Cloudera Manager 7.1.2

Configuring Authentication in Cloudera Manager

Date published: 2020-05-28
Date modified: 2020-07-10

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Overview

Authentication is a process that requires users and services to prove their identity when trying to access a system resource.

Organizations typically manage user identity and authentication through various time-tested technologies, including Lightweight Directory Access Protocol (LDAP) for identity, directory, and other services, such as group management, and Kerberos for authentication.

Cloudera clusters support integration with both of these technologies. For example, organizations with existing LDAP directory services such as Active Directory (included in Microsoft Windows Server as part of its suite of Active Directory Services) can leverage the organization's existing user accounts and group listings instead of creating new accounts throughout the cluster. Using an external system such as Active Directory or OpenLDAP can support the user role authorization mechanism.

For authentication, Cloudera supports integration with MIT Kerberos and with Active Directory. Microsoft Active Directory supports Kerberos for authentication in addition to its identity management and directory functionality, that is, LDAP.

Kerberos provides *strong authentication*, strong meaning that cryptographic mechanisms—rather than passwords alone—are used in the exchange between requesting process and service during the authentication process.

These systems are not mutually exclusive. For example, Microsoft Active Directory is an LDAP directory service that also provides Kerberos authentication services, and Kerberos credentials can be stored and managed in an LDAP directory service. Cloudera Manager Server, CDP nodes, and Cloudera Enterprise components, such as Apache Hive, Hue, and Impala, can all make use of Kerberos authentication.

Note: Cloudera does not provide a Kerberos implementation but rather can use MIT Kerberos or Microsoft Server Active Directory service and its KDC for authentication.

Configuring the cluster to use Kerberos requires having administrator privileges—and access to—the Kerberos server Key Distribution Center (KDC). The process may require debugging issues between the Cloudera Manager cluster and the KDC.

Note: Integrating clusters to use Microsoft Active Directory as a KDC requires the Windows registry setting for AllowTgtSessionKey to be disabled (set to 0). If this registry key has already been enabled, users and credentials are not created—despite the "Successful" message at the end of the configuration/integration process. Before configuring Active Directory for use as KDC, check the value of AllowTgtSessionKey on the Active Directory instance and reset to 0 if necessary.

On each host operating system underlying each node in a cluster, local Linux user:group accounts are created during installation of Cloudera Server and CDP services. To apply per-node authentication and authorization mechanism consistently across all the nodes of a cluster, local user:group accounts are mapped to user accounts and groups in an LDAP-compliant directory service, such as Active Directory or OpenLDAP.

To facilitate the authentication process from each host system (node in the cluster) to the LDAP directory, Cloudera recommends using additional software mechanisms such as SSSD or Centrify Server Suite.

Kerberos Security Artifacts Overview

How Cloudera Clusters use Kerberos artifacts such as principals, keytabs, and delegation tokens.

Cloudera recommends using Kerberos for authentication because native Hadoop authentication alone checks only for valid user:group membership in the context of HDFS, but does not authenticate users or services across all network resources, as Kerberos does. Unlike other mechanisms that may be far easier to deploy, the Kerberos protocol authenticates a requesting user or service for a specific period of time only, and each service that the user may want to use requires the appropriate Kerberos artifact in the context of the protocol. This section describes how Cloudera clusters use some of these artifacts, such as Kerberos principals and keytabs for user authentication, and how delegation tokens are used by the system to authenticate jobs on behalf of authenticated users at runtime.

Kerberos Principals

Each user and service that needs to authenticate to Kerberos needs a *principal*, an entity that uniquely identifies the user or service in the context of possibly multiple Kerberos servers and related subsystems. A principal includes up to three pieces of identifying information, starting with the user or service name (called a *primary*). Typically, the primary portion of the principal consists of the user account name from the operating system, such as jcarlos for the user's Unix account or hdfs for the Linux account associated with the service daemon on the host underlying cluster node.

Principals for users typically consist solely of the primary and the Kerberos *realm* name. The realm is a logical grouping of principals tied to the same Key Distribution Center (KDC) which is configured with many of the same properties, such as supported encryption algorithms. Large organizations may use realms as means of delegating administration to various groups or teams for specific sets of users or functions and distributing the authentication-processing tasks across multiple servers.

Standard practice is to use your organization's domain name as the Kerberos realm name (in all uppercase characters) to easily distinguish it as part of a Kerberos principal, as shown in this user principal pattern:

```
username@REALM.EXAMPLE.COM
```

The combination of the primary and the realm name can distinguish one user from another. For example, jcarlos@ SOME-REALM.EXAMPLE.COM and jcarlos@ANOTHER-REALM.EXAMPLE.COM may be unique individuals within the same organization.

For service role instance identities, the primary is the Unix account name used by Hadoop daemons (hdfs, mapred, and so on) followed by an *instance* name that identifies the specific host on which the service runs. For example, hdfs/hostname.fqdn.example.com@SOME-REALM.EXAMPLE.COM is an example of the principal for an HDFS service instance. The forward slash (/) separates the primary and the instance names using this basic pattern:

```
service-name/hostname.fqdn.example.com@REALM.EXAMPLE.COM
```

The HTTP principal needed for Hadoop web service interfaces does not have a Unix local account for its primary but rather is HTTP.

An instance name can also identify users with special roles, such as administrators. For example, the principal jcar los@SOME-REALM.COM and the principal jcarlos/admin@SOME-REALM.COM each have their own passwords and privileges, and they may or may not be the same individual.

For example, the principal for the HDFS service role instance running on a cluster in an organization with realms for each geographical location might be as follows:

```
hdfs/hostname.fqdn.example.com@OAKLAND.EXAMPLE.COM
```

Generally, the service name is the Unix account name used by the given service role instance, such as hdfs or mapred, as shown above. The HTTP principal for securing web authentication to Hadoop service web interfaces has no Unix account, so the primary for the principal is HTTP.

Kerberos Keytabs

A *keytab* is a file that contains the principal and the encrypted key for the principal. A keytab file for a Hadoop daemon is unique to each host since the principal names include the hostname. This file is used to authenticate a principal on a host to Kerberos without human interaction or storing a password in a plain text file. Because having access to the keytab file for a principal allows one to act as that principal, access to the keytab files should be tightly secured.

They should be readable by a minimal set of users, should be stored on local disk, and should not be included in host backups, unless access to those backups is as secure as access to the local host.

Delegation Tokens

Users in a Hadoop cluster authenticate themselves to the NameNode using their Kerberos credentials. However, once the user is authenticated, each job subsequently submitted must also be checked to ensure it comes from an authenticated user. Since there could be a time gap between a job being submitted and the job being executed, during which the user could have logged off, user credentials are passed to the NameNode using delegation tokens that can be used for authentication in the future.

Delegation tokens are a secret key shared with the NameNode, that can be used to impersonate a user to get a job executed. While these tokens can be renewed, new tokens can only be obtained by clients authenticating to the NameNode using Kerberos credentials. By default, delegation tokens are only valid for a day. However, since jobs can last longer than a day, each token specifies a NodeManager as a renewer which is allowed to renew the delegation token once a day, until the job completes, or for a maximum period of 7 days. When the job is complete, the NodeManager requests the NameNode to cancel the delegation token.

Token Format

The NameNode uses a random masterKey to generate delegation tokens. All active tokens are stored in memory with their expiry date (maxDate). Delegation tokens can either expire when the current time exceeds the expiry date, or, they can be canceled by the owner of the token. Expired or canceled tokens are then deleted from memory. The sequ enceNumber serves as a unique ID for the tokens. The following section describes how the Delegation Token is used for authentication.

```
TokenID = {ownerID, renewerID, issueDate, maxDate, sequenceNumber}
TokenAuthenticator = HMAC-SHA1(masterKey, TokenID)
Delegation Token = {TokenID, TokenAuthenticator}
```

Authentication Process

To begin the authentication process, the client first sends the TokenID to the NameNode. The NameNode uses this TokenID and the masterKey to once again generate the corresponding TokenAuthenticator, and consequently, the Delegation Token. If the NameNode finds that the token already exists in memory, and that the current time is less than the expiry date (maxDate) of the token, then the token is considered valid. If valid, the client and the NameNode will then authenticate each other by using the TokenAuthenticator that they possess as the secret key, and MD5 as the protocol. Since the client and NameNode do not actually exchange TokenAuthenticators during the process, even if authentication fails, the tokens are not compromised.

Token Renewal

Delegation tokens must be renewed periodically by the designated renewer (renewerID). For example, if a NodeManager is the designated renewer, the NodeManager will first authenticate itself to the NameNode. It will then send the token to be authenticated to the NameNode. The NameNode verifies the following information before renewing the token:

- The NodeManager requesting renewal is the same as the one identified in the token by renewerID.
- The TokenAuthenticator generated by the NameNode using the TokenID and the masterKey matches the one previously stored by the NameNode.
- The current time must be less than the time specified by maxDate.

If the token renewal request is successful, the NameNode sets the new expiry date to min(current time+renew perio d, maxDate). If the NameNode was restarted at any time, it will have lost all previous tokens from memory. In this case, the token will be saved to memory once again, this time with a new expiry date. Hence, designated renewers must renew all tokens with the NameNode after a restart, and before relaunching any failed tasks.

A designated renewer can also revive an expired or canceled token as long as the current time does not exceed maxD ate. The NameNode cannot tell the difference between a token that was canceled, or has expired, and one that was erased from memory due to a restart, since only the masterKey persists in memory. The masterKey must be updated regularly.

Kerberos Configuration Strategies for CDP

An overview of integration options when configuring Kerberos for Cloudera Data Platform (CDP).

Table 1: Kerberos Configuration Strategies for CDP

	A. Direct integration of Cloudera Manager with AD/ MIT/IPA KDC	**B. Keytab retrieval through “keytab retrieval script"**	**C. Manual deployment / configuration of keytab for services**
Brief description of functionality	Cloudera Manager can create/ delete service principals directly in the KDC as needed, and Cloudera Manager will retrieve the keytabs for the created principals automatically.	Cloudera Manager will call an external script to retrieve a keytab for a given principal.	Cloudera Manager will maintain keytab/kerberos related configuration for services with either the A or B column methods, but we will override the default location of the keytab with a “Safety Valve” option.
Example	This is the default/recommended setup. • Cloudera Manager will store/ track the available keytabs, and can create missing principals/keytabs after new services/role instances are added. • Cloudera Manager can recreate existing principals / keytabs if needed for some reason.	• Where direct integration with AD / KDC is not feasible due to internal restrictions/policy. • Could be used integrate Cloudera Manager with unsupported KDC types.	Could be used where customizations are needed to the keytabs due to external factors.
Principal naming convention	Kerberos principals used by CDP are following this naming convention: service/fqdn@REALM. Most services use a single username across the cluster (like “hdfs” or “hive”) for all roles of the service. The host part will vary based on the available hosts in the cluster, so a single service will likely need multiple principals, where the username part is the same, and the hostname will match one of the hosts where this service is running on. The default usernames used by the services can be found in “Hadoop Users (user:group) and Kerberos Principals”.		

	A. Direct integration of Cloudera Manager with AD/ MIT/IPA KDC	**B. Keytab retrieval through "keytab retrieval script"**	**C. Manual deployment / configuration of keytab for services**
Scripts involved	Cloudera Manager invokes these external shell scripts to perform credential management: Base directory: /opt/cloudera/cm/ bin delete_credentials_ad.sh delete_credentials_ipa.sh delete_credentials.sh gen_credentials_ad.sh gen_credentials_ipa.sh gen_credentials.sh import_credentials_ipa.sh import_credentials.sh merge_credentials.sh	The "custom keytab retrieval script" will get a single principal name matching the above naming convention, and it should write the matching keytab to a directory also passed as an option to the script. There is a basic sample script provided in "Using a custom Kerberos keytab retrieval scrip".	None
Persistent storage of keytabs	Keytabs that are successfully retrieved will be stored in the Cloudera Manager-backed database to be deployed and used on demand.		None. Required keytabs should be stored on the hosts manually.
On-demand keytab merging	In case a service (role instance) requires multiple principals for operation, Cloudera Manager will merge the individual keytabs into a single combined keytab automatically. For example: HDFS namenode uses both the "hdfs" and "HTTP" principal for operation. Cloudera Manager will create a merged keytab that contains both these principals.		None. In case a service (role instance) requires multiple principals for operation, keytabs provided for the services (role instance) are needed to be merged manually.
On-demand keytab deployment	Cloudera Manager will automatically deploy the required configuration and keytab files into the process directory of the given role instance through the Cloudera Manager Agent when starting a service/role instance.		None. All required keytabs will need to be present on the host(s) for a service before starting it. Cloudera Manager should be used to configure the location of these keytab files though.
Keytab filename	The keytab file & filename will be managed by Cloudera Manager. The keytab file will be deployed by the Cloudera Manager agent to the services process folder automatically. The filenames used by Cloudera Manager can be found in "Hadoop Users (user:group) and Kerberos Principals".		The name & location of the keytab file has to be specified manually, so it can be anything allowed by POSIX conventions.
Keytab file ownership and permissions	The "System User" of the respective service needs read-write access for the keytab file. Cloudera Manager will set the proper owner/group on the file with permission mode 600 (read-write for owner, no access for group/others) Default owner/group and permissions can also be found in "Hadoop Users (user:group) and Kerberos Principals".		The "System User" of the respective service needs read-write access for the keytab file. The proper permission and ownership of the file needs to be set manually.

Related Information

Hadoop Users (user:group) and Kerberos Principals

Using a custom Kerberos keytab retrieval script

Configuring Authentication in Cloudera Manager

Overview of configuring Cloudera authentication.

Cloudera clusters can be configured to use Kerberos for authentication using a manual configuration process or by using the configuration wizard available from the Cloudera Manager Admin Console. Cloudera recommends using the wizard because it automates many of the configuration and deployment tasks. In addition, enabling Kerberos the cluster using the wizard also enables Kerberos authentication for all CDP components set up on the cluster.

Warning: Before enabling Kerberos authentication, configure Cloudera Manager for TLS/SSL. Failure to do so results in Kerberos keytabs being transferred over the network unencrypted.

Cloudera Manager Kerberos Wizard Overview

The Cloudera Manager Kerberos wizard starts by verifying various details of the Kerberos instance that will be used by the cluster. Before using the wizard, be sure to gather all the details about the Kerberos service or engage the Kerberos administrator for help during this process. There are many details of the Kerberos instance, and you will need to enter them in the wizard pages.

The wizard requires a working KDC, either an MIT KDC, a FreeIPA server, or an Active Directory KDC. Make sure that the KDC is set up and working prior to starting the wizard. Administrator-level privileges to the Kerberos instance are required to complete the prompts of the wizard. If you do not have access to credentials with these privileges, the Kerberos administrator will need to assist you.

Given the information provided to the wizard entry screens, the configuration wizard does the following:

- Configures the necessary properties in all configuration files—core-site.xml, hdfs-site.xml, mapred-site.xml, and taskcontroller.cfg—to set Kerberos as the authentication mechanism for the cluster
- Configures the necessary properties in the oozie-site.xml and hue.ini files for Oozie and Hue for Kerberos authentication
- Creates principal and keytab files for core system users, such as hdfs and mapred, and for CDP services
- Distributes the keytab files to each host in the cluster
- Creates keytab files for oozie and hue users and deploys to the appropriate hosts that support these client-focused services
- Distributes a configured krb5.conf to all nodes in the cluster
- Stops all services
- Deploys client configurations
- Restarts all services throughout the cluster

Keytab file for...	Principals
hdfs	hdfs, host
mapred	mapred, host
oozie	oozie, HTTP
hue	hue

The host principal is the same in both hdfs and mapred keytab files.

After making the configuration changes and deploying the keytabs, and configuration files to the appropriate nodes in the cluster, Cloudera Manager starts all services to stand up the cluster.

- To use the Kerberos configuration wizard, see “Enabling Kerberos Authentication for CDP”.
- To configure Kerberos authentication manually, see “How to Configure Clusters to Use Kerberos for Authentication”.

Related Information

Enabling Kerberos Authentication for CDP

Cloudera Manager user accounts

Access to Cloudera Manager features is controlled by user accounts. A user account identifies how a user is authenticated and determines what privileges are granted to the user.

Minimum Required Role: User Administrator (also provided by Full Administrator) This feature is not available when using Cloudera Manager to manage Data Hub clusters.

When you are logged in to the Cloudera Manager Admin Console, the username you are logged in as is located at the far right of the top navigation bar.

A user with the User Administrator or Full Administrator role manages user accounts through the AdministrationUsers page. View active user sessions on the User Sessions tab.

User Authentication

Cloudera Manager provides several mechanisms for authenticating users. You can configure Cloudera Manager to authenticate users against the Cloudera Manager database or against an external authentication service. The external authentication service can be an LDAP server (Active Directory or an OpenLDAP compatible directory), or you can specify another external service. Cloudera Manager also supports using the Security Assertion Markup Language (SAML) to enable single sign-on.

If you are using LDAP or another external service, you can configure Cloudera Manager so that it can use both methods of authentication (internal database and external service), and you can determine the order in which it performs these searches. If you select an external authentication mechanism, Full Administrator users can always authenticate against the Cloudera Manager database. This prevents locking everyone out if the authentication settings are misconfigured, such as with a bad LDAP URL.

With external authentication, you can restrict login access to members of specific groups, and can specify groups whose members are automatically given Full Administrator access to Cloudera Manager.

Users accounts in the Cloudera Manager database page show Cloudera Manager in the User Type column. User accounts in an LDAP directory or other external authentication mechanism show External in the User Type column.

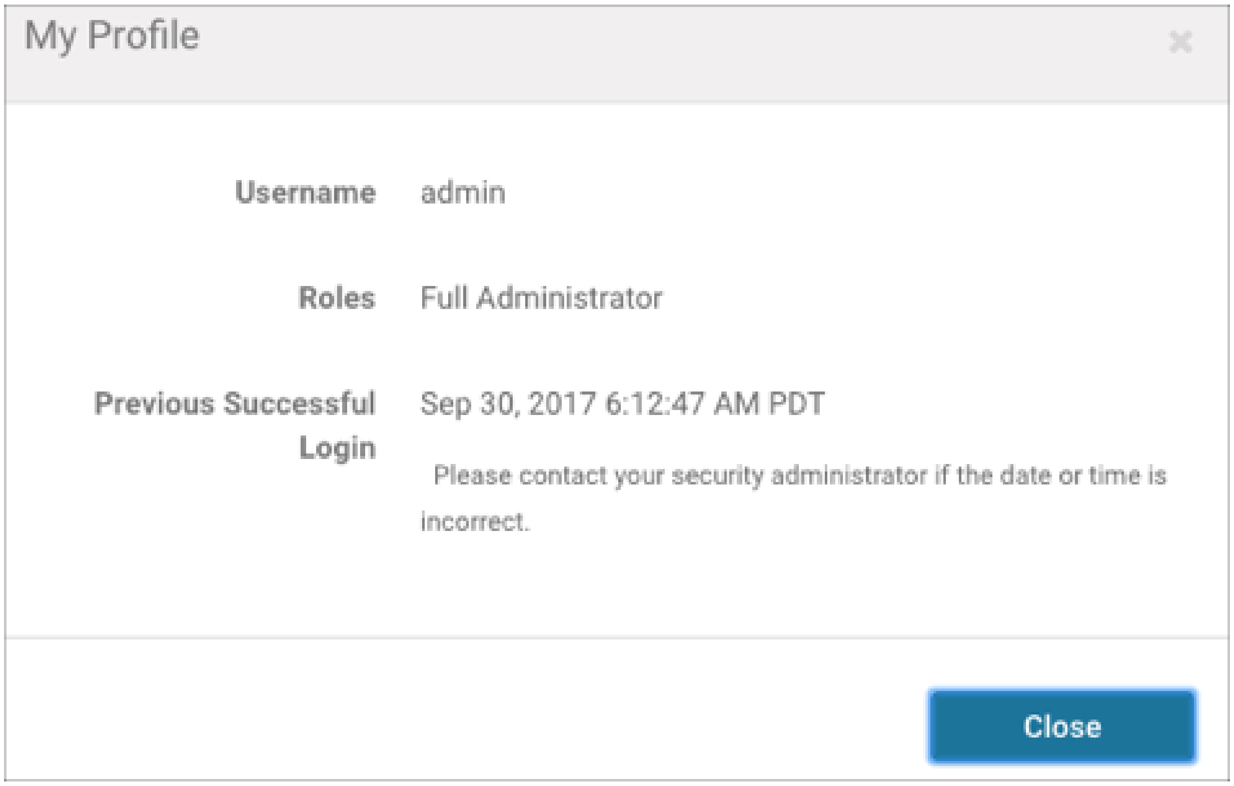
User Roles

User accounts include the user's role, which determines the Cloudera Manager features visible to the user, the actions the user can perform, and, for certain roles, what clusters actions can be performed on. All tasks in the Cloudera Manager documentation indicate which role is required to perform the task.

Determining the Role of the Currently Logged in User

1. Click the logged-in username at the far right of the top navigation bar.

2. Select My Profile. The role displays. For example:



Username admin

Roles Full Administrator

Previous Successful Login Sep 30, 2017 6:12:47 AM PDT

Please contact your security administrator if the date or time is incorrect.

Changing the Logged-In Internal User Password

1. Click the logged-in username at the far right of the top navigation bar and select Change Password.
2. Enter the current password and a new password twice, and then click OK.

Adding an Internal User Account

1. Select AdministrationUsers.
2. Click the Add User button.
3. Enter a username and password.
4. In the Role drop-down menu, select a role for the new user.
5. Click Add.

Assigning User Roles

1. Select AdministrationUsers.
2. Check the checkbox next to one or more usernames.
3. Select Actions for SelectedAssign User Roles.
4. In the drop-down menu, select the role.
5. Click the Assign Role button.

Changing an Internal User Account Password

1. Select AdministrationUsers.
2. Click the Change Password button next to a username with User Type Cloudera Manager.
3. Type the new password and repeat it to confirm.
4. Click the Update button to make the change.

Deleting Internal User Accounts

1. Select AdministrationUsers.
2. Check the checkbox next to one or more usernames with User Type Cloudera Manager.
3. Select Actions for SelectedDelete.
4. Click the OK button. (There is no confirmation of the action.)

