Cloudera Manager 7.13.2

How-to Guides

Date published: 2020-11-30
Date modified: 2026-02-27

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

How to Add Root and Intermediate CAs to Truststore for TLS/SSL

If a signed certificate is from a certificate authority (CA) that does not have certificates in the truststore (internal CA or a public CA not included in the Java truststore, for example), you must explicitly establish trust for the CA.

About this task

The content of the truststore for the Cloudera Manager Server cluster can be modified (certificates added, for example) without restarting the server. Changes to the truststore are adopted by the system within 10 seconds.

Explicit Trust for Certificates

About this task

Before importing the certificate into the keystore of the host system, you must load the root CAs and any intermediate CAs into the truststore.

Procedure

1. Copy the root and intermediate CA certificates to these locations on the Cloudera Manager Server host:

   ```
   /opt/cloudera/security/pki/rootca.cert.pem
   /opt/cloudera/security/pki/intca.cert.pem
   ```

   a) For concatenated files containing root CA and intermediate CA certificates, split the file between the END CERTIFICATE and BEGIN CERTIFICATE boundaries that separate each certificate in the file and make individual files instead.

   b) When extracting multiple intermediate CA certificates from a concatenated file, use unique file names such as intca-1.cert.pem, intca-2.cert.pem, and so on.

2. Import the root CA certificate into the JDK truststore. If you do not have the $JAVA_HOME variable set, replace it with the path to the Oracle JDK.

   ```
   $ sudo keytool -importcert -alias rootca -keystore $JAVA_HOME/jre/lib/se
   curity/jssecacerts \
   -file /opt/cloudera/security/pki/rootca.cert.pem -storepass changeit
   ```

   The default password for the cacerts file is changeit (as shown in the above command). Cloudera recommends changing this password by running the keytool command:

   ```
   keytool -storepasswd -keystore $JAVA_HOME/jre/lib/security/cacerts
   ```

3. Copy the jssecacerts file from the Cloudera Manager Server host to all other cluster hosts. Copy the file to the same location on each host using the path required by Oracle JDK, which is as follows:

   ```
   $JAVA_HOME/jre/lib/security/jssecacerts
   ```

4. On the Cloudera Manager Server host, append the intermediate CA certificate to the signed server certificate. Be sure to use the append (>>) operator—not overwrite (>)—when executing the statement:

   ```
   $ sudo cat /opt/cloudera/security/pki/intca.cert.pem >> \
   ```

```
/opt/cloudera/security/pki/$(hostname -f)-server.cert.pem
```

Amazon Web Services (AWS) Security

Amazon Web Services (AWS) is Amazon's cloud solution that offers compute, storage, networking, and other infrastructure services that can be used for Cloudera cluster deployments, whether completely cloud-based or in combination with on-premises clusters.

For example, Amazon Elastic Compute Cloud (EC2) can be used for the instances that make-up the nodes of a Cloudera cluster deployed to the AWS cloud. Amazon's cloud-based storage solution, Amazon Simple Storage Service (Amazon S3), can be used by both on-premises and AWS-cloud-based clusters in various ways, including as storage for Impala tables for direct use by Data Explorer and Hive, and other Cloudera components such as HDFS client, Hive, Impala, MapReduce.

As of release 5.11, Cloudera Manager supports Amazon's IAM-role based access to Amazon S3 storage, in addition to its prior support of AWS access key and secret key.

For any AWS service, including Amazon S3, you must obtain an Amazon Web Services account and have appropriate access to the AWS Management Console to set up the various services you want, including Amazon S3. Assuming you have an account for AWS, to provide access from your Cloudera cluster to Amazon S3 storage you must configure AWS credentials.

