Hortonworks Data Platform

Ambari Security Guide

(Apr 13, 2015)

docs.hortonworks.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.

1.1. 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.

1.1.1. 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.

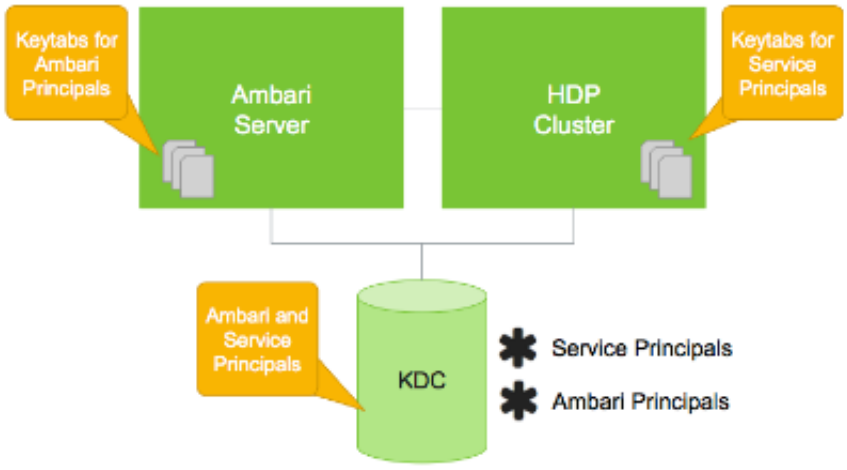
Because cluster resources (hosts or services) cannot provide a password each time to decrypt the TGT, they use a special file, called a **keytab**, which contains the resource principal's authentication credentials. The set of hosts, users, and services over which the Kerberos server has control is called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

1.1.2. 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.

1.1.3. 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.

- Use an Existing MIT KDC
- Use an Existing Active Directory
- (Optional) Install a new MIT KDC

1.1.3.1. Use an Exisiting 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.

1.1.3.2. Use an Existing Active Directory Domain

To use an existing Active Directory domain for the cluster, 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.

1.1.3.3. (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 6

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

   SLES 11

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

   Ubuntu 12

   ```
   apt-get install krb5 krb5-server krb5-client
   ```

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

   ```
   /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
   ```

Create the Kerberos Database

- Use the utility kdb5_util to create the Kerberos database.

  RHEL/CentOS/Oracle Linux 6

  ```
  kdb5_util create -s
  ```

  SLES 11

  ```
  kdb5_util create -s
  ```

  Ubuntu 12

  ```
  kdb5_util create -s
  ```

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
  ```

  SLES 11

  ```
  rckrb5kdc start rckadmind start
  ```

  Ubuntu 12

  ```
  rckrb5kdc start rckadmind start
  ```

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
```

SLES 11

```
chkconfig rckrb5kdc on chkconfig rckadmind on
```

Ubuntu 12

```
update-rc.d rckrb5kdc defaults update-rc.d rckadmind
defaults
```

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**.

   RHEL/CentOS/Oracle Linux 6

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

   SLES 11

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

   Ubuntu 12

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

2. **Confirm that this admin principal has permissions in the KDC ACL.**

   For example, on RHEL/CentOS, check the /var/kerberos/krb5kdc/kadm5.acl file has an entry like so to allow the */admin principal to administer the KDC for your specific realm. In this case, for the EXAMPLE.COM realm: */admin@EXAMPLE.COM *. When using a realm that is different than EXAMPLE.COM, ensure there is an entry for the realm you are using. If not present, principal creation will fail. After editing the kadm5.acl, you must restart the kadmind process.

1.1.4. Enabling Kerberos Security in Ambari

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
- Running the Kerberos Wizard

1.1.4.1. 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. Depending on your choice of JDK and if your Ambari Server has Internet Access, Ambari has a few options and actions for you to pursue.

JCE Options and Actions

Scenario	Action
If you have Internet Access and selected Oracle JDK 1.6 or Oracle JDK 1.7 during Ambari Server setup.	Ambari automatically downloaded the JCE policy files (that match the JDK) and installed the JCE onto the Ambari Server. **You must** distribute and install the JCE **on all hosts.**
If you have Internet Access and selected Custom JDK during Ambari Server setup.	The JCE has not been downloaded or installed on the Ambari Server or the hosts in the cluster. **You must** distribute and install the JCE **on Ambari and all hosts.**
If you do not have Internet Access and selected Custom JDK during Ambari Server setup.	The JCE has not been downloaded or installed on the Ambari Server or the hosts in the cluster. **You must** distribute and install the JCE **on Ambari and all hosts.**
If you have a previous Ambari install and upgraded to Ambari 2.0.0.	The JCE has not been downloaded or installed on the Ambari Server or the hosts in the cluster. **You must** distribute and install the JCE **on Ambari and all hosts.**

1.1.4.2. Distribute and Install the JCE

1. On the Ambari Server, obtain the JCE policy file appropriate for the JDK version in your cluster.
   - For Oracle JDK 1.6:

     http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.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 UnlimitedJCEPolicyJDK7.zip -d /usr/jdk64/jdk1.7.0_67/jre/lib/security/`
4. Restart Ambari Server.