Viewing User Sessions

1. Select AdministrationUsers.
2. Click the tab User Sessions.

Configuring external authentication and authorization for Cloudera Manager

Cloudera Manager supports user authentication against an internal database and against an external service. The following sections describe how to configure the supported external services.

Configure authentication using Active Directory

How to configure authentication using Active Directory in Cloudera Manager.

Procedure

1. Log in to the Cloudera Manager Admin Console.
2. Select Administration Settings
3. Select External Authentication in the Category filter.
4. For Authentication Backend Order, select the order in which Cloudera Manager should look up authentication credentials for login attempts.
5. For External Authentication Type, select Active Directory.
6. In the LDAP URL property, enter the URL of the Active Directory server.
7. In the Active Directory Domain property, provide the domain to authenticate against.

   LDAP URL and Active Directory are the only settings required to allow anyone in Active Directory to log in to Cloudera Manager.

   For example, if you set LDAP URL to ldap://adserver.example.com and the Active Directory Domain to ADRE ALM.EXAMPLE.COM, users can log into Cloudera Manager using just their username, such as sampleuser. They no longer require the complete string: sampleuser@ADREALM.EXAMPLE.COM.
8. Restart the Cloudera Manager Server.

Configure authentication using an LDAP-compliant identity service

How to configure authentication using LDAP in Cloudera Manager.

About this task

An LDAP-compliant identity/directory service, such as OpenLDAP, provides different options for enabling Cloudera Manager to look-up user accounts and groups in the directory:

- Use a single Distinguished Name (DN) as a base and provide a pattern (Distinguished Name Pattern) for matching user names in the directory, or
- Search filter options let you search for a particular user based on somewhat broader search criteria – for example Cloudera Manager users could be members of different groups or organizational units (OUs), so a single pattern does not find all those users. Search filter options also let you find all the groups to which a user belongs, to help determine if that user should have login or admin access.

Procedure

1. Log in to Cloudera Manager Admin Console.
2. Select Administration Settings .
3. Select External Authentication for the Category filter to display the settings.
4. For Authentication Backend Order, select the order in which Cloudera Manager should look up authentication credentials for login attempts.
5. For External Authentication Type, select LDAP.
6. In the LDAP URL property, provide the URL of the LDAP server and (optionally) the base Distinguished Name (DN) (the search base) as part of the URL — for example ldap://ldap-server.corp.com/dc=corp,dc=com.
7. If your server does not allow anonymous binding, provide the user DN and password to be used to bind to the directory. These are the LDAP Bind User Distinguished Name and LDAP Bind Password properties. By default, Cloudera Manager assumes anonymous binding.
8. Search for users and groups. You can search using User or Group search filters, using the LDAP User Search Base, LDAP User Search Filter, LDAP Group Search Base and LDAP Group Search Filter settings. These allow you to combine a base DN with a search filter to allow a greater range of search targets.

   For example, if you want to authenticate users who may be in one of multiple OUs, the search filter mechanism will allow this. You can specify the User Search Base DN as dc=corp,dc=com and the user search filter as uid={0}. Then Cloudera Manager will search for the user anywhere in the tree starting from the Base DN. Suppose you have two OUs—ou=Engineering and ou=Operations—Cloudera Manager will find User "foo" if it exists in either of these OUs, that is, uid=foo,ou=Engineering,dc=corp,dc=com or uid=foo,ou=Operations,dc=corp,dc=com.

   You can use a user search filter along with a DN pattern, so that the search filter provides a fallback if the DN pattern search fails.

   The Groups filters let you search to determine if a DN or username is a member of a target group. In this case, the filter you provide can be something like member={0} where {0} will be replaced with the DN of the user you are authenticating. For a filter requiring the username, {1} may be used, as memberUid={1}. This will return a list of groups the user belongs to, which will be compared to the list in the group properties discussed in
9. Restart the Cloudera Manager Server.

Configuring Cloudera Manager to Use LDAPS

About this task

If the LDAP server certificate has been signed by a trusted Certificate Authority, steps 1 and 2 below may not be necessary.

Procedure

1. Copy the CA certificate file to the Cloudera Manager Server host.
2. Import the CA certificate(s) from the CA certificate file to the local truststore. The default truststore is located in the $JAVA_HOME/jre/lib/security/cacerts file.

   This contains the default CA information shipped with the JDK. Create an alternate default file called jssecacerts in the same location as the cacerts file. You can now safely append CA certificates for any private or public CAs not present in the default cacerts file, while keeping the original file intact.

   For our example, we will follow this recommendation by copying the default cacerts file into the new jssecacerts file, and then importing the CA certificate to this alternate truststore.

   ```
   cp $JAVA_HOME/jre/lib/security/cacerts $JAVA_HOME/jre/lib/security/jssec
   acerts
   ```

   ```
   $ /usr/java/latest/bin/keytool -import -alias nt_domain_name
   ```

```
-keystore /usr/java/latest/jre/lib/security/jssecacerts -f
ile path_to_CA_cert
```

Note: The default password for the cacerts store is changeit. The -alias does not always need to be the domain name.

Alternatively, you can use the Java options: javax.net.ssl.trustStore and javax.net.ssl.trustStorePassword. Open the /etc/default/cloudera-scm-server file and add the following options:

```
export CMF_JAVA_OPTS="-Xmx2G -XX:MaxPermSize=256m -XX:+HeapDumpOnOutOfMe
moryError -XX:HeapDumpPath=/tmp -Djavax.net.ssl.trustStore=/usr/java/def
ault/jre/lib/security/jssecacerts -Djavax.net.ssl.trustStorePassword=cha
ngeit"
```

3. Configure the LDAP URL property to use ldaps://*ldap_server* instead of ldap://*ldap_server*
4. Restart the Cloudera Manager Server.

Configure authentication using Kerberos (SPNEGO)

Cloudera Manager 6.3 and higher support Kerberos authentication using SPNEGO for the Admin Console as well as the API.

Before you begin

Important:

With SPNEGO enabled, the Swagger-based Java and Python SDKs, as well as the older deprecated Java SDK, can still authenticate using HTTP Basic Authentication. The older deprecated Python SDK cannot. Do not enable SPNEGO if you are relying on the deprecated Python client for any operations.

Procedure

1. If you have not already done so, enable Kerberos for cluster services.
2. Navigate to Administration > Settings.
3. Enter SPNEGO in the Search field.
4. Select the Enable SPNEGO/Kerberos Authentication for the Admin Console and API checkbox. Leave the other SPNEGO settings blank to allow Cloudera Manager to automatically generate the principal and keytab.
5. Click Save Changes.
6. Restart Cloudera Manager Server.

Configure authentication using an external program

How to configure authentication using an external program in Cloudera Manager.

About this task

Cloudera Manager can use a custom external authentication program. Typically, this may be a custom script that interacts with a custom authentication service. Cloudera Manager will call the external program with the username as the first command line argument. The password is passed over stdin. Each positive value exit code can be mapped to Cloudera Manager user roles. A negative value is returned for failure to authenticate. Valid values for the exit code are between 0 and 127.

Procedure

1. Log in to Cloudera Manager Admin Console.
2. Select Administration Settings
3. Select External Authentication for the Category filter to display the settings.
4. For External Authentication Type, select External Program.

5. Provide a path to the external program in the External Authentication Program Path property.

What to do next

After you configure authentication for Cloudera Manager, configure authorization for the authenticated users. This is done by mapping the authenticated users to Cloudera Manager user roles.

Configure authentication using SAML

How to configure authentication using SAML in Cloudera Manager..

Before you begin

About this task

Cloudera Manager supports the Security Assertion Markup Language (SAML), an XML-based open standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider (IDP) and a service provider (SP). The SAML specification defines three roles: the principal (typically a user), the IDP, and the SP. In the use case addressed by SAML, the principal (user agent) requests a service from the service provider. The service provider requests and obtains an identity assertion from the IDP. On the basis of this assertion, the SP can make an access control decision—in other words it can decide whether to perform some service for the connected principal.

The primary SAML use case is called web browser single sign-on (SSO). A user wielding a user agent (usually a web browser) requests a web resource protected by a SAML SP. The SP, wanting to know the identity of the requesting user, issues an authentication request to a SAML IDP through the user agent. In the context of this terminology, Cloudera Manager operates as a SP. This topic discusses the Cloudera Manager part of the configuration process; it assumes that you are familiar with SAML and SAML configuration in a general sense, and that you have a functioning IDP already deployed.

Note:

- Cloudera Manager supports both SP- and IDP-initiated SSO.
- The logout action in Cloudera Manager will send a single-logout request to the IDP.
- SAML authentication has been tested with specific configurations of SiteMinder and Shibboleth. While SAML is a standard, there is a great deal of variability in configuration between different IDP products, so it is possible that other IDP implementations, or other configurations of SiteMinder and Shibboleth, may not interoperate with Cloudera Manager.
- To bypass SSO if SAML configuration is incorrect or not working, you can login using a Cloudera Manager local account using the URL: http://*cm_host*:7180/cmf/localLogin

Prepare Files

About this task

You will need to prepare the following files and information, and provide these to Cloudera Manager:

- A Java keystore containing a private key for Cloudera Manager to use to sign/encrypt SAML messages.
- The SAML metadata XML file from your IDP. This file must contain the public certificates needed to verify the sign/encrypt key used by your IDP per the SAML Metadata Interoperability Profile. For example, if you are using the Shibboleth IdP, the metadata file is available at: https://<IdPHOST>:8080/idp/shibboleth.

  Note: For guidance on how to obtain the metadata XML file from your IDP, either contact your IDP administrator or consult the documentation for the version of the IDP you are using.

- The entity ID that should be used to identify the Cloudera Manager instance
- How the user ID is passed in the SAML authentication response:
  - As an attribute. If so, what identifier is used.
  - As the NameID.

- The method by which the Cloudera Manager role will be established:
    - From an attribute in the authentication response:
        - What identifier will be used for the attribute
        - What values will be passed to indicate each role
    - From an external script that will be called for each use:
        - The script takes user ID as $1
        - The script sets an exit code to reflect successful authentication. Valid values for the exit codes are between 0 and 127. These values are used in Cloudera Manager to map authenticated users to user roles within Cloudera Manager.

Configure Cloudera Manager

Procedure

1. Log in to Cloudera Manager Admin Console.
2. Select Administration Settings .
3. Select External Authentication for the Category filter to display the settings.
4. Set the External Authentication Type property to SAML (the Authentication Backend Order property is ignored for SAML).
5. Set the Path to SAML IDP Metadata File property to point to the IDP metadata file.
6. Set the Path to SAML Keystore File property to point to the Java keystore prepared earlier.
7. In the SAML Keystore Password property, set the keystore password.
8. In the Alias of SAML Sign/Encrypt Private Key property, set the alias used to identify the private key for Cloudera Manager to use.
9. In the SAML Sign/Encrypt Private Key Password property, set the private key password.
10. Set the SAML Entity ID property if:
    - There is more than one Cloudera Manager instance being used with the same IDP (each instance needs a different entity ID).
    - Entity IDs are assigned by organizational policy.
11. In the Source of User ID in SAML Response property, set whether the user ID will be obtained from an attribute or the NameID.

    If an attribute will be used, set the attribute name in the SAML attribute identifier for user ID property. The default value is the normal OID used for user IDs and so may not need to be changed.
12. In the SAML Role assignment mechanism property, set whether the role assignment will be done from an attribute or an external script.
    - If an attribute will be used:
        - In the SAML attribute identifier for user role property, set the attribute name if necessary. The default value is the normal OID used for OrganizationalUnits and so may not need to be changed.
    - If an external script will be used, set the path to that script in the Path to SAML Role Assignment Script property. Make sure that the script is executable (an executable binary is fine - it does not need to be a shell script).
13. Save the changes. Cloudera Manager will run a set of validations that ensure it can find the metadata XML and the keystore, and that the passwords are correct. If you see a validation error, correct the problem before proceeding.
14. Restart the Cloudera Manager Server.

What to do next

After you configure authentication for Cloudera Manager, configure authorization for the authenticated users. This is done by mapping the authenticated users to Cloudera Manager user roles.

Configuring the IDP

About this task

After the Cloudera Manager Server is restarted, it will attempt to redirect to the IDP login page instead of showing the normal CM page. This may or may not succeed, depending on how the IDP is configured. In either case, the IDP will need to be configured to recognize CM before authentication will actually succeed. The details of this process are specific to each IDP implementation - refer to your IDP documentation for details.

Procedure

1. Download the Cloudera Manager’s SAML metadata XML file from http://*hostname*:7180/saml/metadata.
2. Inspect the metadata file and ensure that any URLs contained in the file can be resolved by users’ web browsers. The IDP will redirect web browsers to these URLs at various points in the process. If the browser cannot resolve them, authentication will fail. If the URLs are incorrect, you can manually fix the XML file or set the Entity Base URL in the CM configuration to the right value, and then re-download the file.
3. Provide this metadata file to your IDP using whatever mechanism your IDP provides.
4. Ensure that the IDP has access to whatever public certificates are necessary to validate the private key that was provided to Cloudera Manager earlier.
5. Ensure that the IDP is configured to provide the User ID and Role using the attribute names that Cloudera Manager was configured to expect, if relevant.
6. Ensure the changes to the IDP configuration have taken effect (a restart may be necessary).

Verifying Authentication and Authorization

Procedure

1. Return to the Cloudera Manager Admin Console and refresh the login page.
2. Attempt to log in with credentials for a user that is entitled. The authentication should complete and you should see the Home Status tab.
3. If authentication fails, you will see an IDP provided error message. Cloudera Manager is not involved in this part of the process, and you must ensure the IDP is working correctly to complete the authentication.
4. If authentication succeeds but the user is not authorized to use Cloudera Manager, they will be taken to an error page by Cloudera Manager that explains the situation. If an user who should be authorized sees this error, then you will need to verify their role configuration, and ensure that it is being properly communicated to Cloudera Manager, whether by attribute or external script. The Cloudera Manager log will provide details on failures to establish a user’s role. If any errors occur during role mapping, Cloudera Manager will assume the user is unauthorized.

Enabling Kerberos Authentication for CDP

How to use the Cloudera Manager Kerberos wizard to set up authentication.

Minimum Required Role: Cluster Administrator (also provided by Full Administrator) This feature is not available when using Cloudera Manager to manage Data Hub clusters.

Cloudera Manager provides a wizard for integrating your organization's Kerberos instance with your cluster to provide authentication services.

Kerberos must already be deployed in your organization and the Kerberos key distribution center (KDC) must be ready to use, with a realm established. If you are using Red Hat Identity Management/FreeIPA, all of your cluster hosts must already be joined to the IPA domain. For Hue and Oozie, the Kerberos realm must support renewable tickets.

Important: Before integrating Kerberos with your cluster, configure TLS encryption between Cloudera Manager Server and all Cloudera Manager Agent host systems in the cluster. During the Kerberos integration process, Cloudera Manager Server sends keytab files to the Cloudera Manager Agent hosts, and TLS encrypts the network communication so these files are protected.

Cloudera Manager clusters can be integrated with MIT Kerberos, Red Hat Identity Management (or the upstream FreeIPA), or Microsoft Active Directory.

For Active Directory, you must have administrative privileges to the Active Directory instance for initial setup and for on-going management, or you will need to have the help of your AD administrator prior to and during the integration process. For example, administrative access is needed to access the Active Directory KDC, create principals, and troubleshoot Kerberos TGT/TGS-ticket-renewal and take care of any other issues that may arise.

For Red Hat IdM, make sure that all cluster hosts are joined to the IPA domain. You can join a host to the domain by installing the freeipa-client package and then running the ipa-client-install script.

Kerberos client OS-specific packages must be installed on all cluster hosts and client hosts that will authenticate using Kerberos.

OS	Packages Required
RHEL 7 Compatible, RHEL 6 Compatible	• openldap-clients on the Cloudera Manager Server host • krb5-workstation, krb5-libs on ALL hosts • (Red Hat IdM/FreeIPA only) freeipa-client on all cluster hosts
SLES	• openldap2-client on the Cloudera Manager Server host • krb5-client on ALL hosts • (Red Hat IdM/FreeIPA only) freeipa-client on all cluster hosts
Ubuntu	• ldap-utils on the Cloudera Manager Server host • krb5-user on ALL hosts • (Red Hat IdM/FreeIPA only) freeipa-client on all cluster hosts
Windows	• krb5-workstation, krb5-libs on ALL hosts

Cloudera supports the Kerberos version that ships with each supported operating system listed in “Operating System Requirements”.

Related Information

Operating System Requirements

Step 1: Install Cloudera Manager and CDP

Cloudera strongly recommends that you install and configure the Cloudera Manager Server and Cloudera Manager Agents and CDP to set up a fully-functional CDP cluster before trying to configure Kerberos authentication for the cluster.

Required user:group Settings for Security

About this task

When you install the CDP packages and the Cloudera Manager Agents on your cluster hosts, Cloudera Manager takes some steps to provide system security such as creating the OS user:group accounts and setting directory permissions as shown in the table below. These user accounts and directory permissions work with the Hadoop Kerberos security requirements.

This User	Runs These Roles
hdfs	NameNode, DataNodes, and Secondary NameNode
mapred	JobTracker and TaskTrackers (MR1) and Job History Server (YARN)
yarn	ResourceManager and NodeManagers (YARN)

This User	Runs These Roles
oozie	Oozie Server
hue	Hue Server, Beeswax Server, Authorization Manager, and Job Designer

The hdfs user has HDFS superuser privileges.

When you install the Cloudera Manager Server on the server host, a new Unix user account called cloudera-scm is created automatically to support security. The Cloudera Manager Server uses this account to create host principals and deploy the keytabs on your cluster.

Depending on whether you installed CDP and Cloudera Manager at the same time or not, use one of the following sections for information on configuring directory ownerships on cluster hosts.

New Installation, Cloudera Manager and CDP Together

Installing a new Cloudera Manager cluster with CDP components at the same time can save you some of the user:group configuration required if you install them separately. The installation process creates the necessary user accounts on the Linux host system for the service daemons. At the end of the installation process when each cluster node starts up, the Cloudera Manager Agent process on the host automatically configures the directory ownership as shown in the table below, and the Hadoop daemons can then automatically set permissions for their respective directories. Do not change the directory owners on the cluster. They must be configured exactly as shown below.

Step 2: Install JCE policy files for AES-256 encryption

How to install JCE policy files, if required.

Before you begin

Note: This step is not required when using JDK 1.8.0_161 or greater. JDK 1.8.0_161 enables unlimited strength encryption by default.

By default, CentOS and Red Hat Enterprise Linux 5.5 (and higher) use AES-256 encryption for Kerberos tickets, so the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files must be installed on all cluster hosts as detailed below. Alternatively, the Kerberos instance can be modified to not use AES-256.

To manually install the JCE policy files on the host system:

1. Download the jce_policy-<*version*>.zip file from the Oracle website.
2. Unzip the file.
3. Follow the steps in README.txt to install it.

To use Cloudera Manager to install the JCE policy file:

1. Log in to the Cloudera Manager Admin Console.
2. Navigate to AdministrationSecurity.
3. Click Install Java Unlimited Strength Encryption Policy Files.
4. Specify the repository location for your version of Cloudera Manager, and click Continue.
5. Check the box labeled Install Oracle Java SE Development Kit (JDK 8), and then check the box labeled Install Java Unlimited Strength Encryption Policy Files. Click Continue.
6. Provide cluster host login credentials, and then click Continue.
7. After the installation completes, click Finish. If the Upgrade Cloudera Manager page displays, you can exit by clicking the Cloudera Manager logo to return to the home page.

Alternative: Disable AES-256 encryption from the Kerberos instance:

1. Remove any aes256 entries from the supported_enctypes field of the kdc.conf or krb5.conf files.
2. Restart the Kerberos KDC and the kadmin server so the changes take effect.

The keys of relevant principals, such as Ticket Granting Ticket principal (krbtgt/REALM@REALM), might need to change.

Note: If AES-256 remains in use despite disabling it, it may be because the aes256-cts:normal setting existed when the Kerberos database was created. To resolve this issue, create a new Kerberos database and then restart both the KDC and the kadmin server.

To verify the type of encryption used in your cluster:

Procedure

1. For MIT KDC: On the local KDC host, type this command in the kadmin.local or kadmin shell to create a test principal:

   ```
   kadmin:  addprinc test
   ```

   For Active Directory: Create a new AD account with the name, test.
2. On a cluster host, type this command to start a Kerberos session as test:

   ```
   kinit test
   ```
3. On a cluster host, type this command to view the encryption type in use:

   ```
   klist -e
   ```

   If AES is being used, output like the following is displayed after you type the klist command (note that AES-256 is included in the output):

   ```
   Ticket cache: FILE:/tmp/krb5cc_0
   Default principal: test@Cloudera Manager
   Valid starting     Expires            Service principal
   05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/Cloudera Manager@Cloudera Man
   ager
       Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256
    CTS mode with 96-bit SHA-1 HMAC
   ```

Step 3: Create the Kerberos Principal for Cloudera Manager Server

How to create a Kerberos principal for Cloudera Manager Server.

About this task

At the end of the integration process using the configuration wizard, Cloudera Manager Server will create host principals and deploy keytabs for all services configured on the cluster, which means that Cloudera Manager Server requires a principal that has privileges to create these other accounts.

Note: This task requires administrator privileges on the Kerberos instance. If you do not have administrator privileges, ask your Kerberos administrator to setup the principal and password for you. You will need to enter the principal name and password into the wizard later in the install process.

If an administrator principal to act on behalf of Cloudera Manager cannot be created on the Kerberos KDC for whatever reason, Cloudera Manager cannot create or manage principals and keytabs for CDP services. That means these principals must be created manually on the Kerberos KDC and then imported (retrieved) by Cloudera Manager. See “Using a custom Kerberos keytab retrieval script” for details about this process.

Note: The steps below summarize the process of adding a principal specifically for Cloudera Manager Server to an MIT KDC and an Active Directory KDC. See documentation from MIT, Microsoft, or the appropriate vendor for more detailed information.