Getting Started with Amazon Web Services

To get started with AWS, including Amazon S3, you must have:

1. An Amazon Web Services account. Both Amazon and Cloudera recommend that you do not use your primary Amazon account—known as the root account—for working with Amazon S3 and other AWS services. See the AWS IAM documentation for details about how to set up your AWS account.
2. Access to the AWS Management Console and appropriate permissions to create and configure the AWS services needed for your use case, such as the following:

   a. AWS Elastic Compute Cloud (EC2) to deploy your cluster to the AWS cloud.
   b. AWS Identity and Access Management (IAM) to set up users and groups, or to set up an IAM role.
   c. Amazon S3 and the specific storage bucket (or buckets) for use with your cluster.
   d. Amazon DynamoDB to enable the database needed by Cloudera S3Guard, if you plan to enable S3Guard for your cluster. Cloudera S3Guard augments Amazon S3 with a database to track metadata changes so that the 'eventual consistency' model inherent to Amazon S3 does not pose a problem for transactions or other use cases in which changes may not be apparent to each other in real time. See “Configuring and Managing S3Guard” in Cloudera Administration for details. To use S3Guard, you will also need to set up the appropriate access policy (create table, read, write) to DynamoDB for the same AWS identity that owns the Amazon S3 storage.
   e. AWS Key Management Services (KMS) (AWS KMS) to create encryption keys for your Amazon S3 bucket if you plan to use SSE-KMS for server-side encryption (not necessary for SSE-S3 encryption. See “How to Configure Encryption for Amazon S3” for details).

Configuration Properties Reference

This table provides reference documentation for the core-site.xml properties relevant for use with AWS and Amazon S3.

Property	Description
fs.s3a.server-side-encryption-algorithm	Enable server-side encryption for the Amazon S3 storage bucket associated with the cluster. Allowable values: • AES256 Specifies SSE-S3 server-side encryption for Amazon S3. • SSE-KMS Specifies SSE-KMS server-side encryption for Amazon S3. Requires adding the fs.s3a.server-side-encryption-key property with a valid value.
fs.s3a.server-side-encryption-key	Specify the ARN, ARN plus alias, Alias, or globally unique ID of the key created in AWS Key Management Service for use with SSE-KMS.
fs.s3a.awsAccessKeyId	Specify the AWS access key ID. This property is irrelevant and not used to access Amazon S3 storage from a cluster launched using an IAM role.
fs.s3a.awsSecretAccessKey	Specify the AWS secret key provided by Amazon. This property is irrelevant and not used to access Amazon S3 storage from a cluster launched using an IAM role.
fs.s3a.endpoint	Use this property only if the endpoint is outside the standard region (s3.amazonaws.com), such as regions and endpoints in China or in the US GovCloud. See “AWS regions and endpoints” documentation for details.
fs.s3a.connection.ssl.enabled	Enables (true) and disables (false) TLS/SSL connections to Amazon S3. Default is true.

Connecting to Amazon S3 Using TLS

The boolean parameter fs.s3a.connection.ssl.enabled in core-site.xml controls whether the hadoop-aws connector uses TLS when communicating with Amazon S3. Because this parameter is set to true by default, you do not need to configure anything to enable TLS. If you are not using TLS on Amazon S3, the connector will automatically fall back to a plaintext connection.

The root Certificate Authority (CA) certificate that signed the Amazon S3 certificate is trusted by default. If you are using custom truststores, make sure that the configured truststore for each service trusts the root CA certificate.

To import the root CA certificate into your custom truststore, run the following command:

```
$JAVA_HOME/bin/keytool -importkeystore -srckeystore $JAVA_HOME/jre/lib/security/cacerts -destkeystore /PATH/TO/CUSTOM/TRUSTSTORE -srcalias baltimorecybertrustca
```

If you are using S3Guard, you must import an additional CA certificate:

```
$JAVA_HOME/bin/keytool -importkeystore -srckeystore $JAVA_HOME/jre/lib/security/cacerts -destkeystore /PATH/TO/CUSTOM/TRUSTSTORE -srcalias verisignclass3g5ca
```

The aliases are appended with [jdk] as follows:

```
$JAVA_HOME/bin/keytool -importkeystore -srckeystore $JAVA_HOME/jre/lib/security/cacerts -destkeystore /PATH/TO/CUSTOM/TRUSTSTORE -srcalias "baltimorecybertrustca [jdk]"
$JAVA_HOME/bin/keytool -importkeystore -srckeystore $JAVA_HOME/jre/lib/security/cacerts -destkeystore /PATH/TO/CUSTOM/TRUSTSTORE -srcalias "verisignclass3g5ca [jdk]"
```

If you do not have the $JAVA_HOME variable set, replace it with the path to the Oracle JDK (for example, /usr/java/-cloudera). When prompted, enter the password for the destination and source truststores. The default password for the Oracle JDK cacerts truststore is changeit.

