL/CentOS/Oracle Linux 6

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

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

  RHEL/CentOS/Oracle Linux 7

  ```
  systemctl start krb5kdc
  ```

  ```
  systemctl start kadmin
  ```

  SLES 11

  ```
  rckrb5kdc start
  ```

  ```
  rckadmind start
  ```

  Ubuntu/Debian

  ```
  service krb5-kdc restart
  ```

  ```
  service krb5-admin-server restart
  ```

Important

When installing and managing your own MIT KDC, it is ***very important*** to ***set up the KDC server to auto-start on boot***. For example:

RHEL/CentOS/Oracle Linux 6

```
chkconfig krb5kdc on
```

```
chkconfig kadmin on
```

RHEL/CentOS/Oracle Linux 7

```
systemctl enable krb5kdc
```

```
systemctl enable kadmin
```

SLES 11

```
chkconfig rckrb5kdc on
```

```
chkconfig rckadmind on
```

Create a Kerberos Admin

Kerberos 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.

Note

You will need to provide these admin account credentials to Ambari when enabling Kerberos. This allows Ambari to connect to the KDC, create the cluster principals and generate the keytabs.

1. Create a KDC admin by creating an admin principal.

   ```
   kadmin.local -q "addprinc admin/admin"
   ```

2. Confirm that this admin principal has permissions in the KDC ACL. Using a text editor, open the KDC ACL file:

   RHEL/CentOS/Oracle Linux

   ```
   vi /var/kerberos/krb5kdc/kadm5.acl
   ```

   SLES

   ```
   vi /var/lib/kerberos/krb5kdc/kadm5.acl
   ```

   Ubuntu/Debian

   ```
   vi /etc/krb5kdc/kadm5.acl
   ```

3. Ensure that the KDC ACL file includes an entry so to allow the admin principal to administer the KDC for your specific realm. When using a realm that is different than EXAMPLE.COM, **be sure there is an entry for the realm you are using**. If not present, principal creation will fail. For example, for an admin/admin@HADOOP.COM principal, you should have an entry:

   ```
   */admin@HADOOP.COM *
   ```

4. After editing and saving the kadm5.acl file, you must restart the kadmin process.

   RHEL/CentOS/Oracle Linux 6

   ```
   /etc/rc.d/init.d/kadmin restart
   ```

   RHEL/CentOS/Oracle Linux 7

   ```
   systemctl restart kadmin
   ```

   SLES 11

   ```
   rckadmind restart
   ```

   Ubuntu/Debian

   ```
   service krb5-admin-server restart
   ```

2.4. Enabling Kerberos Security

Whether you choose automated or manual Kerberos setup, Ambari provides a wizard to help with enabling Kerberos in the cluster. This section provides information on preparing Ambari before running the wizard, and the steps to run the wizard.

- Installing the JCE [9]
- Running the Kerberos Security Wizard [10]

Important

Prerequisites for enabling Kererbos are having the JCE installed on all hosts on the cluster (including the Ambari Server) and having the Ambari Server host as part of the cluster. This means the Ambari Server host should be running an Ambari Agent.

Note

Ambari Metrics will not be secured with Kerberos unless it is configured for distributed metrics storage. By default, it uses embedded metrics storage and will not be secured as part of the Kerberos Wizard. If you wish to have Ambari Metrics secured with Kerberos, please see this topic to enable distributed metrics storage prior to running the Kerberos Wizard.

2.4.1. Installing the JCE

Before enabling Kerberos in the cluster, you must deploy the Java Cryptography Extension (JCE) security policy files on the Ambari Server and on all hosts in the cluster.

Important

If you are using Oracle JDK, **you must** distribute and install the JCE **on all hosts** in the cluster, including the Ambari Server. **Be sure to restart Ambari Server**

after installng the JCE. If you are using OpenJDK, some distributions of the OpenJDK come with unlimited strength JCE automatically and therefore, installation of JCE is not required.

2.4.1.1. Install the JCE

1. On the Ambari Server, obtain the JCE policy file appropriate for the JDK version in your cluster.
   - For Oracle JDK 1.8:

     http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html
   - For Oracle JDK 1.7:

     http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html
2. Save the policy file archive in a temporary location.
3. On Ambari Server and on each host in the cluster, add the unlimited security policy JCE jars to `$JAVA_HOME/jre/lib/security/`.

   For example, run the following to extract the policy jars into the JDK installed on your host:

   ```
   unzip -o -j -q jce_policy-8.zip -d /usr/jdk64/jdk1.8.0_60/jre/lib/security/
   ```
4. Restart Ambari Server.
5. Proceed to Running the Security Wizard.

2.4.2. Running the Kerberos Security Wizard

Ambari provides three options for enabling Kerberos:

- Existing MIT KDC
- Existing Active Directory
- Manage Kerberos principals and keytabs manually

When choosing **Existing MIT KDC** or **Existing Active Directory**, the Kerberos Wizard prompts for information related to the KDC, the KDC Admin Account and the Service and Ambari principals. Once provided, Ambari will automatically create principals, generate keytabs and distribute keytabs to the hosts in the cluster. The services will be configured for Kerberos and the service components are restarted to authenticate against the KDC. This is the **Automated Setup** option. See Launching the Kerberos Wizard (Automated Setup) for more details.