Related Information

Using a custom Kerberos keytab retrieval script

Creating a Principal in Active Directory

About this task

Check the Microsoft documentation for specific details for your Active Directory KDC. The general process is as follows:

Procedure

1. Create an Organizational Unit (OU) in your Active Directory KDC service that will contain the principals for use by the CDP cluster.
2. Add a new user account to Active Directory, for example, *username*@EXAMPLE.COM. Set the password for the user to never expire.
3. Use the Delegate Control wizard in Active Directory to grant this new user permission to Create, Delete, and Manage User Accounts in the OU created in step 1. Make sure that these permissions are only applied to that specific OU, and nowhere else.

Creating a Principal in an MIT KDC

About this task

For MIT Kerberos, administrator principals are defined in the /var/kerberos/krb5kdc/kadm5.acl file on the KDC host. The default entry is:

```
*/admin@EXAMPLE.COM     *
```

In this example, principals that include the instance name admin designate a user account as an administrator, such as *jdoe*/admin@*EXAMPLE.COM*.

If you modify the kadm5.acl file, such as replacing EXAMPLE.COM with your realm name, make sure to restart the kadmin service:

- RHEL 7 Compatible:

  ```
  systemctl restart kadmin
  ```
- All Others:

  ```
  service kadmin restart
  ```

Create the Cloudera Manager Server administrator principal as shown below, using the admin instance name and your realm name. If your kadm5.acl file defines a different pattern for administrators, make sure that the principal you create matches that pattern.

For MIT Kerberos KDC on a remote host:

Note: To connect to a remote KDC using the kadmin command, your currently authenticated Kerberos principal must be an existing Kerberos administrator. If you do not have an existing admin principal, you must run kadmin.local as described below.

```
kadmin
kadmin: addprinc -pw password cloudera-scm/admin@EXAMPLE.COM
```

For MIT Kerberos KDC on the local host:

Note: The kadmin.local command must be run as root on the same host as the KDC.

```
kadmin.local
```

```
kadmin.local: addprinc -pw password cloudera-scm/admin@EXAMPLE.COM
```

Creating a Principal in Red Hat Identity Management (FreeIPA)

About this task

If you are using Red Hat IdM or FreeIPA, the wizard creates its own user account with the necessary privileges. To do so, it requires admin credentials the first time. These admin credentials are not stored, and are used only to create a new user and role (named cmadin-<*random_id*> and cmadminrole, respectively) and retrieve its keytab. Cloudera Manager stores this keytab for future Kerberos operations, such as regenerating the credentials of the CDP service accounts.

Because the wizard creates the principal it requires, you do not need to manually create an admin principal for Cloudera Manager at this time.

For additional information, refer to the Red Hat documentation:

- RHEL 7: “Linux Domain Identity, Authentication, and Policy Guide”
- RHEL 6: “Identity Management Guide”

Step 4: Enable Kerberos using the wizard

How to enable Kerberos using the Cloudera Manager wizard.

Related Information

Step 3: Create the Kerberos Principal for Cloudera Manager Server

Customizing Kerberos principals

1. Getting Started

Before you begin

- Set up a working KDC in Step 1 of the wizard as described below. Cloudera Manager supports authentication with MIT KDC, Active Directory, and Red Hat Identity Management/FreeIPA.
- Configure the KDC to allow renewable tickets with non-zero ticket lifetimes.

  Active Directory allows renewable tickets with non-zero lifetimes by default. You can verify this by checking Domain Security Settings Account Policies Kerberos Policy in Active Directory.

  For MIT KDC, make sure you have the following lines in the kdc.conf.

  ```
  max_life = 1d
  max_renewable_life = 7d
  ```
- If you are using Active Directory, make sure LDAP over TLS/SSL (LDAPS) is enabled for the Domain Controllers.
- Host names must be in lowercase. If you use uppercase letters in any host name, the cluster services will not start after enabling Kerberos.
- Create an account for Cloudera Manager that has the permissions to create other accounts in the KDC. This should have been completed as part of “Step 3: Create the Kerberos Principal for Cloudera Manager Server”.

Important:

If YARN Resource Manager HA has been enabled in a non-secure cluster, before enabling Kerberos you must clear the StateStore znode in ZooKeeper, as follows:

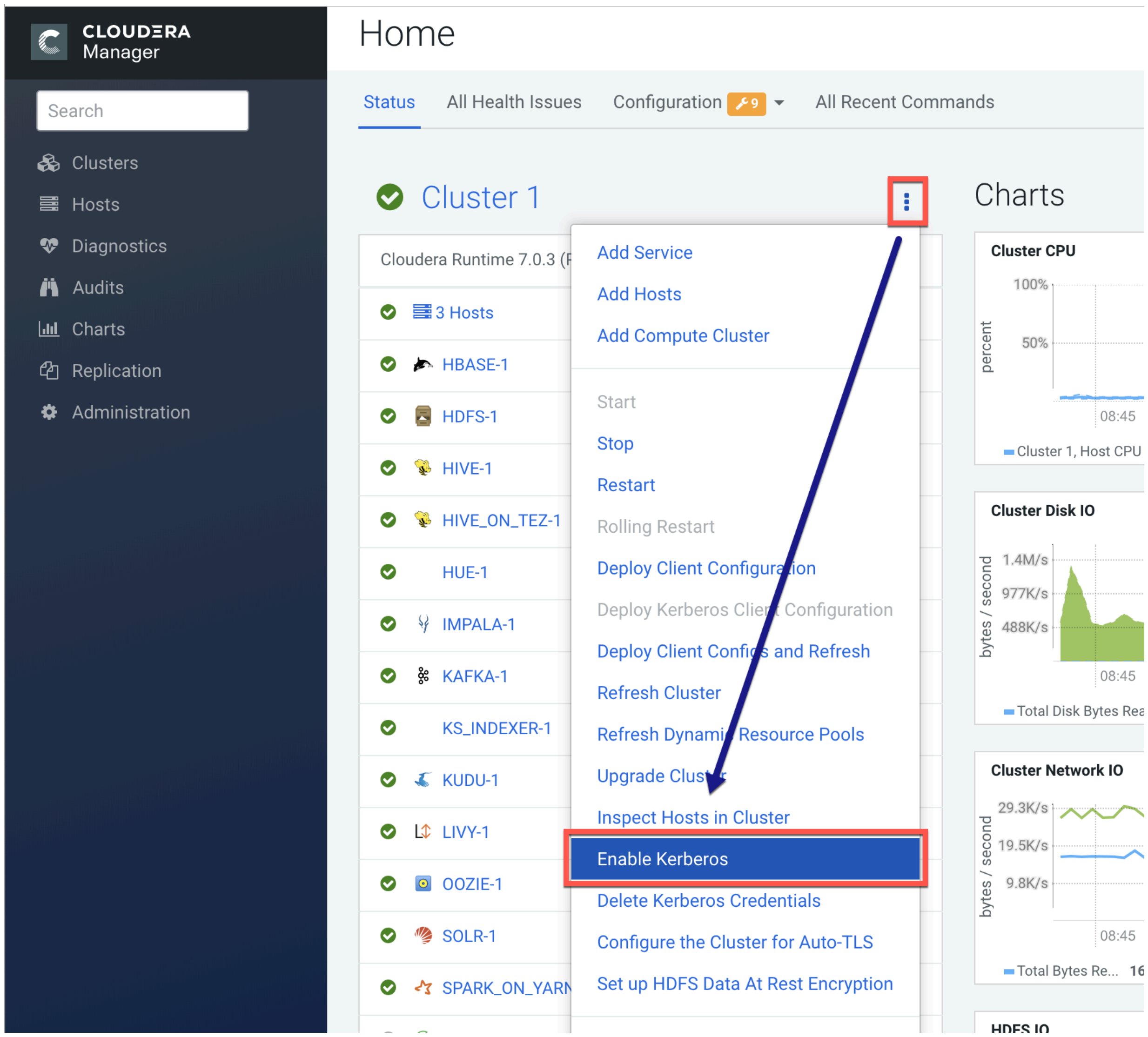
1. Go to the Cloudera Manager Admin Console home page, click to the right of the YARN service and select Stop.
2. When you see a Finished status, the service has stopped.
3. Go to the YARN service and select Actions Format State Store .
4. When the command completes, click Close.

Note:

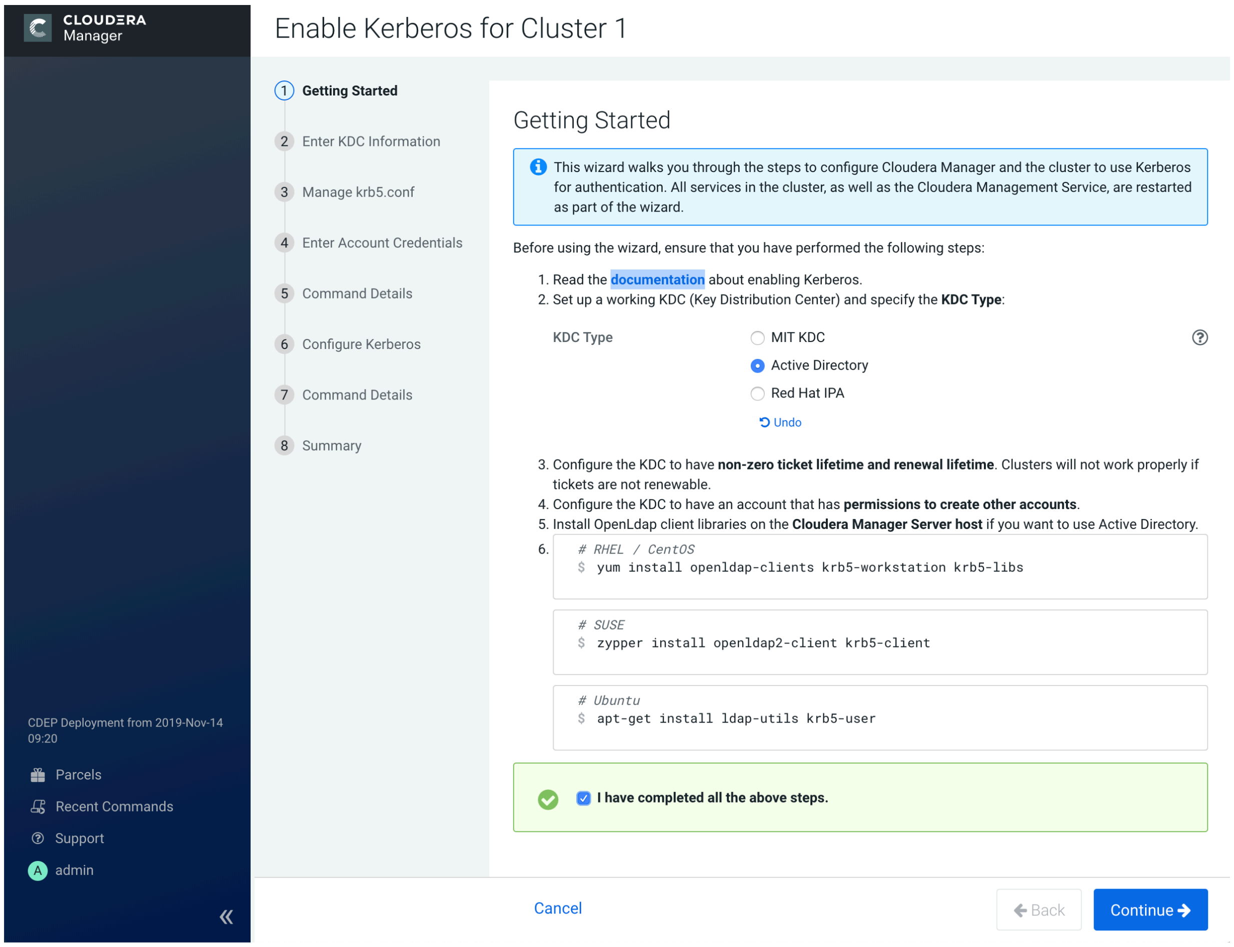
Minimum Required Role: Full Administrator. This feature is not available when using Cloudera Manager to manage Data Hub clusters.

Procedure

1. To start the Kerberos wizard, open the Cloudera Manager Admin Console, click the options menu for the applicable cluster, then click Enable Kerberos.



2. After you open the Kerberos wizard, a Getting Started page appears. Select the applicable KDC type to display configuration steps for your specific type of KDC. When you have completed all of the configuration steps, select the I have completed all the above steps check box, then click Continue.



Getting Started

This wizard walks you through the steps to configure Cloudera Manager and the cluster to use Kerberos for authentication. All services in the cluster, as well as the Cloudera Management Service, are restarted as part of the wizard.

Before using the wizard, ensure that you have performed the following steps:

1. Read the documentation about enabling Kerberos.
2. Set up a working KDC (Key Distribution Center) and specify the **KDC Type**:

KDC Type: MIT KDC [✓ Active Directory] Red Hat IPA

Undo

3. Configure the KDC to have **non-zero ticket lifetime and renewal lifetime**. Clusters will not work properly if tickets are not renewable.
4. Configure the KDC to have an account that has **permissions to create other accounts**.
5. Install OpenLdap client libraries on the **Cloudera Manager Server host** if you want to use Active Directory.
6.

```
# RHEL / CentOS
$ yum install openldap-clients krb5-workstation krb5-libs
```

```
# SUSE
$ zypper install openldap2-client krb5-client
```

```
# Ubuntu
$ apt-get install ldap-utils krb5-user
```

✓ **I have completed all the above steps.**

2. Enter KDC Information

About this task

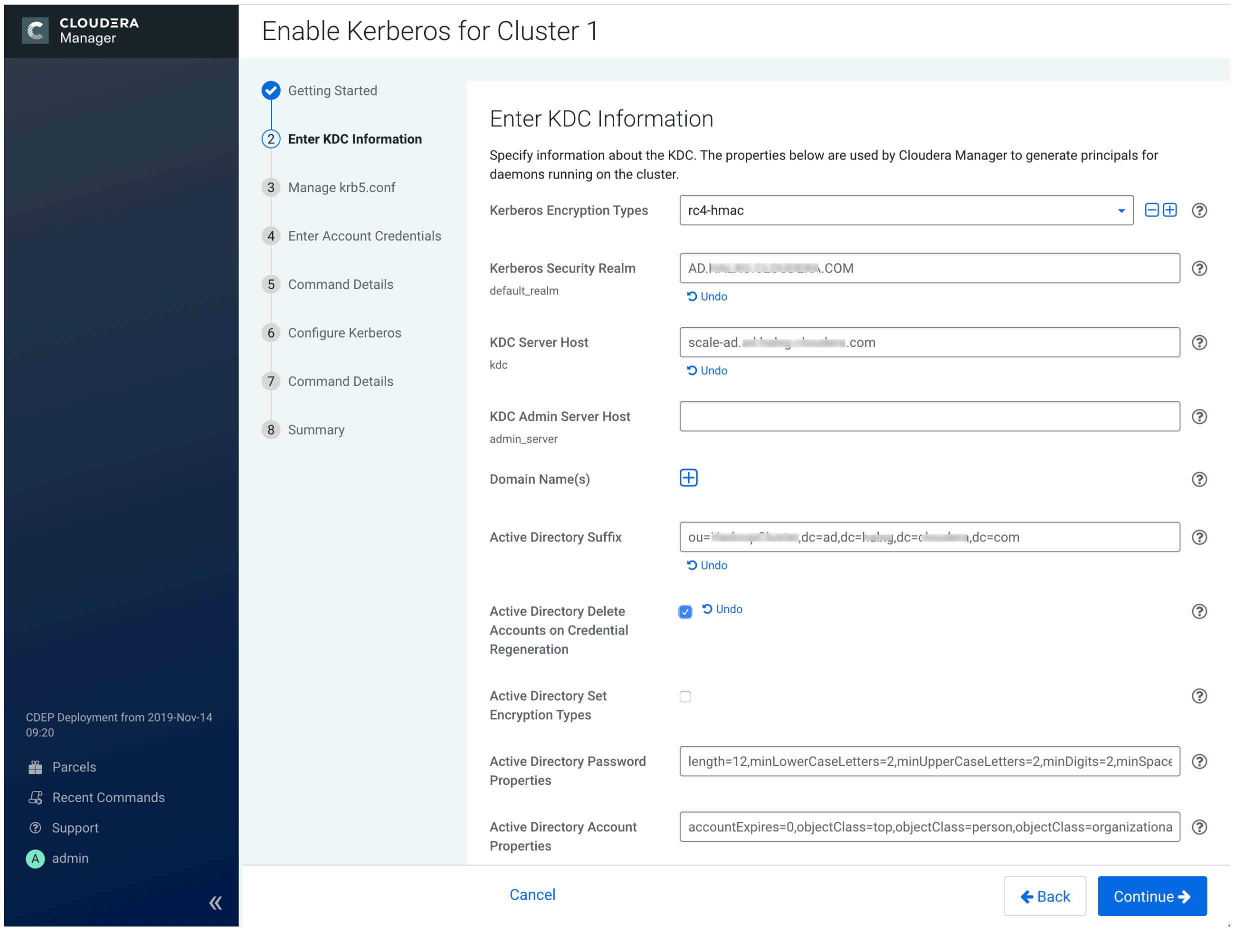
Enter configuration information for the KDC you are using.

Note:

- If you have multiple Domain Controllers (in case of AD) or MIT KDC servers, only enter the name of any one of them in the KDC Server Host field. Cloudera Manager will use that server only for creating accounts. Do not use a round-robin DNS for KDCs in the krb5.conf file. If you choose to use Cloudera Manager to manage the krb5.conf file, you can specify the additional Domain Controllers using the Advanced Configuration Snippet (Safety Valve) for the Default Realm in krb5.conf property as explained below. Fault tolerance and load balancing can be achieved by specifying more than one (usually 2-3) KDCs, using their FQDNs or IP addresses. This, along with a kdc_timeout parameter of 3-5 seconds, allows all Kerberos clients (including Java GSS-API) to attempt communication with the next KDC in the event of a KDC failure.
- Make sure the entries for the Kerberos Encryption Types field matches what your KDC supports.
- If you are using an Active Directory KDC, you can configure Active Directory account properties such as objectClass and accountExpires directly from the Cloudera Manager UI. You can also enable Cloudera Manager to delete existing AD accounts so that new ones can be created when Kerberos credentials are being regenerated.
- To use AES encryption, make sure you have deployed JCE Unlimited Strength Policy File, which is automatically included in Open JDK 1.8 232 (provided by Cloudera at the time of install) and up, but may not be available on some of the earlier JDK 1.8 releases.

Procedure

- In the Active Directory KDC example below, we entered values for the Kerberos Security Realm, the KDC Server Host, and the Active Directory Suffix, and also selected the Active Directory Delete Accounts on Credential Regeneration check box.



Enable Kerberos for Cluster 1

Enter KDC Information

Specify information about the KDC. The properties below are used by Cloudera Manager to generate principals for daemons running on the cluster.

Kerberos Encryption Types	rc4-hmac
Kerberos Security Realm (default_realm)	AD.[redacted].COM
KDC Server Host (kdc)	scale-ad.[redacted].com
KDC Admin Server Host (admin_server)	
Domain Name(s)	
Active Directory Suffix	ou=[redacted],dc=ad,dc=[redacted],dc=[redacted],dc=com
Active Directory Delete Accounts on Credential Regeneration	✓
Active Directory Set Encryption Types	☐
Active Directory Password Properties	length=12,minLowerCaseLetters=2,minUpperCaseLetters=2,minDigits=2,minSpace
Active Directory Account Properties	accountExpires=0,objectClass=top,objectClass=person,objectClass=organizationa

- Click Continue to proceed.

3. Manage krb5.conf

About this task

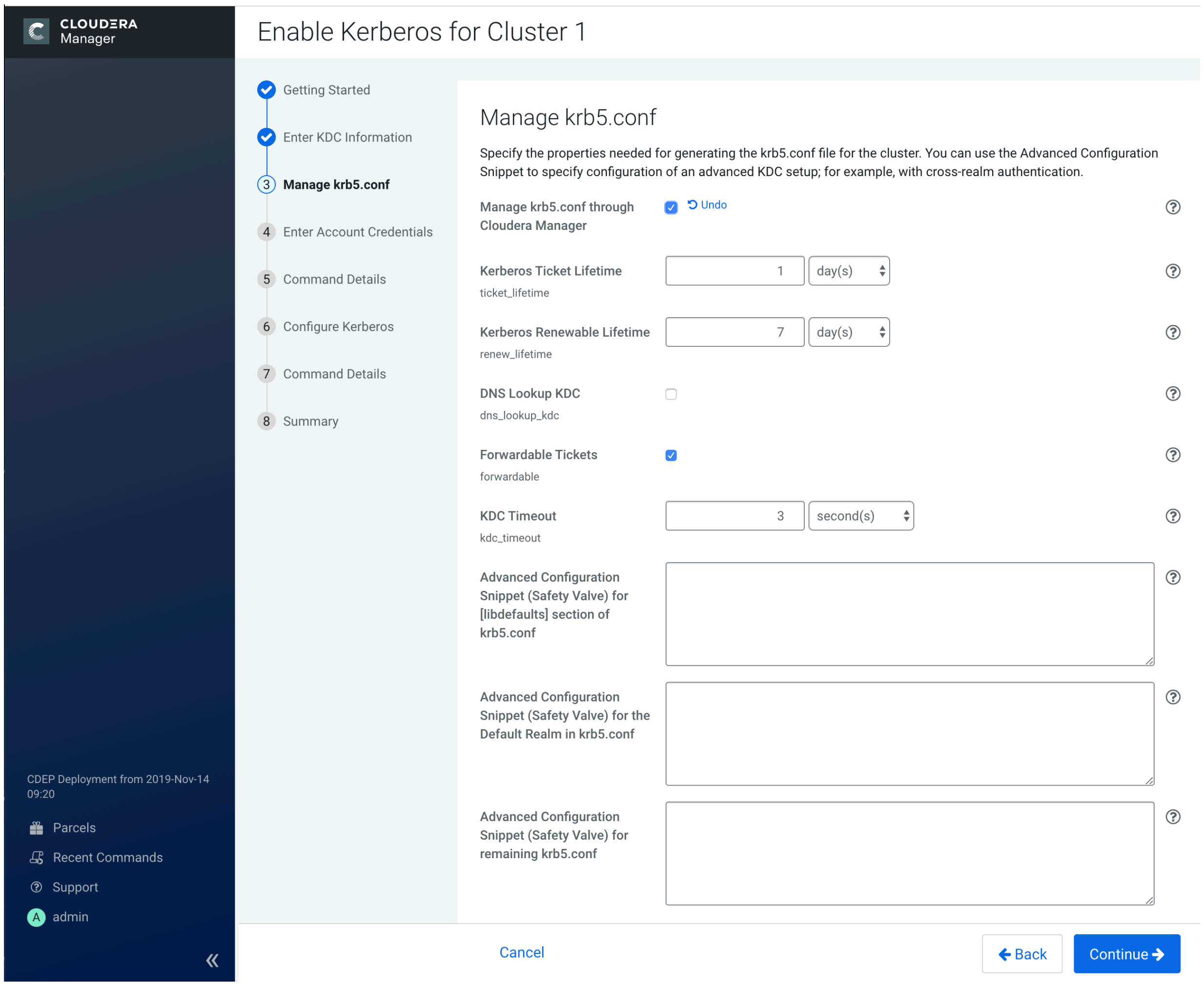
You can use this page to specify whether or not Cloudera Manager deploys and manages the krb5.conf file on your cluster.