The truststore configurations for each service that accesses S3 are as follows:

hadoop-aws Connector

All components that can use Amazon S3 storage rely on the hadoop-aws connector, which uses the built-in Java truststore ($JAVA_HOME/jre/lib/security/cacerts). To override this truststore, create a truststore named jssecacerts in the same directory ($JAVA_HOME/jre/lib/security/jssecacerts) on all cluster nodes. If you are using the jssecacerts truststore, make sure that it includes the root CA certificate that signed the Amazon S3 certificate.

Hive/Beeline CLI

The Hive and Beeline command line interfaces (CLI) rely on the HiveServer2 truststore. To view or modify the truststore configuration:

1. Go to the Hive service in the Cloudera Manager Admin Interface.
2. Select the Configuration tab.
3. Select Scope *HIVE-1* (Service-Wide) .
4. Select Category Security .
5. Locate the HiveServer2 TLS/SSL Certificate Trust Store File and HiveServer2 TLS/SSL Certificate Trust Store Password properties or search for them by typing Trust in the Search box.

Impala Shell

The Impala shell uses the hadoop-aws connector truststore. To override it, create the $JAVA_HOME/jre/lib/security/jssecacerts file, as described in “hadoop-aws Connector”.

Hue S3 File Browser

The S3 file browser uses TLS if it is enabled, and the S3 File Browser trusts the S3 certificate by default. No additional configuration is necessary.

Impala Query Editor (Hue)

The Impala query editor in Data Explorer uses the hadoop-aws connector truststore. To override it, create the $JAVA_HOME/jre/lib/security/jssecacerts file, as described in “hadoop-aws Connector”.

Hive Query Editor (Hue)

The Hive query editor in Data Explorer uses the HiveServer2 truststore. For instructions on viewing and modifying the HiveServer2 truststore, see “Hive/Beeline CLI”.

How to Authenticate Kerberos Principals Using Java

How to authenticate Kerberos principals using Java.

To authenticate Kerberos principals in custom applications using Java, import the UserGroupInformation class from the Hadoop security library into your application and use it to pass principals and keytabs to the Kerberos service. This basic example shows hard-coded passing of principal cloudera and a cloudera.keytab file:

```
// Authenticating Kerberos principal
System.out.println("Principal Authentication: ");
final String user = "cloudera@CLOUDERA.COM";
final String keyPath = "cloudera.keytab";
UserGroupInformation.loginUserFromKeytab(user, keyPath);
```

The UserGroupInformation class (org.apache.hadoop.security.UserGroupInformation) has methods to handle tokens, proxy users, and many other tasks required for your Java application to work with Kerberos. See the Apache API documentation for details.

Nested LDAP Group Resolution

An LDAP group can include both regular users and pointers to other groups. If a group (group A) is a member of another group (group B), then the users belonging to the member, or nested group (group A), are part of the parent group (group B) as well. Nesting can be very useful when delegating access through inheritance. Many large enterprises have their groups in LDAP/AD nested within other groups.

Resolving nested group memberships for an LDAP user can pose a challenge, as most LDAP servers by default do not match nested groups on the server side. Instead, LDAP client applications are expected to perform multiple searches, iterating through the group's member list until all members are found, or the search has reached a configured depth limit. Microsoft Active Directory LDAP is an exception, and returns nested group members when the search request includes the proper extended match operator.

Ranger Usersync supports nested group lookups for both Active Directory and generic LDAPv3 servers (OpenLDAP, freeIPA, and so on). For more information, see *Configuring nested group hierarchies*. For other Cloudera services, nested group lookups are typically possible only with Active Directory, with some limitations.