When choosing **Manage Kerberos principals and keytabs manually**, you must create the principals, generate and distribute the keytabs. Ambari will not do this automatically. This

is the **Manual Setup** option. See Launching the Kerberos Wizard (Manual Setup) for more details.

2.4.2.1. Launching the Kerberos Wizard (Automated Setup)

1. Be sure you have Installed and Configured your KDC and have prepared the JCE on each host in the cluster.
2. Log in to Ambari Web and Browse to `Admin > Kerberos`.
3. Click "Enable Kerberos" to launch the wizard.
4. Select the type of KDC you are using and confirm you have met the prerequisites.
5. Provide information about the KDC and admin account.
   a. In the **KDC** section, enter the following information:
      - In the **KDC Host** field, the IP address or FQDN for the KDC host. Optionally a port number may be included.
      - In the **Realm name** field, the default realm to use when creating service principals.
      - (Optional) In the **Domains** field, provide a list of patterns to use to map hosts in the cluster to the appropriate realm. For example, if your hosts have a common domain in their FQDN such as host1.hortonworks.local and host2.hortonworks.local, you would set this to:

        ```
        .hortonworks.local,hortonworks.local
        ```

   b. In the **Kadmin** section, enter the following information:
      - In the **Kadmin Host** field, the IP address or FQDN for the KDC administrative host. Optionally a port number may be included.
      - The **Admin principal** and **password** that will be used to create principals and keytabs.
      - (Optional) If you have configured Ambari for encrypted passwords, the **Save Admin Credentials** option will be enabled. With this option, you can have Ambari store the KDC Admin credentials to use when making cluster changes. Refer to Managing Admin Credentials for more information on this option.
6. Modify any advanced Kerberos settings based on your environment.
   a. (Optional) To manage your Kerberos client krb5.conf manually (and not have Ambari manage the krb5.conf), expand the **Advanced krb5-conf** section and uncheck the "Manage" option. **You must have the krb5.conf configured on each host**.

Note

When manually managing the krb5.conf it is recommended to ensure that DNS is not used for looking up KDC, and REALM entries. Relying on

DNS can cause negative performance, and functional impact. To ensure that DNS is not used, ensure the following entries are set in the libdefaults section of your configuration.

```
[libdefaults]
dns_lookup_kdc = false
dns_lookup_realm = false
```

b. (Optional) to configure any additional KDC's to be used for this environment, add an entry for each additional KDC to the **realms** section of the **Advanced krb5-conf**'s krb5-conf template.

```
kdc = {{kdc_host}}
kdc = otherkdc.example.com
```

c. (Optional) To not have Ambari install the Kerberos client libraries on all hosts, expand the **Advanced kerberos-env** section and uncheck the "Install OS-specific Kerberos client package(s)" option. **You must have the Kerberos client utilities installed on each host**.

d. (Optional) If your Kerberos client libraries are in non-standard path locations, expand the **Advanced kerberos-env** section and adjust the "Executable Search Paths" option.

e. (Optional) If your KDC has a password policy, expand the **Advanced kerberos-env** section and adjust the Password options.

f. (Optional) Ambari will test your Kerberos settings by generating a test principal and authenticating with that principal. To customize the test principal name that Ambari will use, expand the **Advanced kerberos-env** section and adjust the **Test Kerberos Principal** value. By default, the test princial name is a combination of cluster name and date **(${cluster_name}-${short_date})**. This test principal **will be deleted** after the test is complete.

g. (Optional) If you need to customize the attributes for the principals Ambari will create, when using Active Directory, see the Customizing the Attribute Template for more information. When using MIT KDC, you can pass **Principal Attributes** options in the **Advanced kerberos-env** section. For example, you can set options related to pre-auth or max. renew life by passing:

```
-requires_preauth -maxrenewlife "7 days"
```

7. Proceed with the install.
8. Ambari will install Kerberos clients on the hosts and test access to the KDC by testing that Ambari can create a principal, generate a keytab and distribute that keytab.
9. Customize the Kerberos identities used by Hadoop and proceed to kerberize the cluster.

Important

On the **Configure Identities** step, be sure to review the principal names, particularly the **Ambari Principals** on the **General** tab. These principal names, by default, append the name of the cluster to each of the Ambari principals. You can leave this as default or adjust these by removing the "-${**cluster-name**}" from principal name string. For example, if your cluster is named

HDP and your realm is EXAMPLE.COM, the hdfs principal will be created as hdfs-HDP@EXAMPLE.COM.

10. Confirm your configuration. You can optionally download a CSV file of the principals and keytabs that Ambari will automatically create.

11. Click **Next** to start the process.

12. After principals have been created and keytabs have been generated and distributed, Ambari updates the cluster configurations, then starts and tests the Services in the cluster.

Note

If your cluster includes Storm, after enabling Kerberos, you must also Set Up Ambari for Kerberos for Storm Service Summary information to be displayed in Ambari Web. Otherwise, you will see n/a for Storm information such as Slots, Tasks, Executors and Topologies.