Procedure

- If you select the Manage krb5.conf through Cloudera Manager check box, you can use this page to configure the krb5.conf file properties. In particular, the safety valves on this page can be used to configure cross-realm authentication.

  To configure multiple KDCs (as mentioned in the previous section), you can specify the additional Domain Controllers using the Advanced Configuration Snippet (Safety Valve) for the Default Realm in krb5.conf property box. For example:

```
kdc = kdc2.supportlab.cloudera.com
kdc = kdc3.supportlab.cloudera.com
kdc = kdcN.supportlab.cloudera.com
```



Manage krb5.conf

Specify the properties needed for generating the krb5.conf file for the cluster. You can use the Advanced Configuration Snippet to specify configuration of an advanced KDC setup; for example, with cross-realm authentication.

Manage krb5.conf through Cloudera Manager	✓
Kerberos Ticket Lifetime ticket_lifetime	1 day(s)
Kerberos Renewable Lifetime renew_lifetime	7 day(s)
DNS Lookup KDC dns_lookup_kdc	☐
Forwardable Tickets forwardable	✓
KDC Timeout kdc_timeout	3 second(s)
Advanced Configuration Snippet (Safety Valve) for [libdefaults] section of krb5.conf	
Advanced Configuration Snippet (Safety Valve) for the Default Realm in krb5.conf	
Advanced Configuration Snippet (Safety Valve) for remaining krb5.conf	

- If left unchecked, you must ensure that the krb5.conf is deployed on all hosts in the cluster, including the Cloudera Manager Server host.

  Note:

  If you are using Red Hat IdM/FreeIPA, by default the krb5.conf file contains a line similar to the following:

  ```
  default_ccache_name = KEYRING:persistent:%{uid}
  ```

  CDP does not support the keyring credential cache. Comment out this line on every cluster host by adding a hash mark (#) at the beginning, like this:

  ```
  #default_ccache_name = KEYRING:persistent:%{uid}
  ```

  If you configure Cloudera Manager to manage the krb5.conf file, you do not need to do anything.
- Click Continue to proceed.

4. Enter Account Credentials

About this task

Note: Enter the REALM portion of the principal in upper-case only to conform to Kerberos convention.

Note:

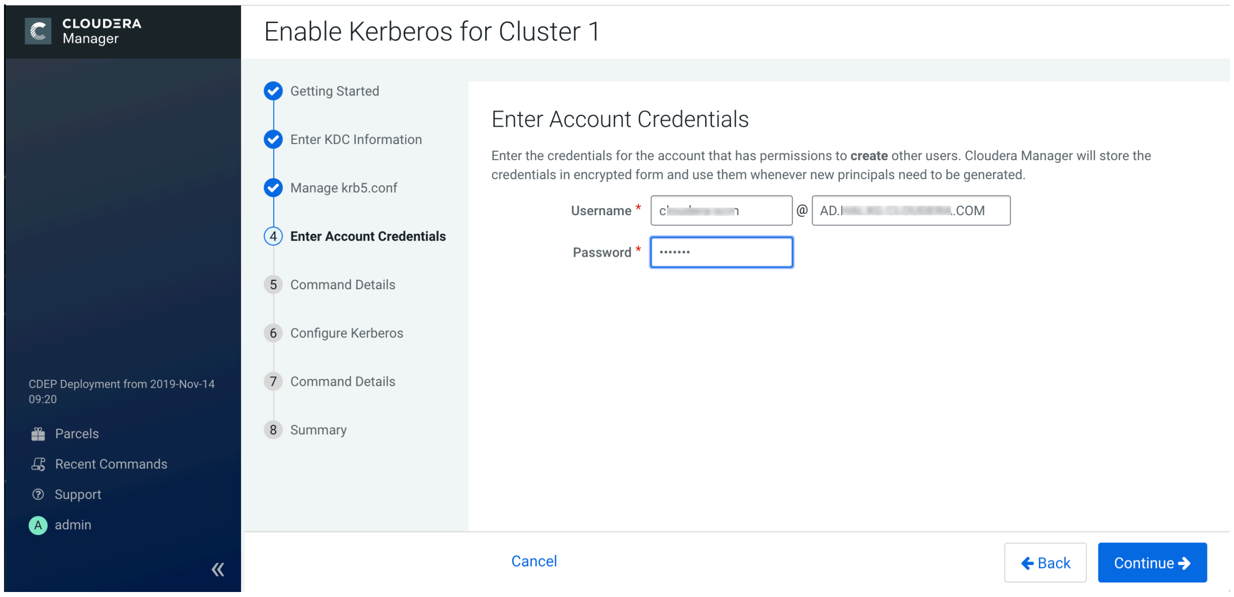
Many enterprises employ policies that require all passwords to be changed after a particular number of days. you can use the following steps to change the password in Cloudera Manager for the Account Manager:

1. In the Cloudera Manager Admin Console, select Administration Security .
2. Go to the Kerberos Credentials tab and click Import Kerberos Account Manager Credentials.
3. In the Import Kerberos Account Manager Credentials dialog box, enter the user name and password for the user that can create principals for CDP cluster in the KDC.

Procedure

1. Enter the user name and password for the user that can create principals for CDP cluster in the KDC. This is the user/principal you created in “Step 3: Create the Kerberos Principal for Cloudera Manager Server”. Cloudera Manager encrypts the user name and password into a keytab and uses it as needed to create new principals.

   If you are using Red Hat IdM/FreeIPA, enter the IPA admin credentials here. These admin credentials are not stored, and are used only to create a new user and role (named cmadin-<*random_id*> and cmadminrole, respectively) and retrieve its keytab. Cloudera Manager stores this keytab for future Kerberos operations, such as regenerating the credentials of the CDP service accounts.



Enter Account Credentials

Enter the credentials for the account that has permissions to **create** other users. Cloudera Manager will store the credentials in encrypted form and use them whenever new principals need to be generated.

Username * [redacted] @ AD.[redacted].COM

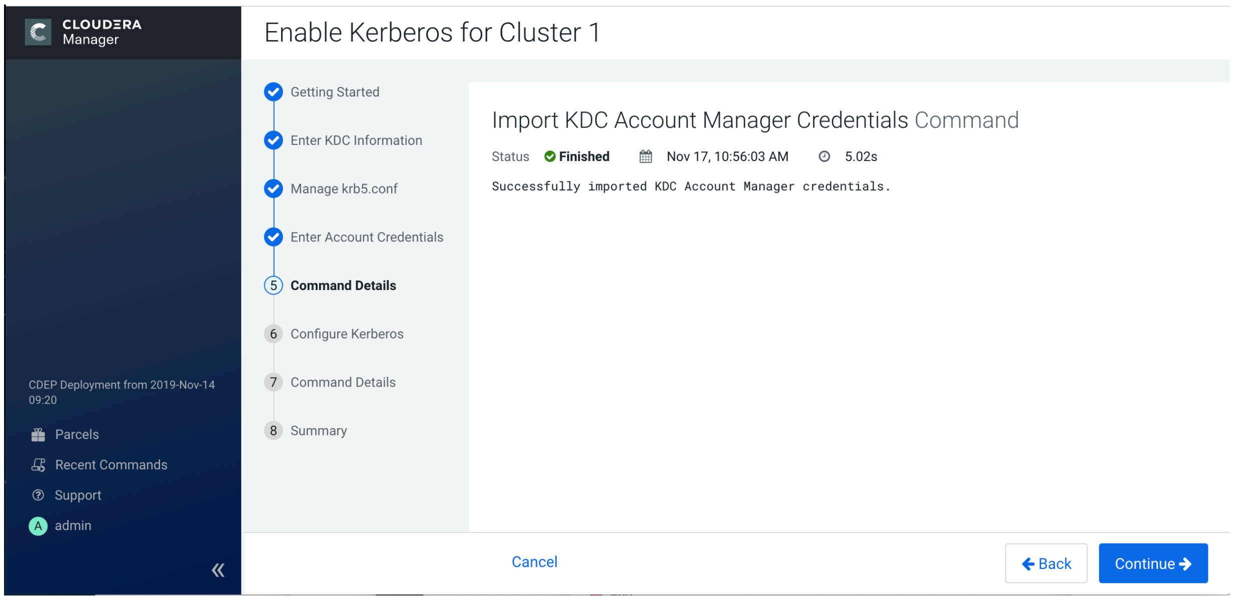
Password * •••••••

2. Click Continue to proceed.

5. Command Details

Procedure

- The Command Details page displays the outcome of the Enter Account Credentials step. Click Continue to proceed.



Import KDC Account Manager Credentials Command

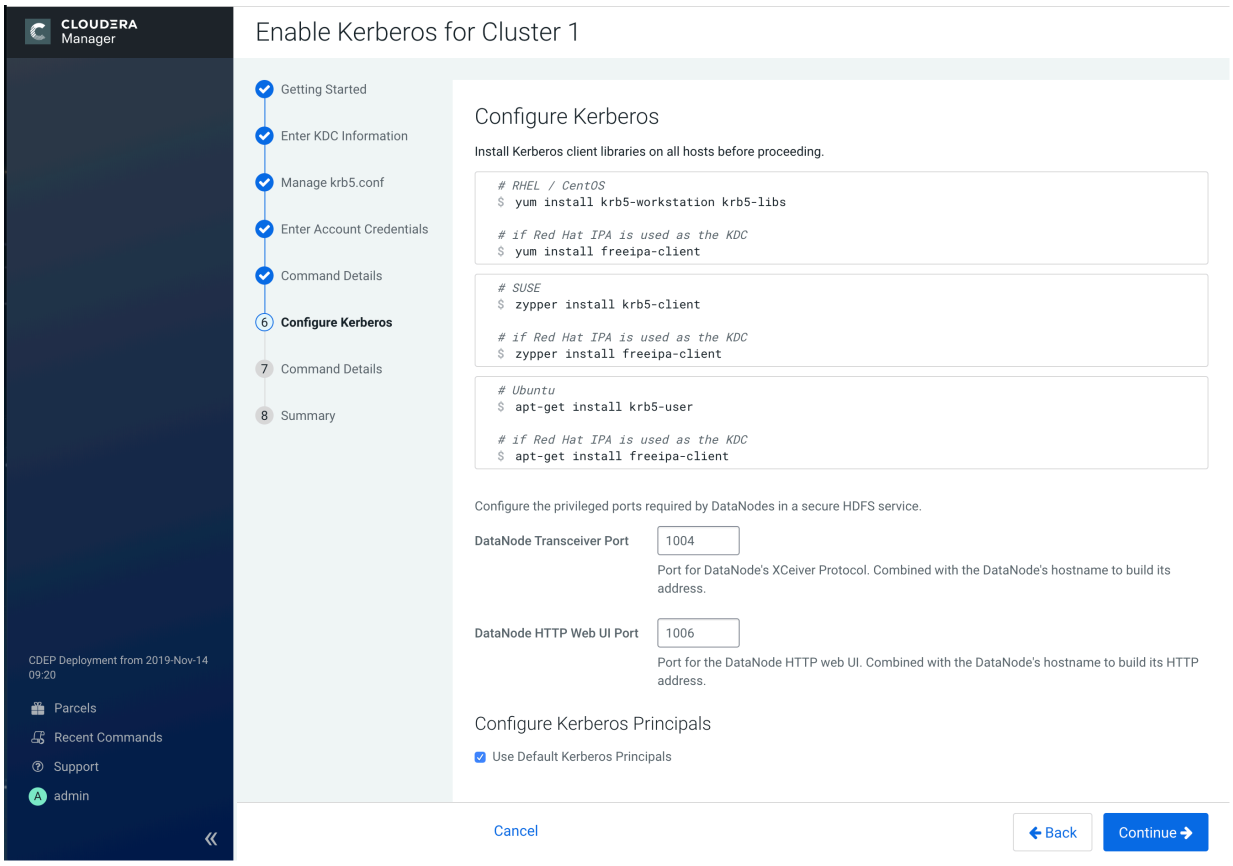
Status Finished Nov 17, 10:56:03 AM 5.02s

```
Successfully imported KDC Account Manager credentials.
```

6. Configure Kerberos

Procedure

1. If you have not already done so, run the provided commands on each cluster host to install the Kerberos client libraries.



Configure Kerberos

Install Kerberos client libraries on all hosts before proceeding.

```
# RHEL / CentOS
$ yum install krb5-workstation krb5-libs

# if Red Hat IPA is used as the KDC
$ yum install freeipa-client
```

```
# SUSE
$ zypper install krb5-client

# if Red Hat IPA is used as the KDC
$ zypper install freeipa-client
```

```
# Ubuntu
$ apt-get install krb5-user

# if Red Hat IPA is used as the KDC
$ apt-get install freeipa-client
```

Configure the privileged ports required by DataNodes in a secure HDFS service.

DataNode Transceiver Port: 1004

Port for DataNode's XCeiver Protocol. Combined with the DataNode's hostname to build its address.

DataNode HTTP Web UI Port: 1006

Port for the DataNode HTTP web UI. Combined with the DataNode's hostname to build its HTTP address.

Configure Kerberos Principals

[✓] Use Default Kerberos Principals

2. The wizard automatically sets the privileged ports needed by the DataNode Transceiver Protocol and the HTTP Web UI, but you can also specify alternate privileged ports.

   Note: If SSL is enabled, the assigned port numbers will be higher than 1024; if SSL is not enabled, the assigned port numbers will be less than 1024.

3. To configure custom service principals, clear the Use Default Kerberos Principals check box, and then specify a custom principal for each service.

   Note: Before configuring custom service principals, see “Customizing Kerberos principals” for additional information.

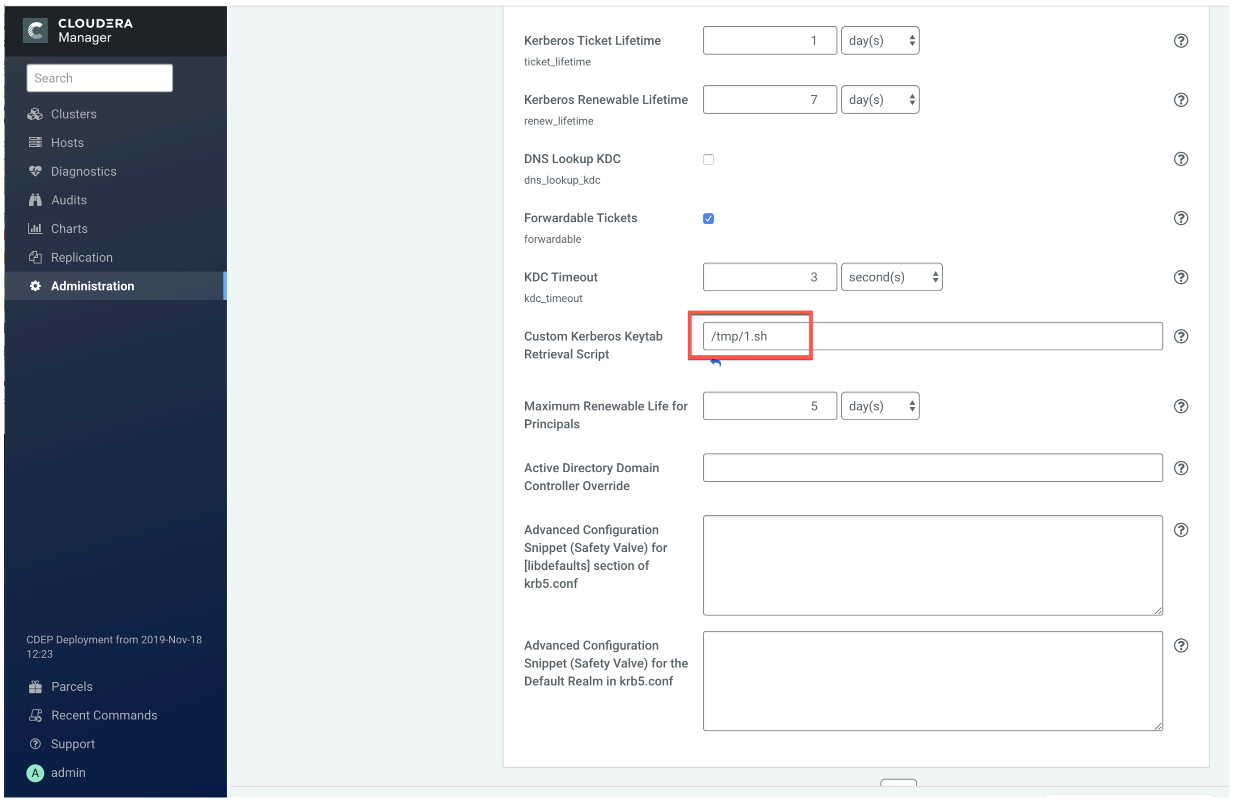
4. Click Continue to proceed.

7. Command Details

Procedure

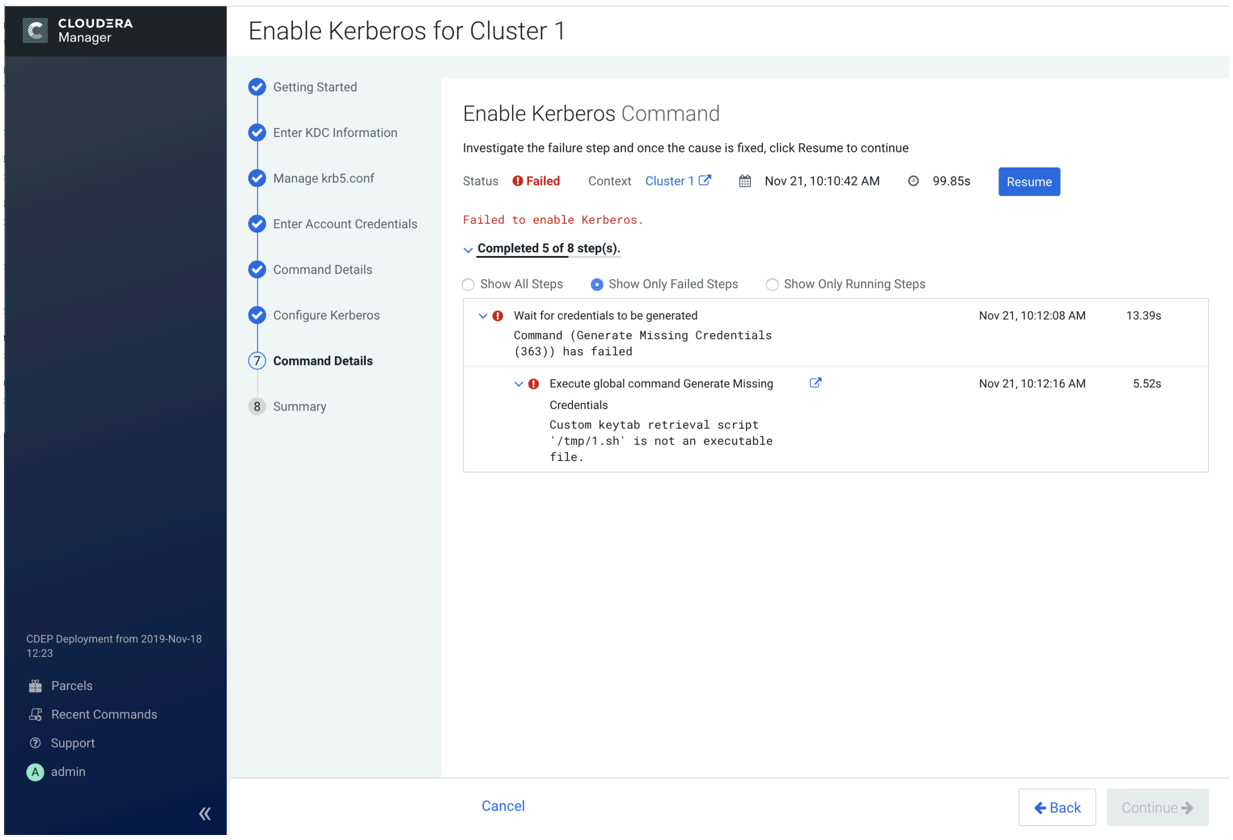
- The Command Details page displays the outcome of the Enable Kerberos command. It may take a few minutes for these steps to complete.

  If an error message appears, you can fix the error, then click Resume. In the following example, we select Administration > Settings > Kerberos, then enter an invalid file name in the Custom Kerberos Keytab Retrieval Script box.



Kerberos Ticket Lifetime ticket_lifetime	1	day(s)
Kerberos Renewable Lifetime renew_lifetime	7	day(s)
DNS Lookup KDC dns_lookup_kdc	☐	
Forwardable Tickets forwardable	☑	
KDC Timeout kdc_timeout	3	second(s)
Custom Kerberos Keytab Retrieval Script	/tmp/1.sh	
Maximum Renewable Life for Principals	5	day(s)
Active Directory Domain Controller Override		
Advanced Configuration Snippet (Safety Valve) for [libdefaults] section of krb5.conf		
Advanced Configuration Snippet (Safety Valve) for the Default Realm in krb5.conf		

  This causes a credential generation error. In this case we can remove the invalid script setting, then click Resume to resume the Enable Kerberos command.