Active Directory LDAP_MATCHING_RULE_IN_CHAIN

By default, Microsoft Active Directory supports the LDAP_MATCHING_RULE_IN_CHAIN extended match operator for memberOf and member attributes (in User and Group objects, respectively), with OID 1.2.840.113556.1.4.1941. This operation can be requested by including the OID in the search filter.

Note: Some OpenLDAP and freeIPA servers might support LDAP_MATCHING_RULE_IN_CHAIN, but this is not enabled by default. If enabled on the server side, the same syntax described below for AD may be used with other LDAP server types.

User Search

Include LDAP_MATCHING_RULE_IN_CHAIN extension for memberOf attribute in a User Search Filter (for LDAP authentication). Match only those users who are members of a specific group (i.e. group B):

```
"(&(sAMAccountName={0})(memberOf:1.2.840.113556.1.4.1941:=CN=gro
upB,ou=Groups,DC=Example,DC=COM))"
```

This form, when used in a User Search Filter, only allows members of group B, including nested group A members to authenticate.

Group Search

Include LDAP_MATCHING_RULE_IN_CHAIN extension for member attribute in a Group Search Filter (for LDAP authorization), to retrieve all groups a user is a member of, including nested groups.

```
"(&(objectCategory=Group)(member:1.2.840.113556.1.4.1941:={0}))"
```

This form may be used with services that support the user DN variable token "{0}" (for example, Cloudera Manager) in the LDAP Group Search Filter. If the user is a member of group A, then the group list returned includes both group A and group B.

Related Information

Configuring nested group hierarchies

How to Check Security Settings on a Cluster

Information on how to quickly perform a high level check of your cluster’s security configuration.

Check Security for Cloudera Manager Clusters

Use Cloudera Manager to verify security mechanisms for your cluster by simply examining the properties for the cluster.

To check Kerberos and HDFS encryption:

1. Log into the Cloudera Manager Admin Console.

2. Select Security from the Administration drop-down selector to display a list of managed clusters:

This shows at a glance that both Kerberos and HDFS transparent encryption have been configured for this cluster.

To check TLS settings:

1. Select Settings from the Administration drop-down selector to open a search field.
2. Enter TLS in the search field to display all TLS related configuration settings.
3. Scroll through the displayed results, looking for “Use TLS...” for various services and processes. For example, the test system shown below is not using TLS for the Cloudera Manager Admin Console:

See “How to Configure TLS Encryption for Cloudera Manager” for complete information about configuring TLS for the cluster.

To find all TLS settings, cluster-wide, enter "TLS enabled" (or simply, "TLS") in the top-most search field on the Cloudera Manager Admin Console. Then you can easily select from among the display list to examine the actual setting.

How to Use Antivirus Software on Cloudera Hosts

If you use antivirus software on your servers, consider configuring it to skip scans on certain types of Hadoop-specific resources.

It can take a long time to scan large files or directories with a large number of files. In addition, if your antivirus software locks files or directories as it scans them, those resources will be unavailable to your Hadoop processes during the scan, and can cause latency or unavailability of resources in your cluster. Consider skipping scans on the following types of resources:

- Scratch directories used by services such as Impala, and Data directories, which can grow to petabytes in size, including Kudu directories on disk.

  To find the local data directories and files:

  1. Go to Cloudera Manager.
  2. On the Home Status tab, click a cluster name.
  3. Select Configuration Local Data Directories and Files .
  4. Find the link of the page.

     For example, https://CM_SERVER:port/cmf/clusters/#/config2?task=ALL_LOCAL_DATA_DIRS_AND_FILES
- Log directories used by various Hadoop services.

  To find log directories:

  1. Go to Cloudera Manager.
  2. On the Home Status tab, click a cluster name.
  3. Select Configuration Log Directories .
  4. Find the link of the page.

     For example, https://CM_SERVER:port/cmf/clusters/#/config2?task=ALL_LOG_DIRECTORIES

Important: For details on parcel locations, see Parcel Locations.

The specific directory names and locations depend on the services your cluster uses and your configuration. In general, avoid scanning very large directories and filesystems. Instead, limit write access to these locations using security mechanisms such as access controls at the level of the operating system, HDFS, or at the service level.