13. Exit the wizard when complete.

2.4.2.2. Launching the Kerberos Wizard (Manual Setup)

1. Be sure you have Installed and Configured your KDC and have prepared the JCE on each host in the cluster.

2. Log in to Ambari Web and Browse to `Admin > Kerberos`.

3. Click “Enable Kerberos” to launch the wizard.

4. Select the **Manage Kerberos principals and keytabs manually** option and confirm you have met the prerequisites.

5. Providing information about the KDC and your Kerberos environment.

   a. If your Kerberos client libraries are in non-standard path locations, expand the **Advanced kerberos-env** section and adjust the “Executable Search Paths” option.

6. Customize the Kerberos identities used by Hadoop and proceed to kerberize the cluster.

Important

On the **Configure Identities** step, be sure to review the principal names, particularly the **Ambari Principals** on the **General** tab. These principal names, by default, append the name of the cluster to each of the Ambari principals. You can leave this as default or adjust these by removing the "-${**cluster-name**}" from principal name string. For example, if your cluster is named HDP and your realm is EXAMPLE.COM, the hdfs principal will be created as hdfs-HDP@EXAMPLE.COM.

7. Confirm your configuration. Since you have chosen the Manual Kerberos Setup option, obtain the CSV file for the list of principals and keytabs required for the cluster to work

with Kerberos. **Do not proceed until you have manually created and distributed the principals and keytabs to the cluster hosts**.

8. Click **Next** to continue.

9. Ambari updates the cluster configurations, then starts and tests the Services in the cluster.

10. Exit the wizard when complete.

2.5. Kerberos Client Packages

If you chose to enable Kerberos using the Automated Kerberos Setup option, as part of the enabling Kerberos process, Ambari installs the Kerberos clients on the cluster hosts. Depending on your operating system, the following packages are installed:

Packages installed by Ambari for the Kerberos Client

Operating System	Packages
RHEL/CentOS/Oracle Linux 7	krb5-workstation
RHEL/CentOS/Oracle Linux 6	krb5-workstation
SLES 11	krb5-client
Ubuntu/Debian	krb5-user, krb5-config

2.6. Disabling Kerberos Security

After Enabling Kerberos Security, you can disable Kerberos.

1. Log in to Ambari Web and Browse to Admin > Kerberos.

2. Click **Disable Kerberos** to launch the wizard.

3. Complete the wizard.

Note

If you have enabled Kerberos with an Automated Setup option, Ambari will attempt to contact the KDC and remove the principals created by Ambari. If the KDC is unavailable, the wizard will fail on the Unkerberize step. You can choose to ignore and continue the failure but removal of principals from the KDC will not be performed.

2.7. Customizing the Attribute Template

If you are using the Kerberos Automated setup with Active Directory, depending on your KDC policies, you can customize the attributes that Ambari sets when creating principals. On the Configure Kerberos step of the wizard, in the **Advanced kerberos-env** section, you have access to the Ambari Attribute Template. This template (which is based on the Apache Velocity templating syntax) can be modified to adjust which attributes are set on the principals and how those attribute values are derived.

The following table lists the set of computed attribute variables available if you choose to modify the template:

Attribute Variables	Example
$normalized_principal	nn/c6401.ambari.apache.org@EXAMPLE.COM
$principal_name	nn/c6401.ambari.apache.org
$principal_primary	nn
$principal_digest	[[MD5 hash of the $normalized_principal]]
$principal_instance	c6401.ambari.apache.org
$realm	EXAMPLE.COM
$password	[[password]]

2.8. Managing Admin Credentials

When you enable Kerberos, if you choose to use an **Existing MIT KDC** or **Existing Active Directory**, the Kerberos Wizard prompts for information related to the KDC, the KDC Admin Account credentials and the Service and Ambari principals. Once provided, Ambari will automatically create principals, generate keytabs and distribute keytabs to the hosts in the cluster. The services will be configured for Kerberos and the service components are restarted to authenticate against the KDC. This is the **Kerberos Automated Setup** option.

By default, Ambari will not retain the KDC Admin Account credentials you provide unless you have configured to encrypt the passwords stored in Ambari. If you have not configured Ambari for password encryption, you will be prompted to provide KDC Admin Account credentials whenever cluster changes are made that require KDC principal and/or keytab changes (such as adding services, components and hosts).

If you have configured Ambari for password encryption, you will have an option to Save Admin Credentials. Ambari will use the retained KDC Admin Account credentials to make the KDC changes automatically.

Important

If you **do not** have password encryption enabled for Ambari, the Save Admin Credentials option **will not be** enabled.

Updating KDC Credentials

If you have chosen to Save Admin Credentials when enabling Kerberos, you can update or remove the credentials from Ambari using the following:

1. In Ambari Web, browse to **Admin > Kerberos** and click the **Manage KDC Credentials** button. The **Manage KDC Credentials** dialog is displayed.

2. If credentials have been previously saved, click **Remove** to remove the credentials currently stored in Ambari. Once removed, if cluster changes that require KDC principal

and/or keytab changes (such as adding services, components and hosts), you will be prompted to enter the KDC Admin Account credentials.

3. Alternatively, to update the KDC Admin Account credentials, enter the Admin principal and password values and click **Save**.