Enable Kerberos Command

Investigate the failure step and once the cause is fixed, click Resume to continue

Status Failed Context Cluster 1 Nov 21, 10:10:42 AM 99.85s Resume

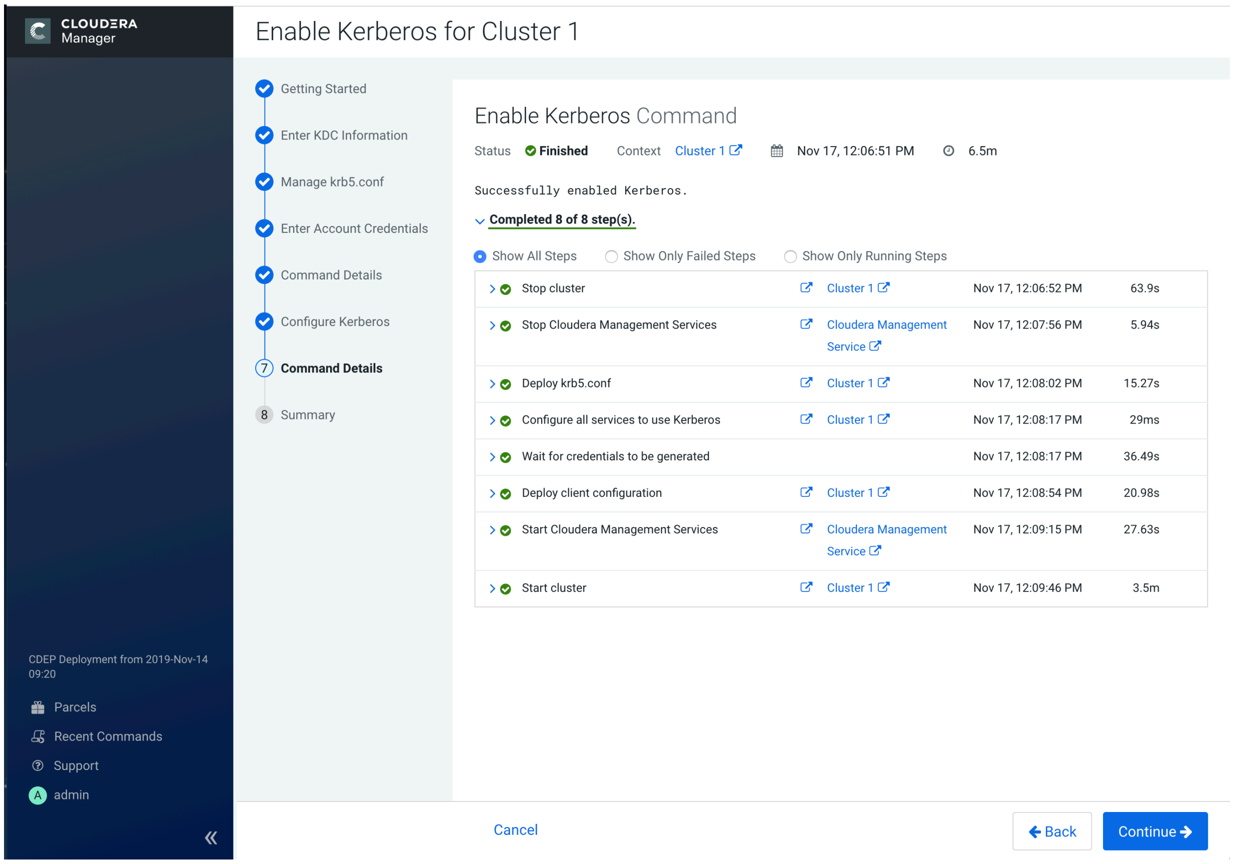
```
Failed to enable Kerberos.
```

Completed 5 of 8 step(s).

Show All Steps [✓ Show Only Failed Steps] Show Only Running Steps

Wait for credentials to be generated `Command (Generate Missing Credentials (363)) has failed`	Nov 21, 10:12:08 AM	13.39s
Execute global command Generate Missing Credentials `Custom keytab retrieval script '/tmp/1.sh' is not an executable file.`	Nov 21, 10:12:16 AM	5.52s

After the command finishes running, click Continue to proceed.



Enable Kerberos Command

Status Finished Context Cluster 1 Nov 17, 12:06:51 PM 6.5m

```
Successfully enabled Kerberos.
```

Completed 8 of 8 step(s).

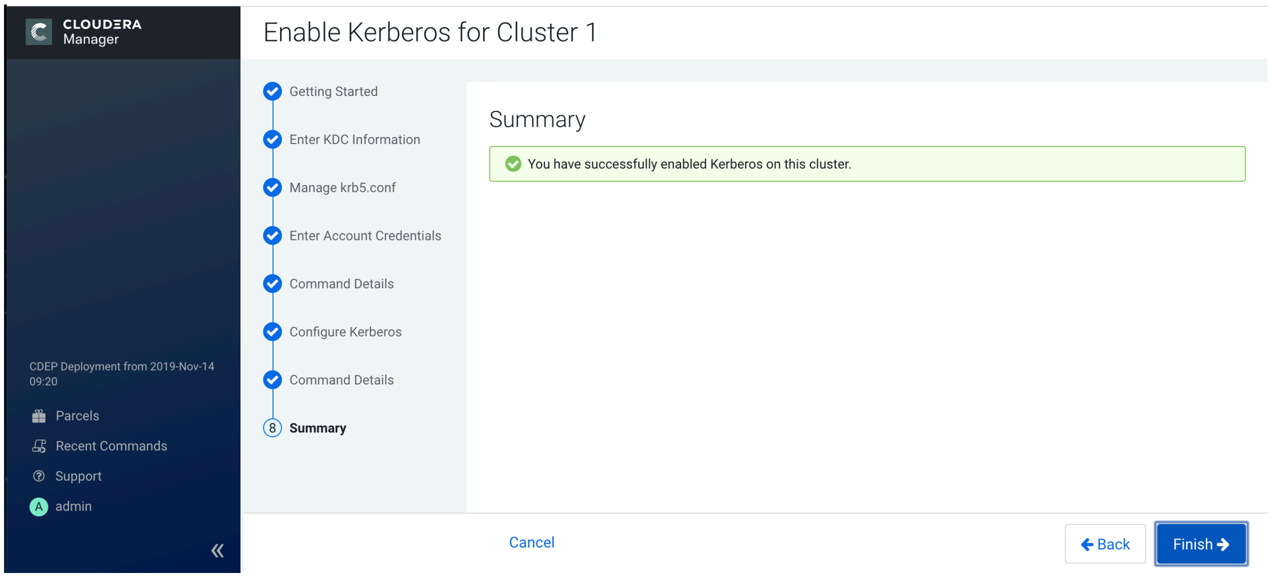
[✓ Show All Steps] Show Only Failed Steps Show Only Running Steps

Stop cluster	Cluster 1	Nov 17, 12:06:52 PM	63.9s
Stop Cloudera Management Services	Cloudera Management Service	Nov 17, 12:07:56 PM	5.94s
Deploy krb5.conf	Cluster 1	Nov 17, 12:08:02 PM	15.27s
Configure all services to use Kerberos	Cluster 1	Nov 17, 12:08:17 PM	29ms
Wait for credentials to be generated		Nov 17, 12:08:17 PM	36.49s
Deploy client configuration	Cluster 1	Nov 17, 12:08:54 PM	20.98s
Start Cloudera Management Services	Cloudera Management Service	Nov 17, 12:09:15 PM	27.63s
Start cluster	Cluster 1	Nov 17, 12:09:46 PM	3.5m

8. Summary

About this task

The final step of the wizard lists the cluster(s) for which Kerberos has been successfully enabled. Click Finish to return to the Cloudera Manager Admin Console home page.

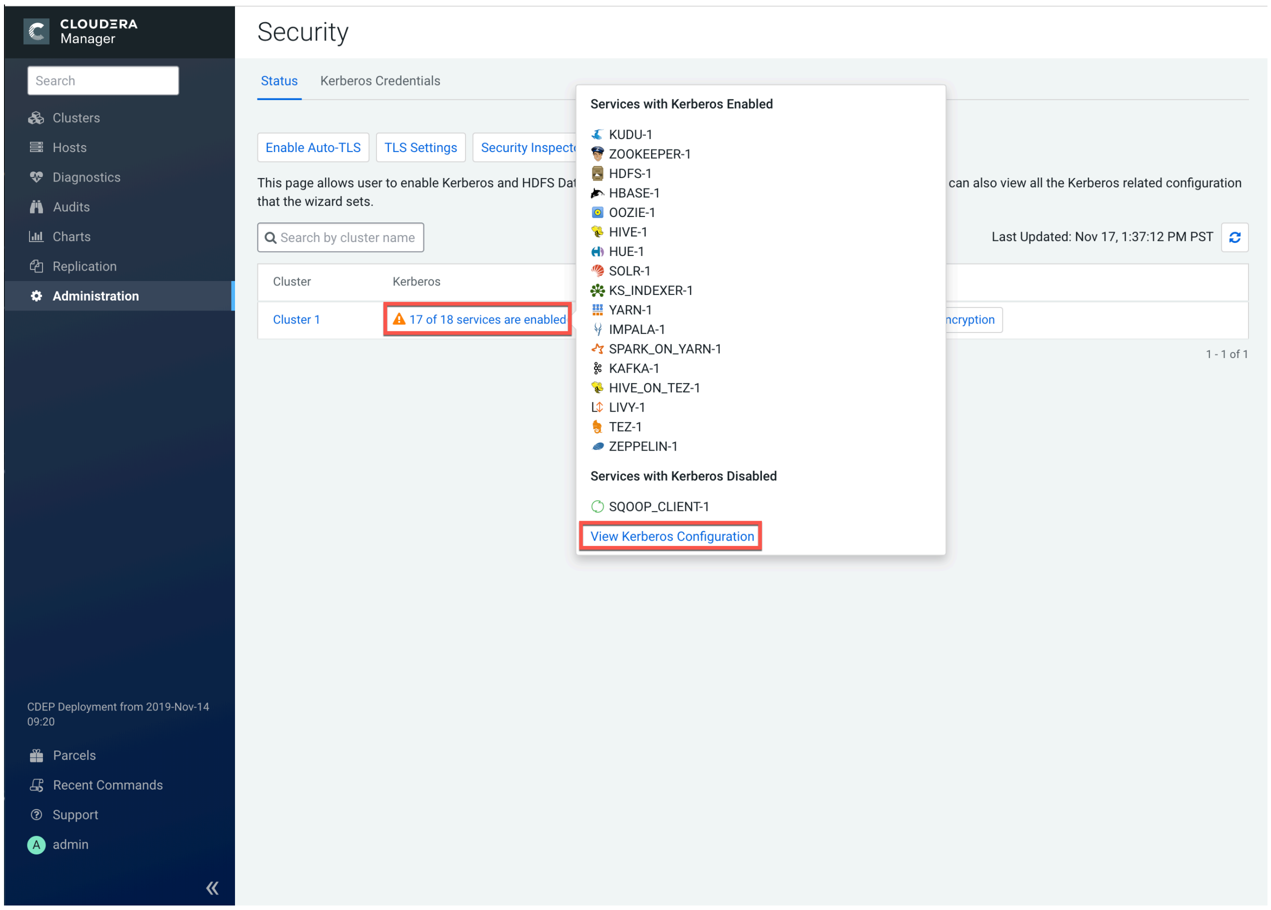


Summary

You have successfully enabled Kerberos on this cluster.

Results

To view details about Kerberos-enabled services, select Administration > Security, then click the applicable row in the Kerberos column.



This page allows user to enable Kerberos and HDFS Dat … can also view all the Kerberos related configuration that the wizard sets.

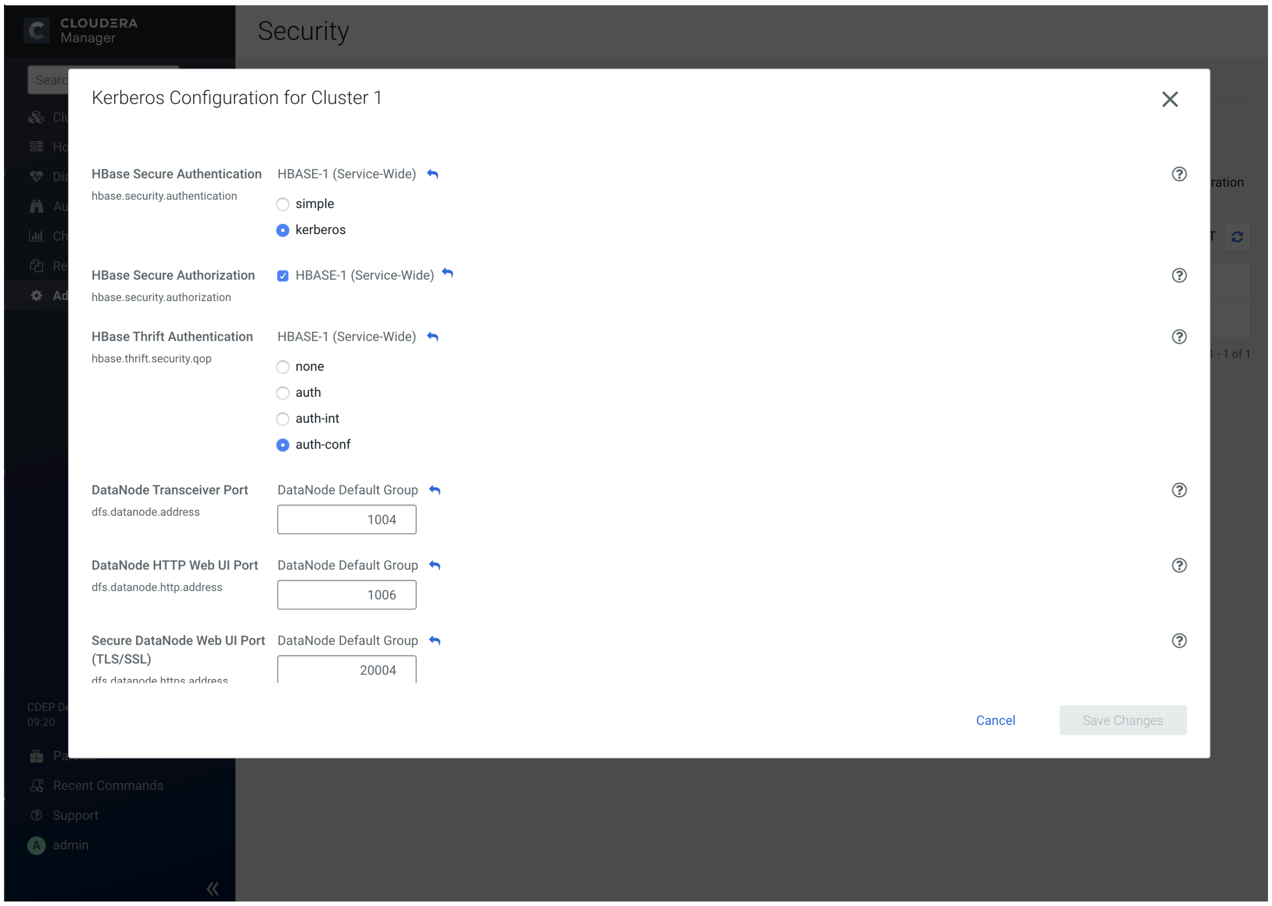
Services with Kerberos Enabled

- KUDU-1
- ZOOKEEPER-1
- HDFS-1
- HBASE-1
- OOZIE-1
- HIVE-1
- HUE-1
- SOLR-1
- KS_INDEXER-1
- YARN-1
- IMPALA-1
- SPARK_ON_YARN-1
- KAFKA-1
- HIVE_ON_TEZ-1
- LIVY-1
- TEZ-1
- ZEPPELIN-1

Services with Kerberos Disabled

- SQOOP_CLIENT-1

Click View Kerberos Configuration to view and edit configuration details for each service. Click Save Changes to save any configuration updates.


Kerberos Configuration for Cluster 1

HBase Secure Authentication
hbase.security.authentication
HBASE-1 (Service-Wide)
simple [✓ kerberos]

HBase Secure Authorization
hbase.security.authorization
✓ HBASE-1 (Service-Wide)

HBase Thrift Authentication
hbase.thrift.security.qop
HBASE-1 (Service-Wide)
none auth auth-int [✓ auth-conf]

DataNode Transceiver Port
dfs.datanode.address
DataNode Default Group
1004

DataNode HTTP Web UI Port
dfs.datanode.http.address
DataNode Default Group
1006

Secure DataNode Web UI Port (TLS/SSL)
dfs.datanode.https.address
DataNode Default Group
20004

Note:

The first five steps of this wizard can also be performed before cluster creation by selecting Administration > Security > Kerberos Credentials > Setup KDC for this Cloudera Manager. If that has already been done, the first five steps of this wizard do not appear.

Step 5: Create the HDFS superuser

To create home directories for users, you need access to the HDFS superuser account.

About this task

CDP automatically creates the HDFS superuser account on each cluster host during CDP installation. When you enable Kerberos for the HDFS service, you lose access to the default HDFS superuser account using sudo -u hdfs commands. Cloudera recommends that you use a different user account as the superuser, rather than the default hdfs account.

Designating a Non-default Superuser Group

About this task

To designate a different group of superusers instead of using the default hdfs account:

Procedure

1. Open the Cloudera Manager Admin console and navigate to the HDFS service.
2. Click the Configuration tab.
3. Select Scope HDFS (Service-Wide) .

4. Select Category Security .
5. Locate the Superuser Group property and change the value to the appropriate group name for your environment. For example, <*superuser*>.
6. Enter a Reason for change, then click Save Changes to commit the changes.
7. Restart the HDFS service.

What to do next

To enable your access to the superuser account now that Kerberos is enabled, you must now create a Kerberos principal or an Active Directory user whose first component is <*superuser*>:

If you are using Active Directory

Add a new user account to Active Directory, <superuser>@YOUR-REALM.COM. The password for this account should be set to never expire.

If you are using MIT KDC

1. In the kadmin.local or kadmin shell, type the following command to create a Kerberos principal called <*superuser*>:

   ```
   kadmin:  addprinc <superuser>@YOUR-LOCAL-REALM.COM
   ```

   This command prompts you to create a password for the <*superuser*> principal. You should use a strong password because having access to this principal provides superuser access to all of the files in HDFS.
2. To run commands as the HDFS superuser, you must obtain Kerberos credentials for the <*superuser*> principal. To do so, run the following command and provide the appropriate password when prompted.

   ```
   kinit <superuser>@YOUR-LOCAL-REALM.COM
   ```

If you are using Red Hat IdM/FreeIPA

1. On the Identity Users page, click the Add button.
2. Specify the superuser principal name in the User login field, complete the remaining fields, then click Add.

Step 6: Get or create a Kerberos principal for each user account

How to create a Kerberos principal for a user account.

About this task

Now that Kerberos is configured and enabled on your cluster, you and every other Hadoop user must have a Kerberos principal or keytab to obtain Kerberos credentials to be allowed to access the cluster and use the Hadoop services. In the next step of this procedure, you will need to create your own Kerberos principals to verify that Kerberos security is working on your cluster. If you and the other Hadoop users already have a Kerberos principal or keytab, or if your Kerberos administrator can provide them, you can skip ahead to the next step.

If you are using Active Directory

Procedure

- Add a new AD user account for each new user that should have access to the cluster. You do not need to make any changes to existing user accounts.

If you are using MIT KDC

Procedure

1. In the kadmin.local or kadmin shell, use the following command to create user principals by replacing YOUR-LOCAL-REALM.COM with the name of your realm, and replacing USERNAME with a username:

   ```
   kadmin:  addprinc USERNAME@YOUR-LOCAL-REALM.COM
   ```

2. Enter and re-enter a password when prompted.

Step 7: Prepare the cluster for each user

Before users access the cluster, there are a few tasks you must complete to prepare the hosts for each user.

Procedure

1. Make sure all hosts in the cluster have a Linux user account with the same name as the first component of that user's principal name. For example, the Linux account joe should exist on every box if the user's principal name is joe@YOUR-REALM.COM. You can use LDAP for this step if it is available in your organization.

   Note: Each account must have a user ID that is greater than or equal to 1000. In the /etc/hadoop/conf/taskcontroller.cfg file, the default setting for the banned.users property is mapred, hdfs, and bin to prevent jobs from being submitted using those user accounts. The default setting for the min.user.id property is 1000 to prevent jobs from being submitted with a user ID less than 1000, which are conventionally Unix super users.

2. Create a subdirectory under /user on HDFS for each user account (for example, /user/joe). Change the owner and group of that directory to be the user.

   ```
   hadoop fs -mkdir /user/joe
   hadoop fs -chown joe /user/joe
   ```

   Note: sudo -u hdfs is not included in the commands above. This is because it is not required if Kerberos is enabled on your cluster. You will, however, need to have Kerberos credentials for the HDFS super user to successfully run these commands. For information on gaining access to the HDFS super user account, see “Step 5: Create the HDFS superuser”.

Related Information

Step 5: Create the HDFS superuser

Step 8: Verify that Kerberos security is working

Verify that Kerberos security is working by running test MapReduce jobs.

Before you begin

After you have Kerberos credentials, you can verify that Kerberos security is working on your cluster by trying to run MapReduce jobs. To confirm, try launching a sleep or a pi job from the provided Hadoop examples (/usr/lib/hadoop/hadoop-examples.jar).

Note:

This section assumes you have a fully-functional CDP cluster and you have been able to access HDFS and run MapReduce jobs before you followed these instructions to configure and enable Kerberos on your cluster. If you have not already done so, you should at a minimum use the Cloudera Manager Admin Console to generate a client configuration file to enable you to access the cluster.

Procedure

1. Acquire Kerberos credentials for your user account.

```
kinit USERNAME@YOUR-LOCAL-REALM.COM
```

2. Enter a password when prompted.
3. Submit a sample pi calculation as a test MapReduce job. Use the following command if you use a package-based setup for Cloudera Manager:

```
$ hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar pi 10 10
000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 38.572 seconds
Estimated value of Pi is 3.14120000000000000000
```

If you have a parcel-based setup, use the following command instead:

```
$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-0.20-mapreduce/hadoop-
examples.jar pi 10 10000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 30.958 seconds
Estimated value of Pi is 3.14120000000000000000
```

What to do next

You have now verified that Kerberos security is working on your cluster.