How to Configure Browser-based Interfaces to Require Kerberos Authentication

Access to the web (browser-based) consoles for HTTP services, such as HDFS, MapReduce, and YARN roles can be restricted to only those users with valid Kerberos credentials.

Required Role: Configurator, Cluster Administrator, or Full Administrator

After configuring the cluster services as detailed below, the web service and the browser negotiate the authentication process using SPNEGO. After following the steps detailed below for any specific service, all browsers that connect to the service over HTTP must have their configurations modified to handle SPNEGO. See “How to Configure Browsers for Kerberos Authentication”.

To require authentication for HTTP services on the cluster, including web-based user interfaces, such as the Cloudera Manager Admin Console, among others:

1. From the Clusters tab, select the service (HDFS, MapReduce, or YARN) for which you want to enable authentication.
2. Click the Configuration tab.
3. Under the Scope filter, click *SERVICE_NAME* (Service-Wide).
4. Under the Category filter, click Security to display the security configuration options.
5. In the Enable Kerberos Authentication for HTTP Web-Consoles setting, click the box to activate authentication requirement for the selected *SERVICE_NAME* (Service-Wide).
6. Click Save Changes to save the change.

Cloudera Manager generates new credentials for the service. When the process finishes, restart all roles for that service.

Repeat this process for the other services for which you want to require Kerberos authentication.

Configure your browser to support authentication to the HTTP services by following the appropriate steps for your browser in “How to Configure Browsers for Kerberos Authentication”.

How to Configure Browsers for Kerberos Authentication

How to enable specific web browsers to use SPNEGO to negotiate Kerberos authentication.

The browser configurations below are required only for those browsers used to connect to component web interfaces with the “Require Authentication for HTTP Web Consoles” configuration property enabled. The settings below enable the respective browser to use SPNEGO to negotiate Kerberos authentication for the browser. The host running the browser must have a valid TGT to authenticate to Kerberos Web Consoles.

Note: Impala users who see the following Alert message: "Bad : The Cloudera Manager Agent is not able to communicate with this role's web server." must disable webserver_htpassword_user and webserver_htpass word_password for Catalog Server, Statestore, and the Impala Daemon.

Mozilla Firefox

1. Open the low level Firefox configuration page by loading the about:config page.
2. In the Search text box, enter: network.negotiate-auth.trusted-uris
3. Double-click the network.negotiate-auth.trusted-uris preference and enter the hostname or the domain of the web server that is protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with a comma.
4. Click OK.
5. In the Search text box, enter: network.auth.use-sspi
6. If the value of the network.auth.use-sspi preference is currently true, double-click it to change it to false.

Internet Explorer

Follow the steps below to configure Internet Explorer.

Configuring the Local Intranet Domain

1. Open Internet Explorer and click the Settings gear icon in the top-right corner. Select Internet options.
2. Select the Security tab.
3. Select the Local Intranet zone and click the Sites button.
4. Make sure that the first two options, Include all local (intranet) sites not listed in other zones and Include all sites that bypass the proxy server are checked.
5. Click Advanced and add the names of the domains that are protected by Kerberos HTTP SPNEGO, one at a time, to the list of websites. For example, myhost.example.com. Click Close.
6. Click OK to save your configuration changes.

Configuring Intranet Authentication

1. Click the Settings gear icon in the top-right corner. Select Internet options.
2. Select the Security tab.
3. Select the Local Intranet zone and click the Custom level... button to open the Security Settings - Local Intranet Zone dialog box.
4. Scroll down to the User Authentication options and select Automatic logon only in Intranet zone.
5. Click OK to save these changes.

Verifying Proxy Settings

Perform these steps only if you have a proxy server already enabled.

1. Click the Settings gear icon in the top-right corner. Select Internet options.
2. Select the Connections tab and click LAN Settings.
3. Verify that the proxy server Address and Port number settings are correct.
4. Click Advanced to open the Proxy Settings dialog box.
5. Add the Kerberos-protected domains to the Exceptions field.
6. Click OK to save any changes.