3. Advanced Security Options for Ambari

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

3.1. Configuring 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.

Note

When synchronizing LDAP users and groups, Ambari uses LDAP results paging controls to synchronize large numbers of LDAP objects. Most modern LDAP servers support these control, but for those that do not, such as Oracle Directory Server Enterprise Edition 11g, Ambari introduces a configuration parameter to disable pagination. The `authentication.ldap.pagination.enabled` property can be set to false in the `/etc/ambari-server/conf/ambari-properties` file to disable result paging controls. This will limit the maximum number of entities that can be imported at any given time to the maximum result limit of the LDAP server. To work around this, import sets of users or groups using the -

`users` and `-groups` options covered in section 3.1.4 - Specific Set of Users and Groups.

3.1.1. Setting Up LDAP User Authentication

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

Note

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.

Ambari Server LDAP Properties

Property	Values	Description
authentication.ldap.primaryUrl	server:port	The hostname and port for the LDAP or AD server. Example: my.ldap.server:389
authentication.ldap.secondaryUrl	server:port	The hostname and port for the secondary LDAP or AD server. Example: my.secondary.ldap.server:389 This is an optional value.
authentication.ldap.useSSL	true or false	If true, use SSL when connecting to the LDAP or AD server.
authentication.ldap.usernameAttribute	[LDAP attribute]	The attribute for username. Example: uid
authentication.ldap.baseDn	[Distinguished Name]	The root Distinguished Name to search in the directory for users. Example: ou=people,dc=hadoop,dc=apache,dc=org
authentication.ldap.referral	[Referral method]	Determines if LDAP referrals should be followed, or ignored.
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.groupObjectClass	[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.

3.1.2. Configure Ambari to use LDAP Server

Note

Only if you are using LDAPS, and 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 /etc/ambari-server/keys`

   where the keys directory does not exist, but should be created.

2. `$JAVA_HOME/bin/keytool -import -trustcacerts -alias root -file $PATH_TO_YOUR_LDAPS_CERT -keystore /etc/ambari-server/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 object class*` prompt, enter the object class that is used for users.

5. At the `User name attribute*` prompt, enter your selection. The default value is `uid`.

6. At the `Group object class*` prompt, enter the object class that is used for groups.

7. At the `Group name attribute*` prompt, enter the attribute for group name.

8. At the `Group member attribute*` prompt, enter the attribute for group membership.

9. At the `Distinguished name attribute*` prompt, enter the attribute that is used for the distinguished name.

10. At the `Base DN*` prompt, enter your selection.

11. At the `Referral method*` prompt, enter to `follow` or `ignore` LDAP referrals.

12. At the `Bind anonymously*` prompt, enter your selection.

13. At the `Manager DN*` prompt, enter your selection if you have set **bind.Anonymously** to false.

14. At the `Enter the Manager Password*` prompt, enter the password for your LDAP manager DN.

15. 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:

      - At the `TrustStore type` prompt, enter `jks`.
      - At the `Path to TrustStore file` prompt, enter `/keys/ldaps-keystore.jks` (or the actual path to your keystore file).
      - 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`.

      - 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.

      - 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.

16. Review your settings and if they are correct, select `y`.

17. Start or restart the Server

    ```
    ambari-server restart
    ```

    The users you have just imported are initially granted the Ambari User privilege. 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.

3.1.2.1. Example Active Directory Configuration

Directory Server implementations use specific object classes and attributes for storing identities. In this example, configurations specific to Active Directory are displayed as an example. Only those properties that are specific to Active Directory are displayed.

Run `ambari-server setup-ldap` and provide the following information about your Domain.

Prompt	Example AD Values
User object class* (posixAccount)	user
User name attribute* (uid)	sAMAccountName
Group object class* (posixGroup)	group
Group member attribute* (memberUid)	member
Distinguished name attribute* (dn)	distinguishedName

3.1.3. Synchronizing LDAP Users and Groups

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

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

Note

To perform this operation, your Ambari Server must be running.

- When prompted, you must provide credentials for an Ambari Admin.
- When syncing ldap, Local user accounts with matching username will switch to LDAP type, which means their authentication will be against the external LDAP and not against the Local Ambari user store.
- LDAP sync only syncs up-to-1000 users. If your LDAP contains over 1000 users and you plan to import over 1000 users, you must use the –users option when syncing and specify a filtered list of users to perform import in batches.

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

Review log files for failed synchronization attempts, at `/var/log/ambari-server/ambari-server.log` on the Ambari Server host.

Note

When synchronizing LDAP users and groups, Ambari uses LDAP results paging controls to synchronize large numbers of LDAP objects. Most

modern LDAP servers support these control, but for those that do not, such as Oracle Directory Server Enterprise Edition 11g, Ambari introduces a configuration parameter to disable pagination. The `authentication.ldap.pagination.enabled` property can be set to false in the `/etc/ambari-server/conf/ambari-properties` file to disable result paging controls. This will limit the maximum number of entities that can be imported at any given time to the maximum result limit of the LDAP server. To work around this, import sets of users or groups using the `-users` and `-groups` options covered in section 3.1.4 - Specific Set of Users and Groups.

3.1.4. 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. The comma separated entries in each of these files should be based off of the values in LDAP of the attributes chosen during setup. The "User name attribute" should be used for the users.txt file, and the "Group name attribute" should be used for the groups.txt file. This command will find, import, and synchronize the matching LDAP entities with Ambari.