Important:

Running a MapReduce job will fail if you do not have a valid Kerberos ticket in your credentials cache. You can examine the Kerberos tickets currently in your credentials cache by running the klist command. You can obtain a ticket by running the kinit command and either specifying a keytab file containing credentials, or entering the password for your principal. If you do not have a valid ticket, you will receive an error such as:

```
11/01/04 12:08:12 WARN ipc.Client:
Exception encountered while connecting to the server :
javax.security.sasl.SaslException:GSS initiate failed
[Caused by GSSException: No valid credentials provided (Mechanism level
: Failed to find any
Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10
.0.0.2:8020 failed on local exception:
java.io.IOException:javax.security.sasl.SaslException: GSS initiate f
ailed
[Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
```

Step 9: (Optional) Enable authentication for HTTP web consoles for Hadoop roles

Authentication for access to the web consoles for the HDFS and YARN roles can be enabled using a configuration option for the appropriate service.

Before you begin

Minimum Required Role: Configurator (also provided by Cluster Administrator, Limited Cluster Administrator , and Full Administrator)

Procedure

1. On the Clusters tab, select the service (HDFS or YARN) for which you want to enable authentication.
2. Click the Configuration tab.
3. Select Scope *service name* Service-Wide .
4. Type Enable Kerberos in the Search box.
5. Select Enable Kerberos Authentication for HTTP Web-Consoles.
6. elect Enable Kerberos Authentication for HTTP Web-Consoles.
7. Enter a Reason for change, then click Save Changes to commit the changes.
8. When the command finishes, restart all roles of that service.

Enabling SPNEGO as an Authentication Backend for Hue

Procedure

1. In Cloudera Manager, set the authentication backend to SpnegoDjangoBackend.
   a) Go to the Cloudera Manager Admin Console. From the Clusters tab, select the Hue service.
   b) Click the Configuration tab.
   c) Select Scope Service-Wide .
   d) Select Category Security .
   e) Locate the Authentication Backend property and select desktop.auth.backend.SpnegoDjangoBackend.
   f) Click Save Changes.
2. Restart the Hue service.
3. If you are using an external load balancer, perform the following steps, otherwise skip the remaining steps. Cloudera Manager creates these configuration automatically:
   a) On the host running the Hue Kerberos Ticket Renewer, switch to the KT_RENEWER process directory. For example:

```
cd /var/run/cloudera-scm-agent/process/`ls -lrt /var/run/cloudera-scm-ag
ent/process/    \
| awk '{print $9}' |grep KT_RENEWER| tail -1`/
```

   b) Verify that the Hue keytab includes the HTTP principal.

```
klist -kte ./hue.keytab
Keytab name: FILE:./hue.keytab
KVNO Timestamp Principal
---- ----------------- --------------------------------------------------
--------
1 03/09/15 20:20:35 hue/host-10-16-8-168.openstacklocal@EXAMPLE.CLOUDER
A.COM (aes128-cts-hmac-sha1-96)
1 03/09/15 20:20:36 HTTP/host-10-16-8-168.openstacklocal@EXAMPLE.CLOUDER
A.COM (aes128-cts-hmac-sha1-96)
```

   c) Copy the hue.keytab file to /var/lib/hue and change ownership to the hue user and group.

```
cp ./hue.keytab /var/lib/hue/
chown hue:hue /var/lib/hue/hue.keytab
```

   d) Go to the Cloudera Manager Admin Console. From the Clusters tab, select the Hue service.
   e) Click the Configuration tab.
   f) Select Scope Service-Wide .
   g) Select Category Advanced .

h) Locate the Hue Service Environment Advanced Configuration Snippet (Safety Valve) property and add the following line:

```
KRB5_KTNAME=/var/lib/hue/hue.keytab
```

i) Enter a Reason for change, then click Save Changes to commit the changes.
j) Restart the Hue service.

Kerberos authentication for non-default users

The steps described here are only applicable if you are running one or more CDP services with non-default users.

Procedure

- If you have modified the default value for the System User property for any service in Cloudera Manager, you must only perform the command (as described below) corresponding to that service, in order to successfully run jobs with the non-default user.

HBase	Give the hbase user ownership of the HBase root directory: `sudo -u hdfs hadoop fs -chown -R hbase ${hbase.rootdir}` By default, hbase.rootdir is /hbase.
Hive	Give the hive user ownership of the /user/hive directory. `sudo -u hdfs hadoop fs -chown hive /user/hive`
YARN	For every NodeManager host, for each path in yarn.nodemanager.local-dirs, run: `rm -rf ${yarn.nodemanager.local-dirs}/usercache/*` This removes the /usercache directory that contains intermediate data stored for previous jobs.

Customizing Kerberos principals

How to configure custom service principals in Cloudera Manager.

Related Information

Hadoop Users (user:group) and Kerberos Principals

About this task

By default, the Cloudera Manager Kerberos wizard configures CDP services to use the same Kerberos principals as the default process users. For example, the hdfs principal for the HDFS service, and the hive principal for the Hive service. The advantage to this is that when Kerberos is enabled, no HDFS directory permissions need to be changed for the new principals. You can also configure custom service principals for CDP services.

Important:

- Using different Kerberos principals for different services will make it easier to track the HDFS directories being accessed by each service.
- If you are using ShellBasedUnixGroupsMapping to obtain user-group mappings, ensure you have the UNIX accounts for the principals present on all hosts of the cluster.

Configuring Directory Permissions

Procedure

- Configure the following HDFS directories to give their corresponding custom service principals read, write and execute permissions.

Service	HDFS Directory
HBase	HBase Root Directory
Hive	• Hive Warehouse Directory • /user/*principal*
Impala	/user/*principal*
Oozie	Oozie ShareLib Root Directory
Solr	HDFS Data Directory
Spark on YARN	• /user/*principal* • Spark History Location • Spark Jar Location

Configuring CDP Services

About this task

The following services will require additional settings if you are using custom principals.

Procedure

- YARN - The principals used by YARN daemons should be part of hadoop group so that they are allowed to read JobHistory Server data.
- Impala - If you are running the Hue service with a custom principal, configure Impala to allow the Hue principal to impersonate other users.
  a) Go to the Impala service.
  b) Click Configuration.
  c) Select Scope Impala (Service-Wide) .
  d) Locate the Proxy User Configuration property and add the custom Hue principal.
  e) Click Save Changes.
- Spark on YARN - The principal used by the Spark service should be part of the spark group.
- Cloudera Management Service
  a) Go to the Cloudera Management Service.
  b) Click Configuration.
  c) Search for *kerberos*.
  d) Locate the Reports Manager Kerberos Principal property and set it to a principal with administrative and superuser privileges on all HDFS services.
  e) Locate the Navigator Kerberos Principal for HDFS property and set it to a principal with administrative and superuser privileges on all HDFS services.
  f) Click Save Changes.

Managing Kerberos credentials using Cloudera Manager

How to regenerate Kerberos principals in Cloudera Manager.

Related Information

Configuring a dedicated MIT KDC for cross-realm trust

Step 3: Create the Kerberos Principal for Cloudera Manager Server

About this task

Minimum Required Role: Full Administrator. This feature is not available when using Cloudera Manager to manage Data Hub clusters.

When Kerberos authentication is enabled for HDFS service instances, Cloudera Manager starts creating Kerberos principals for each role instance on the cluster at the end of the configuration process. Depending on the number of hosts and the number of HDFS role instances in the cluster, the process may take anywhere from a few seconds to several minutes.

After the process completes, view the list of Kerberos principals created for the cluster by using the Cloudera Manager Admin Console. Every host with HDFS role instances should have Kerberos principals.

If no principals have been created after 10 minutes, there may be an issue with the process.

Important:

- Do not regenerate the principals for your cluster unless you have made a global configuration change, such as changing the encryption type.
- Regenerate principals using the Cloudera Manager Admin Console only. Do not use kadmin shell.
- For an MIT KDC, see “Configuring a dedicated MIT KDC for cross-realm trust” to avoid invalidating existing host keytabs.

Updating Kerberos credentials in Cloudera Manager

About this task

If you change the user name or the password (or both) in the Active Directory KDC for the account used by Cloudera Manager for Kerberos authentication, you must also change it in Cloudera Manager. These credentials were stored during the Kerberos integration process (see “Step 3: Create the Kerberos Principal for Cloudera Manager Server”).

Procedure

1. Log in to Cloudera Manager Admin Console.
2. Select Administration Security .
3. Click the Kerberos Credentials tab.
4. Click the Import Kerberos Account Manager Credentials button.
5. Enter the new user and password for the principal added to the Kerberos KDC in “Step 3: Create the Kerberos Principal for Cloudera Manager Server”.

   If you are using Red Hat IdM/FreeIPA, enter the IPA admin credentials here. These admin credentials are not stored, and are used only to create a new user and role (named cmadin-<*random_id*> and cmadminrole, respectively) and retrieve its keytab. Cloudera Manager stores this keytab for future Kerberos operations, such as regenerating the credentials of the CDP service accounts.

Managing Active Directory account properties

About this task

If you are using an Active Directory KDC, Cloudera Manager lets you configure Active Directory accounts and customize the credential regeneration process using the Cloudera Manager Admin Console. You can also use Cloudera Manager to configure the encryption types to be used by your Active Directory account. Once you modify any Active Directory account properties, you must regenerate Kerberos credentials to reflect those changes. The credential regeneration process requires you to delete existing accounts before new ones are created.

By default, Cloudera Manager does not delete accounts in Active Directory, which means that to regenerate Kerberos principals contained in Active Directory, you need to manually delete the existing Active Directory accounts. You can either delete and regenerate all existing Active Directory accounts, or only delete those with the userPrincipalName (or login name) that you will later manually select for regeneration. If the accounts haven't already been deleted manually, the regeneration process will throw an error message saying that deletion of accounts is required before you proceed.

Modifying Active Directory account properties using Cloudera Manager

About this task

If you are using an Active Directory KDC, you can configure Active Directory account properties such as objectCl ass and accountExpires directly from the Cloudera Manager Admin Console. Any changes to these properties will be reflected in the regenerated Kerberos credentials.

Procedure

1. Go to the Cloudera Manager Admin Console and click the Administration tab.
2. Select Administration Settings .
3. Click the Kerberos category.
4. Locate the Active Directory Account Properties and edit as required. By default, the property will be set to:

   ```
   accountExpires=0,objectClass=top,objectClass=person,objectClass=organiza
   tionalPerson,objectClass=user
   ```
5. Locate the Active Directory Password Properties and edit the field as needed. By default, the property will be set to:

   ```
   length=12,minLowerCaseLetters=2,minUpperCaseLetters=2,minDigits=2,minSpa
   ces=0,minSpecialChars=0,specialChars=?.!$%^*()-_+=~
   ```
6. Click Save Changes.
7. Regenerate new credentials with the new properties.

Enabling credential regeneration for Active Directory accounts using Cloudera Manager

About this task

To avoid having to delete accounts manually, enable the Active Directory Delete Accounts on Credential Regeneration property. By default, this property is disabled. After enabling this feature, Cloudera Manager will delete existing Active Directory accounts automatically when new ones are created during regeneration.

Procedure

1. On the Cloudera Manager Admin Console, click the Administration tab.
2. Select Administration Settings .
3. Click the Kerberos category.
4. Locate the Active Directory Delete Accounts on Credential Regeneration and check the property to activate this capability.
5. Click Save Changes.

Configuring encryption types for Active Directory KDC using Cloudera Manager

About this task

Cloudera Manager allows you to configure the encryption types (or enctype) used by an Active Directory KDC to protect its data. Cloudera supports the following encryption types:

- rc4-hmac
- aes128-cts
- aes256-cts
- des-cbc-crc
- des-cbc-md5

To configure encryption types for an Active Directory KDC:

Procedure

1. Go to the Cloudera Manager Admin Console and click the Administration tab.
2. Select Administration Settings .
3. Click the Kerberos category.
4. Locate the Kerberos Encryption Types and click ✚ to add the encryption types you want Active Directory to use (see the list above for supported encryption types enctypes).
5. Check the checkbox for the Active Directory Set Encryption Types property. This will automatically set the Cloudera Manager AD account to use the encryption types configured in the previous step.
6. Click Save Changes.

Moving Kerberos principals to another OU within Active Directory

About this task

If you have a Kerberized cluster configured with an Active Directory KDC, you can use the following steps to move the Kerberos principals from one AD Organizational Unit (OU) to another.

Procedure

1. Create the new OU on the Active Directory Server.
2. Use AD's Delegate Control wizard to set the permissions on the new OU such that the configured Cloudera Manager admin account has the ability to Create, Delete and Manage User Accounts within this OU.
3. Stop the cluster.
4. Stop the Cloudera Management Service.
5. In Active Directory, move all the Cloudera Manager and CDP components' user accounts to the new OU.
6. In Cloudera Manager, select Administration Security .
7. On the Kerberos Credentials tab, click Configuration.
8. Select Scope Settings .
9. Select Category Kerberos .
10. Locate the Active Directory Suffix property and edit the value to reflect the new OU name.
11. Click Save Changes.

Viewing or regenerating Kerberos credentials using Cloudera Manager

About this task

To view Kerberos principals for the cluster from Cloudera Manager for either MIT Kerberos or Active Diretory:

Procedure

1. Log in to Cloudera Manager Admin Console.
2. Select Administration Security .

3. Click the Kerberos Credentials tab. The currently configured Kerberos principals for services running on the cluster display, such as:
    - For HDFS, principals hdfs/hostname and host/hostname
4. To regenerate any principals (if necessary):
    - Select the principal from the list.
    - Click Regenerate.

Running the Security Inspector

About this task

The Security Inspector uses the Host Inspector to run a security-related set of commands on the hosts in your cluster. It reports on matters such as how Java is configured for encryption and on the default realms configured on each host:

Procedure

1. Select Administration Security .
2. Click Security Inspector. Cloudera Manager begins several tasks to inspect the managed hosts.
3. After the inspection completes, click Download Result Data or Show Inspector Results to review the results.

Using a custom Kerberos keytab retrieval script

How to manually create principals and keytabs using a custom retrieval script.

About this task

The Cloudera Manager Kerberos setup procedure requires you to create an administrator account for the Cloudera Manager user. Cloudera Manager will then connect to your KDC and use this admin account to generate principals and keytabs for the remaining CDP services. If for some reason, you cannot create a Cloudera Manager administrator account on your KDC with the privileges to create other principals and keytabs for CDP services, then these will need to be created manually.

Cloudera Manager gives you the option to use a custom script to retrieve keytabs from the local filesystem. To use a custom Kerberos keytab retrieval script:

Procedure

1. The KDC administrators should create the required principals and keytabs, and store them securely on the Cloudera Manager Server host.
2. Create the keytab retrieval script. Your script should take two arguments: a full principal name for which it should retrieve a keytab, and a destination to which it can write the keytab. The script must be executable by the Cloudera Manager admin user, cloudera-scm. Depending on the principal name input by Cloudera Manager, the script should locate the corresponding keytab on the Cloudera Manager Server host (stored in step 1), and copy it into a location accessible to the cloudera-scm user. Here is a simple example:

```
#!/bin/bash
# Cloudera Manager will input a destination path
DEST="$1"
# Cloudera Manager will input the principal name in the format: <service>/
<fqdn>@REALM
PRINC="$2"

# Assuming the '<service>_<fqdn>@REALM.keytab' naming convention for key
tab files
IN=$(echo $PRINC | sed -e 's/\//_/')
SRC="/keytabs/${IN}.keytab"
```

```
# Copy the keytab to the destination input by Cloudera Manager
cp -v $SRC $DEST
```

Note that the script will change according to the keytab naming convention followed by your organization.

There are special requirements when providing your own keytabs and using this sample script:

- Do not use the Keytab File Owner:Group values listed in “Hadoop Users (user:group) and Kerberos Principals”; instead ensure that the Cloudera Manager user (cloudera-scm) has read access to the keytabs that you are creating.
- Create keytabs for the individual principals (e.g., "hive/host.example.com@EXAMPLE.COM")
- Name the keytab files for these individual principals matching the paradigm "<service>_<fqdn>@REALM.keytab". For example, Filename (*.keytab) = "hive_host.example.com@realm.keytab".

3. Configure the location for the script in Cloudera Manager:
   a) Go to the Cloudera Manager Admin console.
   b) Select Administration Settings .
   c) Select Category Kerberos .
   d) Locate the Custom Kerberos Keytab Retrieval Script and set it to point to the script created in step 2.
   e) Click Save Changes.

4. Once the Custom Kerberos Keytab Retrieval Script property is set, whenever Cloudera Manager needs a keytab, it will ignore all other Kerberos configuration and run the keytab retrieval script to copy the required keytab to the desired destination.

5. Cloudera Manager can now distribute the keytab to the services that need access to it.

Related Information

Hadoop Users (user:group) and Kerberos Principals

Adding trusted realms to the cluster

How to specify trusted realms in Cloudera Manager.

Before you begin

The Kerberos instance associated with a given cluster has its *REALM-NAME* specified as the default_realm in the Kerberos configuration file (krb5.conf) on the cluster's NameNode. Rules defined in the hadoop.security.auth_to_local property translate the Kerberos principals to local account names at the host level. The default rules simply remove the @REALM portion of the Kerberos principal and leave the short name.

To allow principals from other realms to use the cluster, the trusted realms must be specified in Cloudera Manager. For example, the Kerberos realm used by your cluster may have a trust relationship to a central Active Directory or MIT Kerberos realm. Add the central realm to the cluster as detailed in the following steps so that Cloudera Manager can apply the appropriate mapping rules to the principals from the trusted realm to access the cluster's services.

To specify trusted realms using Cloudera Manager:

Procedure

1. Log in to the Cloudera Manager Admin Console.
2. Select ClustersHDFS Service.
3. Click the Configuration tab.
4. Select HDFS (Service-Wide) for the Scope filter.
5. Select Security for the Category filter.

6. In the Search field, type Kerberos Realms to find the Trusted Kerberos Realms and Additional Rules to Map Kerberos Principals to Short Names settings.
7. Add other Kerberos realms that the cluster's realm can trust. Use all upper-case letters to specify the REALM name for MIT Kerberos or Active Directory realms:

   ```
   ANOTHER-REALM.EXAMPLE.COM
   ```

   To add multiple realms, click the plus (+) button.
8. Click Save Changes.

What to do next

For each trusted realm identified in Trusted Kerberos Realms, default mapping rules automatically strip the REALM name. To customize the mapping rules, specify additional rules in the Additional Rules to Map Kerberos Principals to Short Names setting, one rule per line. Cloudera Manager will wrap each rule in the appropriate XML tags and add to the generated core-site.xml file. To create custom rules and translate translate mixed-case Kerberos principals to lower-case Hadoop usernames.

If you specify custom mapping rules for a Kerberos realm using the Additional Rules to Map Kerberos Principals to Short Names setting, ensure that the same realm is not specified in the Trusted Kerberos Realms setting. If it is, the auto-generated rule (which only strips the realm from the principal and does no additional transformations) takes precedent, and the custom rule is ignored.

For these changes to take effect, you must restart the cluster and redeploy the client configuration, as follows:

1. On the Cloudera Manager Admin Console, Clusters Cluster-*n* to choose cluster-wide actions.
2. From the Actions drop-down button, select Deploy Client Configuration.

Using auth-to-local rules to isolate cluster users

How to use auth-to-local rules to restrict user access to specific clusters.

About this task

By default, the Hadoop auth-to-local rules map a principal of the form <username>/<hostname>@<REALM> to <username>. This means if there are multiple clusters in the same realm, then principals associated with hosts of one cluster would map to the same user in all other clusters.

For example, if you have two clusters, cluster1-host-[1..4].example.com and cluster2-host- [1..4].example.com, that are part of the same Kerberos realm, EXAMPLE.COM, then the cluster2 principal, hdfs/cluster2-host1.example.com@EXAMPLE.COM, will map to the hdfs user even on cluster1 hosts.

To prevent this, use auth-to-local rules as follows to ensure only principals containing hostnames of cluster1 are mapped to legitimate users.

Procedure

1. Go to the HDFS Service Configuration tab.
2. Select Scope HDFS (Service-Wide) .
3. Select Category Security .
4. In the Search field, type Additional Rules to find the Additional Rules to Map Kerberos Principals to Short Names settings.

5. Additional mapping rules can be added to the Additional Rules to Map Kerberos Principals to Short Names property. These rules will be inserted before the rules generated from the list of trusted realms (configured above) and before the default rule.

```
RULE:[2:$1/$2@$0](hdfs/cluster1-host1.example.com@EXAMPLE.COM)s/(.*)@EXA
MPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs/cluster1-host2.example.com@EXAMPLE.COM)s/(.*)@EXAM
PLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs/cluster1-host3.example.com@EXAMPLE.COM)s/(.*)@EXAMP
LE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs/cluster1-host4.example.com@EXAMPLE.COM)s/(.*)@EX
AMPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs.*@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/nobody/
```

   In the example, the principal hdfs/<hostname>@REALM is mapped to the hdfs user if <hostname> is one of the cluster hosts. Otherwise it gets mapped to nobody, thus ensuring that principals from other clusters do not have access to cluster1.

   If the cluster hosts can be represented with a regular expression, that expression can be used to make the configuration easier and more conducive to scaling. For example:

```
RULE:[2:$1/$2@$0](hdfs/cluster1-host[1-4].example.com@EXAMPLE.COM)s/(.*)
@EXAMPLE.COM/hdfs/
RULE:[2:$1/$2@$0](hdfs.*@EXAMPLE.COM)s/(.*)@EXAMPLE.COM/nobody/
```

   Note:

   It is not possible to use alternatives in capturing or non-capturing groups within the matching portion of the rule, because the use of round brackets in the expression is not currently supported. For example, the following rule would result in an error:

```
RULE:[2:$1/$2@$0](hdfs/cluster1-(hosta|hostb|hostc).example.com@EXA
MPLE.COM)s/(.*)@EXAMPLE.COM/hdfs/
```

6. Click Save Changes.
7. Restart the HDFS service and any dependent services.

Configuring a dedicated MIT KDC for cross-realm trust

How to configure a cluster-dedicated KDC and default realm in Cloudera Manager.