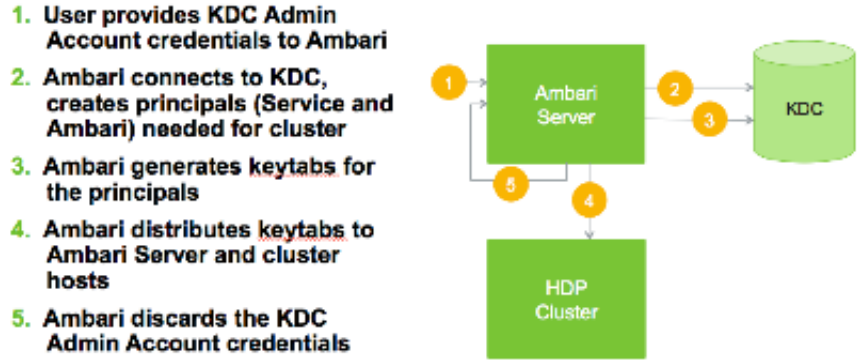
5. Proceed to Running the Security Wizard.

1.1.4.3. Running the Kerberos Wizard

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.

Important

Since Ambari will automatically create principals in the KDC and generate keytabs, you must have Kerberos Admin Account credentials available when running the wizard.



High-Level View of Principal Creation, Keytab Generation, and Distribution Flow

Launching the Kerberos Wizard

1. Be sure you've 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.
6. Proceed with the install. (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. (Optional) If you need to customize the attributes for the principals Ambari will create, see the Customizing the Attribute Template for more information.
7. 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.
8. Customize the Kerberos identities used by Hadoop and proceed to kerberize the cluster.

Note

Pay particular attention to the Ambari principal names. For example, if you want the Ambari Smoke User Principal name to be unique and include the cluster name , you can append `${cluster_name}` to the identity setting. `${cluster-env/smokeuser}-${cluster_name}@{realm}`

9. 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 you 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.

1.1.5. Customizing the Attribute Template

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]]

1.1.6. Kerberos Client Packages

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 5 + 6	krb5-workstation
SLES 11	krb5-client

Operating System	Packages
Ubuntu 12	krb5-user, krb5-config

1.1.7. Post-Kerberos Wizard User/Group Mapping

If you have chosen to use existing MIT or Active Directory Kerberos infrastructure with your cluster, it is important to tell the cluster how to map usernames from those existing systems to principals within the cluster. This is required to properly translate username syntaxes from existing systems to Hadoop to ensure usernames can be mapped successfully.

Hadoop uses a rule-based system to create mappings between service principals and their related UNIX usernames. The rules are specified using the configuration property hadoop.security.auth_to_local as part of core-site.

The default rule is simply named DEFAULT. It translates all principals in your default domain to their first component. For example, myusername@EXAMPLE.COM and myusername/admin@EXAMPLE.COM both become myusername, assuming your default domain is EXAMPLE.COM. In this case, EXAMPLE.COM represents the Kerberos realm, or Active Directory Domain that is being used.

1.1.8. Creating Auth-to-Local Rules

To accommodate more complex translations, you can 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 username 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@EXAMPLE.COM to myusername@EXAMPLE.COM [2:$1] translates myusername/admin@EXAMPLE.COM to myusername [2:$1%$2] translates myusername/admin@EXAMPLE.COM to “myusername %admin

- **The Filter**

  The filter consists of a regular expression (regex) in a parentheses. It 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 X's in the name with Y

Examples

- If your default realm was EXAMPLE.COM, but you also wanted to take all principals from ACME.COM that had a single component joe@ACME.COM, the following rule would do this:

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

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

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

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

  RULE[2:$1%$2@$0](.%admin@EXAMPLE.COM)s/./admin/ DEFAULT

After your mapping rules have been configured and are in place, Hadoop uses those rules to map principals to UNIX users. By default, Hadoop uses the UNIX shell to resolve a user's UID, GID, and list of associated groups for secure operation on every node in the cluster. This is because in a kerberized cluster, individual tasks run as the user who submitted the application. In this case, the user's identity is propagated all they way down to local JVM processes to ensure tasks are run as the user who submitted them. For this reason, typical enterprise customers choose to use technologies such as PAM, SSSD, Centrify, or other solutions to integrate with a corporate directory. As Linux is commonly used in the enterprise, there is most likely an existing enterprise solution that has been adopted for your organization. The assumption going forward is that such a solution has been integrated successfully, so logging into each individual DataNode using SSH can be accomplished using LDAP credentials, and typing in id results in a UID, GID, and list of associated groups being returned.

Note

If you use Hue, you must install and configure Hue manaully, after running the Kerberos wizard. For information about installing Hue manually, see Installing Hue .

2. 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 SSL for Ambari
- Set Up Kerberos for Ambari Server
- Set Up Two-Way SSL Between Ambari Server and Ambari Agents

2.1. 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.

2.1.1. 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

Property	Values	Description
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.

2.1.2. 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.

2.1.3. 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)	cn
Group object class* (posixGroup)	group
Group member attribute* (memberUid)	member

2.1.4. 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.

2.1.5. 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.

2.1.6. 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.

2.1.7. 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.

2.2. 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
   ```

   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. 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.

   c. 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 or restart the Server

```
ambari-server restart
```

2.2.1. 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.

2.2.2. 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.

2.2.3. 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
    ```

2.3. Optional: Set Up SSL for Ambari

2.3.1. 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
   ```

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

     ```
     ambari-server restart
     ```

2.4. Optional: 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 Jobs View and 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.

   ```
   /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
```

2.5. 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.

2.6. 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
```