Note

Group membership is determined using the Group Membership Attribute (groupMembershipAttr) specified during setup-ldap. User name is determined by using the Username Attribute (usernameAttribute) specified during setup-ldap.

3.1.5. 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.

Note

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

3.1.6. All Users and Groups

Important

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.

3.2. Setting Up Hadoop Group Mapping for LDAP/AD

To ensure that LDAP/AD group level authorization is enforced in Hadoop, you should set up Hadoop group mapping for LDAP/AD.

Prerequisites: Access to LDAP and the connection details. Note that LDAP settings can vary depending on what LDAP implementation you are using.

There are three ways to set up Hadoop group mapping:

- Configure Hadoop Group Mapping for LDAP/AD Using SSSD (Recommended) [23]
- Configure Hadoop Group Mapping in core-site.xml [23]
- Manually Create the Users and Groups in the Linux Environment [25]

3.2.1. Configure Hadoop Group Mapping for LDAP/AD Using SSSD (Recommended)

The recommended method for group mapping is to use SSSD or one of the following services to connect the Linux OS with LDAP:

- Centrify
- NSLCD
- Winbind
- SAMBA

Note that most of these services allow you to not only look up a user and enumerate their groups, but also allow you to perform other actions on the host. None of these features are required for LDAP group mapping on Hadoop – all that is required is the ability to lookup (or "validate") a user within LDAP and enumerate their groups. Therefore, when evaluating these services, take the time to understand the difference between the NSS module (which performs user/group resolution) and the PAM module (which performs user authentication). NSS is required. PAM is not required, and may represent a security risk.

3.2.2. Configure Hadoop Group Mapping in core-site.xml

You can use the following steps to configure Hadoop to use LDAP-based group mapping in `core-site.xml`.

1. Add the properties shown in the example below to the `core-site.xml` file. You will need to provide the value for the bind user, the bind password, and other properties specific to you LDAP instance, and make sure that object class, user, and group filters match the values specified in your LDAP instance.

```
<property>
<name>hadoop.security.group.mapping</name>
<value>org.apache.hadoop.security.LdapGroupsMapping</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.bind.user</name>
<value>cn=Manager,dc=hadoop,dc=apache,dc=org</value>
</property>

<!–
<property>
<name>hadoop.security.group.mapping.ldap.bind.password.file</name>
<value>/etc/hadoop/conf/ldap-conn-pass.txt</value>
</property>
–>

<property>
<name>hadoop.security.group.mapping.ldap.bind.password</name>
<value>hadoop</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/dc=hadoop,dc=apache,dc=org</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/dc=hadoop,dc=apache,dc=org</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.base</name>
<value></value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.user</name>
<value>(&amp;(|(objectclass=person)(objectclass=applicationProcess))(cn=
{0}))</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.group</name>
<value>(objectclass=groupOfNames)</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.member</name>
<value>member</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.group.name</name>
<value>cn</value>
</property>
```

2. Depending on your configuration, you may be able to refresh user and group mappings using the following HDFS and YARN commands:

```
hdfs dfsadmin -refreshUserToGroupsMappings
yarn rmadmin -refreshUserToGroupsMappings
```

   If a restart is required, you can use the applicable instructions on this page to re-start the HDFS NameNode and the YARN ResourceManager.

3. Verify LDAP group mapping by running the `hdfs groups` command. This command will fetch groups from LDAP for the current user. Note that with LDAP group mapping configured, the HDFS permissions can leverage groups defined in LDAP for access control.

3.2.3. Manually Create the Users and Groups in the Linux Environment

You can also manually create users and groups in your Linux environment.

3.3. Configuring Ambari for Non-Root

In most secure environments, restricting access to and limiting services that run as root is a hard requirement. For these environments, Ambari can be configured to operate without direct root access. Both Ambari Server and Ambari Agent components allow for non-root operation, and the following sections will walk you through the process.

- How to Configure Ambari Server for Non-Root [25]
- How to Configure an Ambari Agent for Non-Root [26]

3.3.1. How to Configure Ambari Server for Non-Root

You can configure the Ambari Server to run as a non-root user.

During the ambari-server setup process, when prompted to `Customize user account for ambari-server daemon?`, choose `y`.

The setup process prompts you for the appropriate, non-root user to run the Ambari Server as; for example: ambari.

Note

The non-root user you choose to run the Ambari Server should be part of the Hadoop group. This group must match the service Hadoop group accounts referenced in the Customize Services > Misc tab during the Install Wizard configuration step. The default group name is `hadoop` but if you customized this value during cluster install, be sure to make the non-root user a part of that group. See Customizing HDP Services for more information on service account users and groups.

Note

If Ambari Server is running as a non-root user, such as 'ambari', and you are planning on using Ambari Views, the following properties in **Services > HDFS > Configs > Advanced core-site** must be added:

```
hadoop.proxyuser.ambari.groups=*
hadoop.proxyuser.ambari.hosts=*
```

3.3.2. How to Configure an Ambari Agent for Non-Root

You can configure the Ambari Agent to run as a non-privileged user as well. That user requires specific sudo access in order to su to Hadoop service accounts and perform specific privileged commands. Configuring Ambari Agents to run as non-root requires that you manually install agents on all nodes in the cluster. For these details, see Installing Ambari Agents Manually. After installing each agent, you must configure the agent to run as the desired, non-root user. In this example we will use the `ambari` user.