Related Information

Hadoop Users (user:group) and Kerberos Principals

About this task

Using Cloudera Manager to configure Kerberos authentication for the cluster creates several principals and keytabs automatically. Cloudera Manager also deploys the keytab files to every host in the cluster. See “Hadoop Users (user:group) and Kerberos Principals” for complete listing.

Note: The example below is specific for creating and deploying principals and keytab files for MIT Kerberos. See the appropriate documentation for other Kerberos implementations, such as Microsoft Active Directory, as needed.

Local and Remote Kerberos Admin Tools

About this task

Kerberos administrator commands can be run directly on the KDC server host or remotely, as shown in the table:

kadmin.local	Requires root access or Kerberos admin account. Use to log on directly to the KDC host.
kadmin	Use the logon to the KDC host system from another remote host over the network.

- To run Kerberos administration commands locally on the KRB host system:

  ```
  sudo kadmin.local
  ```

  Enter your Linux system password (for the sudo).
- To run Kerberos administration commands from any host:

  ```
  kadmin
  ```

  Enter your Kerberos administrator password.

Note: Commands shown for the kadmin shell can also be run at the kadmin.local shell.

Setting up a Cluster-Dedicated KDC and Default Realm for the Hadoop Cluster

About this task

Cloudera has tested the following configuration approaches to Kerberos security for clusters managed by Cloudera Manager. For administration teams that are just getting started with Kerberos security, we recommend starting with these approaches to the configuration of KDC services for a number of reasons.

The number of Service Principal Names (SPNs) that are created and managed by the Cloudera Manager server for a CDP cluster can be significant, so it is important to realize the potential impact on cluster uptime and overall operations if you choose to manage keytabs manually instead. The Cloudera Manager server manages the creation of service keytabs on the proper hosts based on the current configuration of the database. Manual keytab management can be error prone and introduce delays when deploying or moving services within the cluster, especially under time-sensitive conditions.

Cloudera Manager creates SPNs within a KDC that it can access with the kadmin command based on configuration of the /etc/krb5.conf file on the Cloudera Manager host. SPNs are created with the format service-name/host.fqdn.name@EXAMPLE.COM where service-name is the relevant CDP service name such as hue or hbase or hdfs.

If your site already has a working KDC, and any existing principals share the same name as any of the principals that Cloudera Manager creates, the Cloudera Manager Server generates a new randomized key for those principals, and consequently causes existing keytabs to become invalid.

This is why Cloudera recommends using a dedicated local MIT Kerberos KDC and realm for the Hadoop cluster. You can set up a one-way cross-realm trust from the cluster-dedicated KDC and realm to your existing central MIT Kerberos KDC, or to an existing Active Directory realm. Using this method, there is no need to create Hadoop service principals in the central MIT Kerberos KDC or in Active Directory, but principals (users) in the central MIT KDC or in Active Directory can be authenticated to Hadoop. The steps to implement this approach are as follows:

Procedure

1. Install and configure a cluster-dedicated MIT Kerberos KDC that will be managed by Cloudera Manager for creating and storing principals for the services supported by the cluster.

   Note: The krb5-server package includes a logrotate policy file to rotate log files monthly. To take advantage of this, install the logrotate package. No additional configuration is necessary.

2. See the example kdc.conf and krb5.conf files in in Sample Kerberos Configuration Files for configuration considerations for the KDC and Kerberos clients.
3. Configure a default Kerberos realm for the cluster you want Cloudera Manager to manage and set up one-way cross-realm trust between the cluster-dedicated KDC and either your central KDC or Active Directory, using the appropriate steps:

   - Using a Cluster-Dedicated KDC with a Central MIT KDC
   - Using a Cluster-Dedicated MIT KDC with Active Directory

   Cloudera strongly recommends the method above because:

   - It requires minimal configuration in Active Directory.
   - It is comparatively easy to script the creation of many principals and keytabs. A principal and keytab must be created for every daemon in the cluster, and in a large cluster this can be extremely onerous to do directly in Active Directory.
   - There is no need to involve central Active Directory administrators to get service principals created.
   - It allows for incremental configuration. The Hadoop administrator can completely configure and verify the functionality the cluster independently of integrating with Active Directory.

Using a Cluster-Dedicated KDC with a Central MIT KDC

About this task

Important: If you plan to use Oozie or the Hue Kerberos Ticket Renewer in your cluster, you must configure your KDC to allow tickets to be renewed, and you must configure krb5.conf to request renewable tickets. Typically, you can do this by adding the max_renewable_life setting to your realm in kdc.conf, and by adding the renew_lifetime parameter to the libdefaults section of krb5.conf.

Procedure

1. In the /var/kerberos/krb5kdc/kdc.conf file on the local dedicated KDC server host, configure the default realm for the Hadoop cluster by substituting your Kerberos realm in the following realms property:

```
[realms]
 HADOOP.EXAMPLE.COM = {
```

2. In the /etc/krb5.conf file on all cluster hosts and all Hadoop client user hosts, configure the default realm for the Hadoop cluster by substituting your Kerberos realm in the following realms property. Also specify the local dedicated KDC server hostname in the /etc/krb5.conf file (for example, kdc01.example.com).

```
[libdefaults]
  default_realm = HADOOP.EXAMPLE.COM
[realms]
  HADOOP.EXAMPLE.COM = {
    kdc = kdc01.hadoop.example.com:88
    admin_server = kdc01.hadoop.example.com:749
    default_domain = hadoop.example.com
  }
  EXAMPLE.COM = {
    kdc = kdc01.example.com:88
    admin_server = kdc01.example.com:749
    default_domain = example.com
```

```
   }
[domain_realm]
  .hadoop.example.com = HADOOP.EXAMPLE.COM
  hadoop.example.com = HADOOP.EXAMPLE.COM
  .example.com = EXAMPLE.COM
  example.com = EXAMPLE.COM
```

3. To set up the cross-realm trust in the cluster-dedicated KDC, type the following command in the kadmin.local or kadmin shell on the cluster-dedicated KDC host to create a krbtgt principal. Substitute your cluster-dedicated KDC realm for HADOOP.EXAMPLE.COM, and substitute your central KDC realm for EXAMPLE.COM. Enter a trust password when prompted. Note the password because you will need to enter the exact same password in the central KDC in the next step.

```
kadmin:  addprinc krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
```

4. Each of your Hadoop client users must also place this information in their local core-site.xml file. The easiest way to do so is by using the Cloudera Manager Admin Console to generate a client configuration file.
5. To set up the cross-realm trust in the central KDC, type the same command in the kadmin.local or kadmin shell on the central KDC host to create the exact same krbtgt principal and password.

```
kadmin:  addprinc krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
```

Important: For a cross-realm trust to operate properly, both KDCs must have the same krbtgt principal and password, and both KDCs must be configured to use the same encryption type.

6. To properly translate principal names from the central KDC realm into the cluster-dedicated KDC realm for the Hadoop cluster, configure the Trusted Kerberos Realms property of the HDFS service.
   a) Open the Cloudera Manager Admin Console.
   b) Go to the HDFS service.
   c) Click the Configuration tab.
   d) Select Scope HDFS (Service Wide) .
   e) Select Category Security .
   f) Type Kerberos in the Search box.
   g) Edit the Trusted Kerberos Realms property to add the name of your central KDC realm. If you need to use more advanced mappings which do more than just allow principals from another domain, you may enter them in the Additional Rules to Map Kerberos Principals to Short Names property.
7. Each of your Hadoop client users must also place this information in their local core-site.xml file. The easiest way to do so is by using the Cloudera Manager Admin Console to generate a client configuration file.
8. Later in this procedure, you will restart the services to have the configuration changes in core-site.xml take effect.

Using a Cluster-Dedicated MIT KDC with Active Directory

About this task

For Cloudera Manager clusters, the openldap-clients package must be installed on the Cloudera Manager Server host before configuring the cluster to use Kerberos for authentication.

On the Active Directory Server

Procedure

1. On the Active Directory server host, type the following command to add the local realm trust to Active Directory:

```
netdom trust HADOOP.EXAMPLE.COM /Domain:EXAMPLE.COM /add /realm /passwor
dt:TrustPassword
```

2. On the Active Directory server host, type the following command to set the proper encryption type:

   Windows 2003 RC2

   Windows 2003 server installations do not support AES encryption for Kerberos.

   ```
   ktpass /MITRealmName HADOOP.EXAMPLE.COM /TrustEncryp RC4
   ```

   Windows 2008

   ```
   ksetup /SetEncTypeAttr HADOOP.EXAMPLE.COM <enc_type>
   ```

   Where the <enc_type> parameter can be replaced with parameter strings for AES, DES, or RC4 encryption modes. For example, for AES encryption, replace <enc_type> with AES256-CTS-HMAC-SHA1-96 or AES128-CTS-HMAC-SHA1-96 and for RC4 encryption, replace with RC4-HMAC-MD5.

   Important: Make sure that the encryption type you specify is supported on both your version of Windows Active Directory and your version of MIT Kerberos.

On the MIT KDC Server

Procedure

1. In the /var/kerberos/krb5kdc/kdc.conf file on the local dedicated KDC server host, configure the default realm for the Hadoop cluster by substituting your Kerberos realm in the following realms property:

   ```
   [realms]
    HADOOP.EXAMPLE.COM = {
   ```

2. Each of your Hadoop client users must also place this information in their local core-site.xml file. The easiest way to do so is by using the Cloudera Manager Admin Console to generate a client configuration file.
3. On the local MIT KDC server host, type the following command in the kadmin.local or kadmin shell to add the cross-realm krbtgt principal:

   ```
   kadmin:  addprinc -e "<keysalt_list>" krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.
   COM
   ```

   where the <keysalt_list> parameter specifies the types of keys and their salt to be used for encryption of the password for this cross-realm krbtgt principal. It can be set to AES, or RC4 keytypes with a salt value of :normal. Note that DES is deprecated and should no longer be used. You can specify multiple keysalt types using the parameter in the command above. Make sure that at least one of the encryption types corresponds to the encryption types found in the tickets granted by the KDC in the remote realm. For an example of the values to use, see the examples based on the Active Directory functional domain level, below.

   Examples by Active Directory Domain or Forest "Functional level"

   Active Directory will, based on the Domain or Forest functional level, use encryption types supported by that release of the Windows Server operating system. It is not possible to use AES encryption types with an AD 2003 functional level. If you notice that DES encryption types are being used when authenticating or requesting service tickets to Active Directory then it might be necessary to enable weak encryption types in the /etc/krb5.conf.

   - Windows 2003

     ```
     kadmin: addprinc -e "rc4-hmac:normal" krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.
     COM
     ```

- Windows 2008

  ```
  kadmin: addprinc -e "aes256-cts:normal aes128-cts:normal rc4-hmac:normal
  " krbtgt/HADOOP.EXAMPLE.COM@EXAMPLE.COM
  ```

Note: The cross-realm krbtgt principal that you add in this step must have at least one entry that uses the same encryption type as the tickets that are issued by the remote KDC. If there are no matching encryption types, principals in the local realm can successfully access the Hadoop cluster, but principals in the remote realm are unable to.

On All Cluster Hosts

Procedure

1. In the /etc/krb5.conf file on all cluster hosts and all Hadoop client user hosts, configure both Kerberos realms. Note that default_realm should be configured as the local MIT Kerberos realm for the cluster. Your krb5.conf may contain more configuration properties than those demonstrated below. This example is provided to clarify configuration parameters.

   ```
   [libdefaults]
     default_realm = HADOOP.EXAMPLE.COM
   [realms]
     EXAMPLE.COM = {
       kdc = dc01.example.com:88
       admin_server = dc01.example.com:749
     }
     HADOOP.EXAMPLE.COM = {
       kdc = kdc01.hadoop.example.com:88
       admin_server = kdc01.hadoop.example.com:749
     }
   [domain_realm]
     .hadoop.example.com = HADOOP.EXAMPLE.COM
     hadoop.example.com = HADOOP.EXAMPLE.COM
     .example.com = EXAMPLE.COM
     example.com = EXAMPLE.COM
   ```

2. Use one of the following methods to properly translate principal names from the Active Directory realm into the cluster-dedicated KDC realm for the Hadoop cluster.

   - Using Cloudera Manager: Configure the Trusted Kerberos realms property of the HDFS service:

     a. Open the Cloudera Manager Admin Console.
     b. Go to the HDFS service.
     c. Click the Configuration tab.
     d. Select Scope > HDFS (Service Wide)
     e. Select Category > Security.
     f. Type Kerberos in the Search box.
     g. Edit the Trusted Kerberos Realms property to add the name of your central KDC realm. If you need to use more advanced mappings which do more than just allow principals from another domain, you may enter them in the Additional Rules to Map Kerberos Principals to Short Names property.

   - Using the Command Line: Configure the hadoop.security.auth_to_local setting in the core-site.xml file on all of the cluster hosts. The following example translates all principal names with the realm EXAMPLE.COM into the first component of the principal name only. It also preserves the standard translation for the default realm (the cluster realm).

     ```
     <property>
       <name>hadoop.security.auth_to_local</name>
       <value>
         RULE:[1:$1@$0](^.*@EXAMPLE\.COM$)s/^(.*)@EXAMPLE\.COM$/$1/g
     ```

```
    RULE:[2:$1@$0](^.*@EXAMPLE\.COM$)s/^(.*)@EXAMPLE\.COM$/$1/g
    DEFAULT
  </value>
</property>
```

Integrating MIT Kerberos and Active Directory

How to integrate MIT Kerberos and Active Directory in a Cloudera Manager cluster.

About this task

Several different subsystems are involved in servicing authentication requests, including the Key Distribution Center (KDC), Authentication Service (AS), and Ticket Granting Service (TGS). The more nodes in the cluster and the more services provided, the heavier the traffic between these services and the services running on the cluster.

As a general guideline, Cloudera recommends using a dedicated Active Directory instance (Microsoft Server Domain Services) for every 100-200 nodes in the cluster. However, this is just a loose guideline. Monitor utilization and deploy additional instances as needed to meet the demand.

By default, Kerberos uses UDP for client/server communication which is typically faster at delivering packets than TCP, but does not guarantee delivery. Additionally, using UDP packets that get too large are frequently dropped, as is the case when a user is a member of a large number of groups. To avoid this problem, force Kerberos to use TCP by modifying the Kerberos configuration file (krb5.conf) as follows:

```
[libdefaults]
udp_preference_limit = 1
...
```

This is especially useful if the domain controllers are not on the same subnet as the cluster or are separated by firewalls.

In general, Cloudera recommends setting up the Active Directory domain controller (Microsoft Server Domain Services) on the same subnet as the cluster and never over a WAN connection which results in considerable latency and affects cluster performance.

Troubleshooting cluster operations when Active Directory is being used for Kerberos authentication requires administrative access to the Microsoft Server Domain Services instance. Administrators may need to enable Kerberos event logging on the Microsoft Server KDC to resolve issues.

Deleting Cloudera Manager roles or nodes requires manually deleting the associate Active Directory accounts. Cloudera Manager cannot delete entries from Active Directory.

Related Information

Mapping Kerberos Principals to Short Names

Integrating MIT Kerberos and Active Directory

About this task

Prior to release 5.1, clusters managed by Cloudera Manager could not integrate directly with a Microsoft Active Directory KDC. Rather, integrating the Cloudera Manager cluster with an Active Directory KDC required the additional setup of a local MIT KDC (local, meaning in the same subnet as the cluster).

The steps below can be used for Cloudera Manager clusters prior to release 5.1, or if the Active Directory KDC cannot be accessed directly for whatever reason. The setup process assumes that:

- An MIT Kerberos KDC is running in the same subnet as the cluster and that a Kerberos REALM is local to the cluster

- A Microsoft Server Active Directory instance (Microsoft Server Domain Services) is running elsewhere on the network, in its own Kerberos realm.

Given these two systems, you can then:

1. Create principals for all services running on the cluster in the MIT Kerberos realm local to the cluster.
2. Set up one-way cross-realm trust from the MIT Kerberos realm to the Active Directory realm, as detailed in "Configuring a Local MIT Kerberos Realm to Trust Active Directory" below.

The result of this setup is that Active Directory principals (users) can authenticate to the cluster without needing service principals.

Configuring a Local MIT Kerberos Realm to Trust Active Directory

On the Active Directory Server

Procedure

1. Add the local realm trust to Active Directory with this command:

   ```
   netdom trust YOUR-LOCAL-REALM.COMPANY.COM /Domain:AD-REALM.COMPANY.COM /
   add /realm /passwordt:<TrustPassword>
   ```

2. Set the proper encryption type with this command:

   On Microsoft Windows 2008, 2012, or 2016:

   ```
   ksetup /SetEncTypeAttr YOUR-LOCAL-REALM.COMPANY.COM <enc_type>
   ```

   The <enc_type> parameter specifies AES, DES, or RC4 encryption. Refer to the documentation for your version of Windows Active Directory to find the <enc_type> parameter string to use.
3. Get and verify the list of encryption types set with this command:

   On Microsoft Windows 2008, 2012, or 2016:

   ```
   ksetup /GetEncTypeAttr YOUR-LOCAL-REALM.COMPANY.COM
   ```

   Important: Make sure the encryption type you specify is supported on both your version of Windows Active Directory and your version of MIT Kerberos.

On the MIT KDC Server

Procedure

- Type the following command in the kadmin.local or kadmin shell to add the cross-realm krbtgt principal. Use the same password you used in the netdom command on the Active Directory Server.

  ```
  kadmin:  addprinc -e "enc_type_list" krbtgt/YOUR-LOCAL-REALM.COMPANY.CO
  M@AD-REALM.COMPANY.COM
  ```

  where the *enc_type_list* parameter specifies the types of encryption this cross-realm krbtgt principal can support —AES, DES, or RC4. You can specify multiple encryption types using the parameter in the command above, what's important is that at least one of the encryption types corresponds to the encryption type found in the tickets granted by the KDC in the remote realm. For example:

```
kadmin:  addprinc -e "rc4-hmac:normal des3-hmac-sha1:normal" krbtgt/YOUR-
LOCAL-REALM.COMPANY.COM@AD-REALM.COMPANY.COM
```

Note: The cross-realm krbtgt principal that you add in this step must have at least one entry that uses the same encryption type as the tickets that are issued by the remote KDC. If no entries have the same encryption type, then the problem you will see is that authenticating as a principal in the local realm will allow you to successfully run Hadoop commands, but authenticating as a principal in the remote realm will not allow you to run Hadoop commands.

On All of the Cluster Hosts

Procedure

1. Verify that both Kerberos realms are configured on all of the cluster hosts. Note that the default realm and the domain realm should remain set as the MIT Kerberos realm which is local to the cluster.

```
[realms]
  AD-REALM.CORP.FOO.COM = {
    kdc = ad.corp.foo.com:88
    admin_server = ad.corp.foo.com:749
    default_domain = foo.com
  }
  CLUSTER-REALM.CORP.FOO.COM = {
    kdc = cluster01.corp.foo.com:88
    admin_server = cluster01.corp.foo.com:749
    default_domain = foo.com
  }
```

2. To properly translate principal names from the Active Directory realm into local names within Hadoop, you must configure the hadoop.security.auth_to_local setting in the core-site.xml file on all of the cluster machines. The following example translates all principal names with the realm AD-REALM.CORP.FOO.COM into the first component of the principal name only. It also preserves the standard translation for the default realm (the cluster realm).

   a.
```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
    RULE:[1:$1@$0](^.*@AD-REALM\.CORP\.FOO\.COM$)s/^(.*)@AD-REALM\.CORP
\.FOO\.COM$/$1/g
    RULE:[2:$1@$0](^.*@AD-REALM\.CORP\.FOO\.COM$)s/^(.*)@AD-REALM\.CORP
\.FOO\.COM$/$1/g
    DEFAULT
  </value>
</property>
```

   For more information about name mapping rules, see “Mapping Kerberos Principals to Short Names”.

Hadoop Users (user:group) and Kerberos Principals

During the Cloudera Manager installation process, several Linux user accounts and groups are created by default. These are listed in the table below. Integrating the cluster to use Kerberos for authentication requires creating Kerberos principals and keytabs for these user accounts.

Table 2: Users and Groups

Component (Version)	Unix User ID	Groups	Functionality
Apache Atlas	atlas	atlas, hadoop	Apache Atlas by default has atlas as user and group. It is configurable
Apache Flink	flink	flink	The Flink Dashboard runs as this user.
Apache HBase	hbase	hbase	The Master and the RegionServer processes run as this user.
Apache HBase Indexer	hbase	hbase	The indexer servers are run as this user.
Apache HDFS	hdfs	hdfs, hadoop	The NameNode and DataNodes run as this user, and the HDFS root directory as well as the directories used for edit logs should be owned by it.
Apache Hive Hive on Tez	hive	hive	The HiveServer2 process and the Hive Metastore processes run as this user.A user must be defined for Hive access to its Metastore DB (for example, MySQL or Postgres) but it can be any identifier and does not correspond to a Unix uid. This is javax.jdo.option.ConnectionUserName in hive-site.xml.
Apache Impala	impala	impala, hive	Impala services run as this user.
Apache Kafka	kafka	kafka	Kafka brokers, mirrorMaker, and Connect workers run as this user.
Apache Knox	knox	knox	Apache Knox Gateway Server runs as this user
Apache Kudu	kudu	kudu	Kudu services run as this user.
Apache Livy	livy	livy	The Livy Server process runs as this user
Apache NiFi	nifi	nifi	Runs as the nifi user
Apache NiFi Registry	nifiregistry	nifiregistry	Runs as the nifiregistry user
Apache Oozie	oozie	oozie	The Oozie service runs as this user.
Apache Ozone	hdfs	hdfs, hadoop	Ozone Manager, Storage Container Manager (SCM), Recon and Ozone Datanodes run as this user.
Apache Parquet	~	~	No special users.
Apache Phoenix	phoenix	phoenix	The Phoenix Query Server runs as this user
Apache Ranger	ranger	ranger, hadoop	Ranger Admin, Usersync and Tagsync services by default have ranger as user and ranger, hadoop as groups. It is configurable.
Apache Ranger KMS	kms	kms	Ranger KMS runs with kms user and group. It is configurable.
Apache Ranger Raz	rangerraz	ranger	Ranger Raz runs with rangerraz user and is part of the ranger group.