Change the `run_as_user` property in the `/etc/ambari-agent/conf/ambari-agent.ini` file, as illustrated below:

```
run_as_user=ambari
```

Once this change has been made, the ambari-agent must be restarted to begin running as the non-root user.

The non-root functionality relies on sudo to run specific commands that require elevated privileges as defined in the Sudoer Configuration. The sudo configuration is split into three sections: Customizable Users, Non-Customizable Users, Commands, and Sudo Defaults.

3.3.2.1. Sudoer Configuration

The Customizable Users, Non-Customizable Users, Commands, and Sudo Defaults sections will cover how sudo should be configured to enable Ambari to run as a non-root user. Each of the sections includes the specific sudo entries that should be placed in `/etc/sudoers` by running the `visudo` command.

3.3.2.2. Customizable Users

This section contains the `su` commands and corresponding Hadoop service accounts that are configurable on install:

```
# Ambari Customizable Users
ambari ALL=(ALL) NOPASSWD:SETENV: /bin/su hdfs *,/bin/su ambari-qa *,/bin/su
 ranger *,/bin/su zookeeper *,/bin/su knox *,/bin/su falcon *,/bin/su ams *,
 /bin/su flume *,/bin/su hbase *,/bin/su spark *,/bin/su accumulo *,/bin/su
 hive *,/bin/su hcat *,/bin/su kafka *,/bin/su mapred *,/bin/su oozie *,/bin/
su sqoop *,/bin/su storm *,/bin/su tez *,/bin/su atlas *,/bin/su yarn *,/bin/
su kms *
```

Note

These user accounts must match the service user accounts referenced in the `Customize Services > Misc` tab during the Install Wizard configuration

step. For example, if you customize YARN to run as xyz_yarn, modify the `su` command above to be `/bin/su xyz_yarn`.

These user accounts must match the service user accounts referenced in the `Customize Services > Misc` tab during the Install Wizard configuration step. For example, if you customize YARN to run as xyz_yarn, modify the `su` command above to be `/bin/su xyz_yarn`.

3.3.2.3. Non-Customizable Users

This section contains the su commands for the system accounts that cannot be modified, and is **only required** if you are using the Ambari-installed-and-managed MySQL instance for the Hive Metastore. If you choose to use an existing MySQL, PostgreSQL or Oracle database for the Hive Metastore, you do not need to include this sudoer command.

```
# Ambari Non-Customizable Users
ambari ALL=(ALL) NOPASSWD:SETENV: /bin/su mysql *
```

3.3.2.4. Commands

This section contains the specific commands that must be issued for standard agent operations:

```
# Ambari Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/bin/yum,/usr/bin/zypper,/usr/bin/apt-
get, /bin/mkdir, /usr/bin/test, /bin/ln, /bin/chown, /bin/chmod, /bin/chgrp,
 /usr/sbin/groupadd, /usr/sbin/groupmod, /usr/sbin/useradd, /usr/sbin/usermod,
 /bin/cp, /usr/sbin/setenforce, /usr/bin/test, /usr/bin/stat, /bin/mv, /bin/
sed, /bin/rm, /bin/kill, /bin/readlink, /usr/bin/pgrep, /bin/cat, /usr/bin/
unzip, /bin/tar, /usr/bin/tee, /bin/touch, /usr/bin/hdp-select, /usr/bin/
conf-select, /usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh, /usr/lib/
hadoop/bin/hadoop-daemon.sh, /usr/lib/hadoop/sbin/hadoop-daemon.sh, /sbin/
chkconfig gmond off, /sbin/chkconfig gmetad off, /etc/init.d/httpd *, /sbin/
service hdp-gmetad start, /sbin/service hdp-gmond start, /usr/sbin/gmond, /
usr/sbin/update-rc.d ganglia-monitor *, /usr/sbin/update-rc.d gmetad *, /etc/
init.d/apache2 *, /usr/sbin/service hdp-gmond *, /usr/sbin/service hdp-gmetad
 *, /sbin/service mysqld *, /usr/bin/python2.6 /var/lib/ambari-agent/data/tmp/
validateKnoxStatus.py *, /usr/hdp/current/knox-server/bin/knoxcli.sh *, /usr/
bin/dpkg *, /bin/rpm *, /usr/sbin/hst *
```

```
# Ambari Ranger Commands
ambari ALL=(ALL) NOPASSWD:SETENV: /usr/hdp/*/ranger-usersync/setup.sh, /usr/
bin/ranger-usersync-stop, /usr/bin/ranger-usersync-start, /usr/hdp/*/ranger-
admin/setup.sh *, /usr/hdp/*/ranger-knox-plugin/disable-knox-plugin.sh *, /
usr/hdp/*/ranger-storm-plugin/disable-storm-plugin.sh *, /usr/hdp/*/ranger-
hbase-plugin/disable-hbase-plugin.sh *, /usr/hdp/*/ranger-hdfs-plugin/disable-
hdfs-plugin.sh *,  /usr/hdp/current/ranger-admin/ranger_credential_helper.py,
 /usr/hdp/current/ranger-kms/ranger_credential_helper.py, /usr/hdp/*/ranger-*/
ranger_credential_helper.py
```

Important

Do not modify the command lists, only the usernames in the Customizable Users section may be modified.

To re-iterate, you must do this sudo configuration on every node in the cluster. To ensure that the configuration has been done properly, you can su to the ambari user and run sudo

-l. There, you can double check that there are no warnings, and that the configuration output matches what was just applied.

3.3.2.5. Sudo Defaults

Some versions of sudo have a default configuration that prevents sudo from being invoked from a non-interactive shell. In order for the agent to run it's commands non-interactively, some defaults need to be overridden.

```
Defaults exempt_group = ambari
Defaults !env_reset,env_delete-=PATH
Defaults: ambari !requiretty
```

To re-iterate, this sudo configuration must be done on every node in the cluster. To ensure that the configuration has been done properly, you can su to the ambari user and run sudo -l. There, you can double-check that there are no warnings, and that the configuration output matches what was just applied.

3.4. Optional: Encrypt Database and LDAP Passwords

If you plan to configure Ambari to retain the Kerberos KDC Admin Account credentials when Configuring Kerberos, or you wish to encrypt the Ambari database and LDAP server passwords, you need to setup encryption for the passwords stored in the Ambari database.

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
   ```

   a. When prompted, select Option 2 to "Encrypt the passwords stored in ambari.properties file".

   b. Provide a master key for encrypting the passwords. You are prompted to enter the key twice for accuracy.

   c. Once the passwords are encrypted, you need access to the master key to start Ambari Server. You have three options for maintaining the master key:

      - Persist it to a file on the server by pressing y at the prompt.
      - 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.

   d. Start the Ambari Server:

      ```
      ambari-server start
      ```

3.4.1. Reset Encryption

There may be situations in which you want to:

- Remove Encryption Entirely [29]
- 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.

3.4.2. 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.

3.4.3. 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
     ```

     a. Select Option 2: Choose one of the following options:

     - [1] Enable HTTPS for Ambari server.
     - [2] Encrypt passwords stored in ambari.properties file.
     - [3] Setup Ambari kerberos JAAS configuration.

     b. Enter the current master key when prompted if necessary (if it is not persisted or set as an environment variable).

     c. At the `Do you want to reset Master Key` prompt, enter `yes`.

     d. At the prompt, enter the new master key and confirm.

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

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

  ```
  ambari-server restart
  ```

3.5. Optional: Set Up SSL for Ambari

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
   ```

   - Select `1` for `Enable HTTPS for Ambari server`.

- Respond y to `Do you want to configure HTTPS ?`
- Select the port you want to use for SSL. The default port number is 8443.
- Provide the complete path to your certificate file ($wserver.crt from above) and private key file ($wserver.key from above).
- Provide the password for the private key.
- Start or restart the Server

  ```
  ambari-server restart
  ```

4. Trust Store Setup - If you plan to use Ambari Views with your Ambari Server, after enabling SSL for Ambari using the instructions below, you must also configure a Truststore for the Ambari Server. Refer to Set Up Truststore for Ambari Server for more information.

Note

If you have deployed the Tez View with your Ambari Server, after enabling SSL for Ambari, you must also change the property **tez.tez-ui.history-url.base** in **Service > Tez > Configs > Advanced > Advanced tez-site** section. Refer to Create Tez View Instance for more information about **tez.tez-ui.history-url.base** property for the Tez View.

3.6. Optional: Ambari Web Inactivity Timeout

Ambari is capable of automatically logging a user out of Ambari Web after a period of inactivity. After a configurable amount of time, the user's session will be terminated and they will be redirected to the login page.

This capability can be separately configured for Operators and Read-Only users. This allows you to distinguish a read-only user (useful when Ambari Web is used as a monitoring dashboard) from other operators. Alternatively, you can set both inactivity timeout values to be the same so that regardless of the user type, automatic logout will occur after a set period of time.

By default, the Ambari Web inactivity timeout is not enabled (i.e. is set to 0). The following instructions should be used to enable inactivity timeout and set as the amount of time in seconds before users are automatically logged out.

Ensure the Ambari Server is completely stopped before making changes to the inactivity timeout. Either make these changes before you start Ambari Server the first time, or bring the server down before making these changes.

1. On the Ambari Server host, open

   `/etc/ambari-server/conf/ambari.properties` with a text editor.

2. There are two properties for the inactivity timeout setting. Both are initially set to 0 (which means this capability is disabled).

Property	Description
user.inactivity.timeout.default	Sets the inactivity timeout (in seconds) for all users except Read-Only users.
user.inactivity.timeout.role.readonly.default	Sets the inactivity timeout (in seconds) for all Read-Only users.

3. Modify the values to enable the capability. The values are in seconds.
4. Save changes and restart Ambari Server.
5. After a user logs into Ambari Web, once a period of inactivity occurs, the user will be presented with an Automatic Logout dialog 60 seconds from logout. The user can click to remain logged in or if no activity occurs, Ambari Web will automatically log the user out and redirect the application to the login page.

3.7. Set Up Kerberos for Ambari Server

When a cluster is enabled for Kerberos, the component REST endpoints (such as the YARN ATS component) require SPNEGO authentication.