Component (Version)	Unix User ID	Groups	Functionality
Apache Ranger RMS	rangerrms	ranger	Ranger RMS runs with rangerrms user and is part of the ranger group.
Apache Solr	solr	solr	The Solr processes run as this user.
Apache Spark	spark	spark	The Spark History Server process runs as this user.
Apache Sqoop	sqoop	sqoop	This user is only for the Sqoop1 Metastore, a configuration option that is not recommended.
Apache YARN	yarn	yarn, hadoop	Without Kerberos, all YARN services and applications run as this user. The LinuxContainerExecutor binary is owned by this user for Kerberos.
Apache Zeppelin	zeppelin	zeppelin	The Zeppelin Server process runs as this user
Apache ZooKeeper	zookeeper	zookeeper	The ZooKeeper processes run as this user. It is not configurable.
Cloudera Manager (all versions)	cloudera-scm	cloudera-scm	Clusters managed by Cloudera Manager run Cloudera Manager Server, monitoring roles, and other Cloudera Server processes as cloudera-scm. Requires keytab file named cmf.keytab because name is hard-coded in Cloudera Manager.
Cruise Control	cruisecontrol	hadoop	The Cruise Control process runs as this user.
HttpFS	httpfs	httpfs	The HttpFS service runs as this user. See “HttpFS authentication” for instructions on how to generate the merged httpfs-http.keytab file.
Hue	hue	hue	Hue services run as this user.
Hue Load Balancer	apache	apache	The Hue Load balancer has a dependency on the apache2 package that uses the apache user name. Cloudera Manager does not run processes using this user ID.
Key Trustee Server	keytrustee	keytrustee	The Key Trustee Server service runs as this user.
Schema Registry	schemaregistry	hadoop	The Schema Registry process runs as this user.
Streams Messaging Manager	streamsmsgmgr	streamsmsgmgr	The Streams Messaging Manager processes runs as this user.
Streams Replication Manager	streamsrepmgr	streamsrepmgr	The Streams Replication Manager processes runs as this user.

Keytabs and Keytab File Permissions

Linux user accounts, such as hdfs, are mapped to the username portion of the Kerberos principal names, as follows:

```
username/host.example.com@EXAMPLE.COM
```

For example, the Kerberos principal for Apache Hive would be:

```
hive/host.example.com@EXAMPLE.COM
```

Keytabs that contain multiple principals are merged automatically from individual keytabs by Cloudera Manager. If you override a service configuration to not use the CM-provided keytab, then you must ensure that all the principals required for the given role instance on a specific host are merged together in the keytab file you deploy manually on that host.

For example, for Filename (*.keytab), the Atlas keytab filename would be atlas.keytab, HBase would be hbase.ke ytab, and Cloudera Manager would be cmf.keytab and scm.keytab.

Keytab File Owner:Group matters when Cloudera Manager starts a role. For example, Cloudera Manager starts the role "DataNode"". Cloudera Manager launches the DataNode process as a user (here, "hdfs"). Because that process needs to access the HDFS keytab, Cloudera Manager puts the HDFS keytab in the DataNode's process directory, and the keytab is given the owner:group that is listed in the table. Thus, the DataNode process properly owns the keytab file.

The tables below lists the usernames to use for Kerberos principal names, for clusters managed by Cloudera Manager.

Apache Atlas

Role: atlas-ATLAS_SERVER

Kerberos Principals

atlas

Filename (*.keytab)

atlas

Keytab File Owner:Group

atlas:atlas

File Permission (octal)

600

Apache Flink

Role: flink

Kerberos Principals

flink

Filename (*.keytab)

flink

Keytab File Owner:Group

flink:flink

File Permission (octal)

600

Apache HBase

Role: hbase-HBASETHRIFTSERVER

Kerberos Principals

hbase, HTTP

Filename (*.keytab)

hbase, HTTP

Keytab File Owner:Group

hbase:hbase

File Permission (octal)

600

Role: hbase-REGIONSERVER
Kerberos Principals

hbase, HTTP

Filename (*.keytab)

hbase, HTTP

Keytab File Owner:Group

hbase:hbase

File Permission (octal)

600

Role: hbase-HBASERESTSERVER
Kerberos Principals

hbase, HTTP

Filename (*.keytab)

hbase, HTTP

Keytab File Owner:Group

hbase:hbase

File Permission (octal)

600

Role: hbase-MASTER
Kerberos Principals

hbase, HTTP

Filename (*.keytab)

hbase, HTTP

Keytab File Owner:Group

hbase:hbase

File Permission (octal)

600

Apache HBase indexer

Role: ks_indexer-HBASE_INDEXER
Kerberos Principals

hbase, HTTP

Filename (*.keytab)

hbase

Keytab File Owner:Group

hbase:hbase

File Permission (octal)

600

Apache HDFS

Role: hdfs-NAMENODE
Kerberos Principals

hdfs, HTTP

Filename (*.keytab)

hdfs

Keytab File Owner:Group

hdfs:hdfs

File Permission (octal)

600

Role: hdfs-DATANODE
Kerberos Principals

hdfs, HTTP

Filename (*.keytab)

hdfs

Keytab File Owner:Group

hdfs:hdfs

File Permission (octal)

600

Role: hdfs-SECONDARYNAMENODE
Kerberos Principals

hdfs, HTTP

Filename (*.keytab)

hdfs

Keytab File Owner:Group

hdfs:hdfs

File Permission (octal)

600

Apache Hive, Hive on Tez

Role: hive-HIVESERVER2
Kerberos Principals

hive

Filename (*.keytab)

hive

Keytab File Owner:Group

hive:hive

File Permission (octal)

600

Role: hive-HIVEMETASTORE
Kerberos Principals

hive

Filename (*.keytab)

hive

Keytab File Owner:Group

cloudera-scm:cloudera-scm

File Permission (octal)

600

Apache Impala

Role: impala-STATESTORE

Kerberos Principals

impala, HTTP

Filename (*.keytab)

impala

Keytab File Owner:Group

impala:impala

File Permission (octal)

600

Role: impala-CATALOGSERVER

Kerberos Principals

impala, HTTP

Filename (*.keytab)

impala

Keytab File Owner:Group

impala:impala

File Permission (octal)

600

Role: impala-IMPALAD

Kerberos Principals

impala, HTTP

Filename (*.keytab)

impala

Keytab File Owner:Group

impala:impala

File Permission (octal)

600

Apache Kafka

Role: kafka-KAFKA_BROKER

Kerberos Principals

kafka

Filename (*.keytab)

kafka

Keytab File Owner:Group

kafka:kafka

File Permission (octal)

600

Role: kafka-KAFKA_MIRROR_MAKER

Kerberos Principals

kafka_mirror_maker

Filename (*.keytab)

kafka

Keytab File Owner:Group

kafka:kafka

File Permission (octal)

600

Role: kafka-KAFKA_CONNECT
Kerberos Principals

kafka

Filename (*.keytab)

kafka

Keytab File Owner:Group

kafka:kafka

File Permission (octal)

600

Apache Knox

Role: knox-KNOX_GATEWAY
Kerberos Principals

knox, HTTP

Filename (*.keytab)

hbase

Keytab File Owner:Group

knox:knox

File Permission (octal)

600

Apache Kudu

Role: kudu-KUDU_MASTER
Kerberos Principals

kudu

Filename (*.keytab)

kudu

Keytab File Owner:Group

kudu:kudu

File Permission (octal)

600

Role: kudu-KUDU_TSERVER
Kerberos Principals

kudu

Filename (*.keytab)

kudu

Keytab File Owner:Group

kudu:kudu

File Permission (octal)

600

Apache Livy

Role: livy-LIVY_SERVER

Kerberos Principals

livy

Filename (*.keytab)

livy

Keytab File Owner:Group

livy:livy

File Permission (octal)

600

Apache NiFi

Role: nifi

Kerberos Principals

nifi, HTTP

Filename (*.keytab)

nifi

Keytab File Owner:Group

nifi:nifi

File Permission (octal)

600

Apache NiFi Registry

Role: nifiregistry

Kerberos Principals

nifiregistry, HTTP

Filename (*.keytab)

nifiregistry

Keytab File Owner:Group

nifiregistry:nifiregistry

File Permission (octal)

600

Apache Oozie

Role: oozie-OOZIE_SERVER

Kerberos Principals

oozie, HTTP

Filename (*.keytab)

oozie

Keytab File Owner:Group

oozie:oozie

File Permission (octal)

600

Apache Ozone

Role: ozone-OZONE_MANAGER

Kerberos Principals
: om, HTTP

Filename (*.keytab)
: ozone

Keytab File Owner:Group
: hdfs:hdfs

File Permission (octal)
: 600

Role: ozone-STORAGE_CONTAINER_MANAGER

Kerberos Principals
: scm, HTTP

Filename (*.keytab)
: ozone

Keytab File Owner:Group
: hdfs:hdfs

File Permission (octal)
: 600

Role: ozone-OZONE_DATANODE

Kerberos Principals
: dn, HTTP

Filename (*.keytab)
: ozone

Keytab File Owner:Group
: hdfs:hdfs

File Permission (octal)
: 600

Role: ozone-OZONE_RECON

Kerberos Principals
: recon, HTTP

Filename (*.keytab)
: ozone

Keytab File Owner:Group
: hdfs:hdfs

File Permission (octal)
: 600

Role: ozone-S3_GATEWAY

Kerberos Principals
: HTTP

Filename (*.keytab)
: ozone

Keytab File Owner:Group
: hdfs:hdfs

File Permission (octal)

600

Apache Phoenix

Role: phoenix-PHOENIX_QUERY_SERVER
Kerberos Principals

phoenix, HTTP

Filename (*.keytab)

phoenix

Keytab File Owner:Group

phoenix:phoenix

File Permission (octal)

600

Apache Ranger

Role: ranger-RANGER_ADMIN
Kerberos Principals

rangeradmin, rangerlookup, HTTP

Filename (*.keytab)

ranger

Keytab File Owner:Group

ranger:ranger

File Permission (octal)

600

Role: ranger-RANGER_USERSYNC
Kerberos Principals

rangerusersync

Filename (*.keytab)

ranger

Keytab File Owner:Group

ranger:ranger

File Permission (octal)

600

Role: ranger-RANGER_TAGSYNC
Kerberos Principals

rangertagsync

Filename (*.keytab)

ranger

Keytab File Owner:Group

ranger:ranger

File Permission (octal)

600

Apache Ranger KMS

Role: ranger-RANGER_TAGSYNC
Kerberos Principals

rangerkms, HTTP

Filename (*.keytab)

ranger_kms

Keytab File Owner:Group

kms:kms

File Permission (octal)

600

Apache Ranger Raz

Role: ranger-RANGER_RAZ

Kerberos Principals

rangerraz, HTTP

Filename (*.keytab)

rangerraz

Keytab File Owner:Group

ranger:rangerraz

File Permission (octal)

600

Apache Ranger RMS

Role: ranger-RANGER_RMS

Kerberos Principals

rangerrms

Filename (*.keytab)

rangerrms

Keytab File Owner:Group

ranger:rangerrms

File Permission (octal)

600

Apache Solr

Role: solr-SOLR_SERVER

Kerberos Principals

solr, HTTP

Filename (*.keytab)

solr

Keytab File Owner:Group

solr:solr

File Permission (octal)

600

Apache Spark

Role: spark_on_yarn-SPARK_YARN_HISTORY_SERVER

Kerberos Principals

spark

Filename (*.keytab)

spark

Keytab File Owner:Group

spark:spark

File Permission (octal)

600

Apache YARN

Role: yarn-NODEMANAGER
Kerberos Principals

yarn, HTTP

Filename (*.keytab)

yarn

Keytab File Owner:Group

yarn:hadoop

File Permission (octal)

644

Role: yarn-RESOURCEMANAGER
Kerberos Principals

yarn, HTTP

Filename (*.keytab)

yarn

Keytab File Owner:Group

yarn:hadoop

File Permission (octal)

600

Role: yarn-JOBHISTORY
Kerberos Principals

mapred

Filename (*.keytab)

mapred

Keytab File Owner:Group

yarn:hadoop

File Permission (octal)

600

Apache Zeppelin

Role: zeppelin-ZEPPELIN_SERVER
Kerberos Principals

zeppelin, HTTP

Filename (*.keytab)

zeppelin

Keytab File Owner:Group

zeppelin:zeppelin

File Permission (octal)

600

Apache ZooKeeper

Role: zookeeper-server

Kerberos Principals

zookeeper

Filename (*.keytab)

zookeeper

Keytab File Owner:Group

zookeeper:zookeeper

File Permission (octal)

600

Cloudera Management

Role: cloudera-mgmt-REPORTSMANAGER

Kerberos Principals

hdfs

Filename (*.keytab)

headlamp

Keytab File Owner:Group

cloudera-scm:cloudera-scm

File Permission (octal)

600

Role: cloudera-mgmt-SERVICEMONITOR

Kerberos Principals

hue

Filename (*.keytab)

cmon

Keytab File Owner:Group

cloudera-scm:cloudera-scm

File Permission (octal)

600

Role: cloudera-mgmt-ACTIVITYMONITOR

Kerberos Principals

hue

Filename (*.keytab)

cmon

Keytab File Owner:Group

cloudera-scm:cloudera-scm

File Permission (octal)

600

Cloudera Manager

Kerberos Principals

cloudera-scm, HTTP

Filename (*.keytab)

cmf, scm

Keytab File Owner:Group

cloudera-scm:cloudera-scm

File Permission (octal)

600

CruiseControl

Role: cruise_control-CRUISE_CONTROL_SERVER

Kerberos Principals

cruisecontrol, kafka, HTTP

Filename (*.keytab)

cruise_control

Keytab File Owner:Group

cruisecontrol:hadoop

File Permission (octal)

600

HttpFS

Role: hdfs-HTTPFS

Kerberos Principals

httpfs, HTTP

Filename (*.keytab)

httpfs

Keytab File Owner:Group

httpfs:httpfs

File Permission (octal)

600

Hue

Role: hue-KT_RENEWER

Kerberos Principals

hue

Filename (*.keytab)

hue

Keytab File Owner:Group

hue:hue

File Permission (octal)

600

Schema Registry

Role: schemaregistry-SCHEMA_REGISTRY_SERVER

Kerberos Principals

schemaregistry, HTTP

Filename (*.keytab)

schemaregistry

Keytab File Owner:Group

schemaregistry:hadoop

File Permission (octal)

600

Streams Messaging Manager

Role: streams_messaging_manager-STREAMS_MESSAGING_MANAGER_SERVER

Kerberos Principals

streamsmsgmgr, HTTP

Filename (*.keytab)

streams_messaging_manager

Keytab File Owner:Group

streamsmsgmgr:streamsmsgmgr

File Permission (octal)

600

Streams Replication Manager

Role: streams_replication_manager-STREAMS_REPLICATION_MANAGER_DRIVER

Kerberos Principals

streamsrepmgr

Filename (*.keytab)

streams_replication_manager

Keytab File Owner:Group

streamsrepmgr:streamsrepmgr

File Permission (octal)

600

Role: streams_replication_manager-STREAMS_REPLICATION_MANAGER_SERVICE

Kerberos Principals

streamsrepmgr

Filename (*.keytab)

streams_replication_manager

Keytab File Owner:Group

streamsrepmgr:streamsrepmgr

File Permission (octal)

600

Related Information

Using a custom Kerberos keytab retrieval script

Customizing Kerberos principals

Mapping Kerberos Principals to Short Names

You configure the mapping from Kerberos principals to short names in the hadoop.security.auth_to_local property setting in the core-site.xml file. Kerberos has this support natively, and Hadoop's implementation uses Kerberos's configuration language to specify the mapping.

A mapping consists of a set of rules that are evaluated in the order listed in the hadoop.security.auth_to_local property. The first rule that matches a principal name is used to map that principal name to a short name. Any later rules in the list that match the same principal name are ignored.

You specify the mapping rules on separate lines in the hadoop.security.auth_to_local property as follows:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
  RULE:[<principal translation>](<acceptance filter>)<short name substitut
ion>
  RULE:[<principal translation>](<acceptance filter>)<short name substituti
on>
  DEFAULT
  </value>
</property>
```

Note: You can also set auth_to_local rules in krb5.conf using MIT Kerberos. If you do, be aware that the format is different, and the rules described below will fail if used in this context.

Mapping Rule Syntax

To specify a mapping rule, use the prefix string RULE: followed by three sections—principal translation, acceptance filter, and short name substitution—described in more detail below. The syntax of a mapping rule is:

```
RULE:[<principal translation>](<acceptance filter>)<short name substitution>
```

Principal Translation

The first section of a rule, <principal translation>, performs the matching of the principal name to the rule. If there is a match, the principal translation also does the initial translation of the principal name to a short name. In the <princip al translation> section, you specify the number of components in the principal name and the pattern you want to use to translate those principal component(s) and realm into a short name. In Kerberos terminology, a principal name is a set of components separated by slash ("/") characters.

The principal translation is composed of two parts that are both specified within "[ ]" using the following syntax:

```
[<number of components in principal name>:<initial specification of short na
me>]
```

where:

<number of components in principal name> – This first part specifies the number of components in the principal name (not including the realm) and must be 1 or 2. A value of 1 specifies principal names that have a single component (for example, hdfs), and 2 specifies principal names that have two components (for example, hdfs/fully.q ualified.domain.name). A principal name that has only one component will only match single-component rules, and a principal name that has two components will only match two-component rules.

<initial specification of short name> – This second part specifies a pattern for translating the principal component(s) and the realm into a short name. The variable $0 translates the realm, $1 translates the first component, and $2 translates the second component.

Here are some examples of principal translation sections. These examples use atm@YOUR-REALM.COM and atm/ fully.qualified.domain.name@YOUR-REALM.COM as principal name inputs:

This Principal Translation	Translates atm@YOUR-REALM.COM into this short name	Translates atm/fully.qualified.domain.name@YOUR-REALM.COM into this short name
[1:$1@$0]	atm@YOUR-REALM.COM	Rule does not match1

This Principal Translation	Translates atm@YOUR-REALM.COM into this short name	Translates atm/fully.qualified.domain.name@YOUR-REALM.COM into this short name
[1:$1]	atm	Rule does not match1
[1:$1.foo]	atm.foo	Rule does not match1
[2:$1/$2@$0]	Rule does not match2	atm/fully.qualified.domain.name@YOUR-REALM.COM
[2:$1/$2]	Rule does not match2	atm/fully.qualified.domain.name
[2:$1@$0]	Rule does not match2	atm@YOUR-REALM.COM
[2:$1]	Rule does not match2	atm

Footnotes:

1Rule does not match because there are two components in principal name atm/fully.qualified.domain.name@YOUR-REALM.COM

2Rule does not match because there is one component in principal name atm@YOUR-REALM.COM

Acceptance Filter

The second section of a rule, (<acceptance filter>), matches the translated short name from the principal translation (that is, the output from the first section). The acceptance filter is specified in "( )" characters and is a standard regular expression. A rule matches only if the specified regular expression matches the entire translated short name from the principal translation. That is, there's an implied ^ at the beginning of the pattern and an implied $ at the end.

Short Name Substitution

The third and final section of a rule is the (<short name   substitution>). If there is a match in the second section, the acceptance filter, the (<short name   substitution>) section does a final translation of the short name from the first section. This translation is a sed replacement expression (s/.../.../g) that translates the short name from the first section into the final short name string. The short name substitution section is optional. In many cases, it is sufficient to use the first two sections only.

Converting Principal Names to Lowercase

In some organizations, naming conventions result in mixed-case usernames (for example, John.Doe) or even uppercase usernames (for example, JDOE) in Active Directory or LDAP. This can cause a conflict when the Linux username and HDFS home directory are lowercase.

To convert principal names to lowercase, append /L to the rule.

Example Rules

Suppose all of your service principals are either of the form App.service-name/fully.qualified.domain.name@YOUR-REALM.COM or App.service-name@YOUR-REALM.COM, and you want to map these to the short name string service-name. To do this, your rule set would be:

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
 RULE:[1:$1](App\..*)s/App\.(.*)/$1/g
  RULE:[2:$1](App\..*)s/App\.(.*)/$1/g
  DEFAULT
  </value>
</property>
```

The first $1 in each rule is a reference to the first component of the full principal name, and the second $1 is a regular expression back-reference to text that is matched by (.*).

In the following example, suppose your company's naming scheme for user accounts in Active Directory is FirstnameLastname (for example, JohnDoe), but user home directories in HDFS are /user/firstnamelastname. The following rule set converts user accounts in the CORP.EXAMPLE.COM domain to lowercase.

```
<property>
  <name>hadoop.security.auth_to_local</name>
  <value>
  RULE:[2:$1@$0](HTTP@\QCORP.EXAMPLE.COM\E$)s/@\QCORP.EXAMPLE.COM\E$//
  RULE:[1:$1@$0](.*@\QCORP.EXAMPLE.COM\E$)s/@\QCORP.EXAMPLE.COM\E$///L
  RULE:[2:$1@$0](.*@\QCORP.EXAMPLE.COM\E$)s/@\QCORP.EXAMPLE.COM\E$///L
  DEFAULT
  </value>
</property>
```

In this example, the JohnDoe@CORP.EXAMPLE.COM principal becomes the johndoe HDFS user.

Default Rule

You can specify an optional default rule called DEFAULT (see example above). The default rule reduces a principal name down to its first component only. For example, the default rule reduces the principal names atm@YOUR-REALM.COM or atm/fully.qualified.domain.name@YOUR-REALM.COM down to atm, assuming that the default domain is YOUR-REALM.COM.

The default rule applies only if the principal is in the default realm.

If a principal name does not match any of the specified rules, the mapping for that principal name will fail.

Testing Mapping Rules

You can test mapping rules for a long principal name by running:

```
hadoop org.apache.hadoop.security.HadoopKerberosName name1 name2 name3
```