Depending on the Services in your cluster, Ambari Web needs access to these APIs. As well, views such as the Tez View need access to ATS. Therefore, the Ambari Server requires a Kerberos principal in order to authenticate via SPNEGO against these APIs. This section describes how to configure Ambari Server with a Kerberos principal and keytab to allow views to authenticate via SPNEGO against cluster components.

1. Create a principal in your KDC for the Ambari Server. For example, using kadmin:

```
addprinc -randkey ambari-server@EXAMPLE.COM
```

2. Generate a keytab for that principal.

```
xst -k ambari.server.keytab ambari-server@EXAMPLE.COM
```

3. Place that keytab on the Ambari Server host. Be sure to set the file permissions so the user running the Ambari Server daemon can access the keytab file.

```
/etc/security/keytabs/ambari.server.keytab
```

4. Stop the ambari server.

```
ambari-server stop
```

5. Run the setup-security command.

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

6. Select 3 for Setup Ambari kerberos JAAS configuration.
7. Enter the Kerberos principal name for the Ambari Server you set up earlier.
8. Enter the path to the keytab for the Ambari principal.
9. Restart Ambari Server.

```
ambari-server restart
```

3.8. Set Up Truststore for Ambari Server

If you plan to Set Up SSL for Ambari or to enable wire encryption for HDP, you must configure the Truststore for Ambari and add certificates.

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. On the Ambari Server, create a new keystore that will contain the Ambari Server's HTTPS certificate.

```
keytool -import -file <path_to_the_Ambari_Server's_SSL_Certificate> -alias
 ambari-server -keystore ambari-server-truststore
```

   When prompted to 'Trust this certificate?' type "yes".

2. Configure the ambari-server to use this new trust store:

```
ambari-server setup-security
Using python  /usr/bin/python2.6
Security setup options...
===========================================================================
Choose one of the following options:
  [1] Enable HTTPS for Ambari server.
  [2] Encrypt passwords stored in ambari.properties file.
  [3] Setup Ambari kerberos JAAS configuration.
  [4] Setup truststore.
  [5] Import certificate to truststore.
===========================================================================
Enter choice, (1-5): *4*
Do you want to configure a truststore [y/n] (y)? *y*
TrustStore type [jks/jceks/pkcs12] (jks): *jks*
Path to TrustStore file : *<path to the ambari-server-truststore keystore>*
Password for TrustStore:
Re-enter password:
Ambari Server 'setup-security' completed successfully.
```

3. Once configured, the Ambari Server must be restarted for the change to take effect.

```
ambari-server restart
```

3.9. 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.

3.10. 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
   ```

3.11. Optional: HTTP Cookie Persistence

During HTTP authentication, a cookie is dropped. This is a persistent cookie that is valid across browser sessions. For clusters that require enhanced security, it is desirable to have a session cookie that gets deleted when the user closes the browser session.

In HDP-2.3.4 and higher versions, you can use the following property in the `etc/hadoop/conf/core-site.xml` file to specify cookie persistence across browser sessions.

```
<property>
   <name>hadoop.http.authentication.cookie.persistent</name>
   <value>true</value>
</property>
```

The default value for this property is `false`.

4. Enabling SPNEGO Authentication for Hadoop

By default, access to the HTTP-based services and UI's for the cluster are not configured to require authentication. Kerberos authentication can be configured for the Web UIs for HDFS, YARN, MapReduce2, HBase, Oozie, Falcon and Storm.

4.1. Configure Ambari Server for Authenticated HTTP

In order for Ambari to work with a cluster in which authenticated HTTP access to the Web UI's is required, you must configure the Ambari Server for Kerberos. Refer to Set Up Kerberos for Ambari Server for more information.

4.2. Configuring HTTP Authentication for HDFS, YARN, MapReduce2, HBase, Oozie, Falcon and Storm

1. Create a secret key used for signing authentication tokens. This file should contain random data and be placed on every host in the cluster. It should also be owned by the hdfs user and group owned by the hadoop group. Permissions should be set to 440. For example:

   ```
   dd if=/dev/urandom of=/etc/security/http_secret bs=1024 count=1

   chown hdfs:hadoop /etc/security/http_secret

   chmod 440 /etc/security/http_secret
   ```

2. In Ambari Web, browse to **Services > HDFS > Configs** .

3. Add or modify the following configuration properties to Advanced core-site .

Property	New Value
hadoop.http.authentication.simple.anonymous.allowed	false
hadoop.http.authentication.signature.secret.file	/etc/security/http_secret
hadoop.http.authentication.type	kerberos
hadoop.http.authentication.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab
hadoop.http.authentication.kerberos.principal	HTTP/_HOST@**EXAMPLE.COM**
hadoop.http.filter.initializers	org.apache.hadoop.security.AuthenticationFilterInitializer

Property	New Value
hadoop.http.authentication.cookie.domain	**hortonworks.local**

Important

The entries listed in the above table in **bold** and italicized are site-specific. The hadoop.http.authentication.cookie.domain property is based off of the fully qualified domain names of the servers in the cluster. For example if the FQDN of your NameNode is host1.hortonworks.local, the hadoop.http.authentication.cookie.domain should be set to hortonworks.local.

4. Save the configuration, then restart the affected services.