Google Chrome

For Windows:

- Open the Control Panel to access the Internet Options dialog. Use the same configuration as detailed in Configuration changes required are the same as those described above for Internet Explorer.

For Linux or MacOS:

- Add the --auth-server-allowlist parameter to the google-chrome command. For example, to run Chrome from a Linux prompt, run the google-chrome command as follows:

  ```
  > google-chrome --auth-server-allowlist = "hostname/domain"
  ```

How to Configure Clusters to Use Kerberos for Authentication

Cloudera clusters can use Kerberos to authenticate services running on the cluster and the users who need access to those services. This How To guide provides the requirements, pre-requisites, and high-level summary of the steps needed to integrate clusters with Kerberos for authentication.

Note: For clusters deployed using Cloudera Manager Server, Cloudera recommends using the Kerberos configuration wizard available through the Cloudera Manager Admin Console. See “Enabling Kerberos Authentication Using the Wizard” for details.

The following are the general steps for integrating Kerberos with Cloudera clusters without using the Cloudera Manager configuration wizard.

Step 1: Verify Requirements and Assumptions

The steps outlined below assume that:

- The Kerberos instance has been setup, is running, and is available during the configuration process.
- The Cloudera cluster has been installed and is operational, with all services fully-functional—Cloudera Manager Server, Cloudera, and Cloudera Manager Agent processes on all cluster nodes.

Hosts Configured for AES-256 Encryption

By default, CentOS and RHEL 6 (and higher) use AES-256 encryption for Kerberos tickets. If you use either of these platforms for your cluster, the “Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File” must be installed on all cluster hosts.

To install the JCE Policy file on the host system at the OS layer:

1. Download the jce_policy-x.zip
2. Unzip the file
3. Follow the steps in the README.txt to install it.

Note: The AES-256 encryption can also be configured on a running cluster by using Cloudera Manager Admin Console. See “To use Cloudera Manager to install the JCE policy file” for details.

Required Administrator Privileges

Setting up the Cloudera cluster to use Kerberos for authentication requires complete administrator access to the cluster and administrator privileges on the Kerberos instance:

- Cluster Administrator or Full Administrator
- Kerberos administrator privileges:

```
SOMEONE/admin@YOUR-DOMAIN.FQDN.EXAMPLE.COM
```

If you do not have administrator privileges on the Kerberos instance, you will need help from the Kerberos administrator before you can complete the process.

Required User (Service Account) Directories

During installation, the cloudera-scm account is created on the host system. When Cloudera Manager and Cloudera services are installed at the same time, Cloudera Manager creates other accounts as needed to support the service role daemons. However, if the Cloudera services and Cloudera Manager are installed separately, you may need to specifically set directory permissions for certain Hadoop user (service daemon) accounts for successful integration with Kerberos. The following table shows the accounts used for core service roles. Note that hdfs acts as superuser for the system.

User	Service Roles
hdfs	NameNode, DataNodes, Secondary NameNode (and HDFS superuser)
mapred	JobTracker, TaskTrackers (MR1), Job History Server (YARN)
yarn	ResourceManager, NodeManager (YARN)

User	Service Roles
oozie	Oozie Server
hue	Hue Server, Beeswax Server, Authorization Manager, Job Designer

These accounts require ownership control over specific directories.

- For newly installed Cloudera clusters (Cloudera Manager and Cloudera installed at the same time)—The Cloudera Manager Agent process on each cluster host automatically configures the appropriate directory ownership when the cluster launches.
- For existing Cloudera clusters using HDFS and running MapReduce jobs prior to Cloudera Manager installation—The directory ownership must be manually configured, as shown in the table below. The directory owners cannot differ from those shown in the table to ensure that the service daemons can set permissions as needed on each directory.

Directory Specified in this Property	Owner
dfs.namenode.name.dir	hdfs:hadoop
dfs.data.dir	hdfs:hadoop
mapred.local.dir	mapred:hadoop
mapred.system.dir in HDFS	mapred:hadoop
yarn.nodemanager.local-dirs	yarn:yarn
yarn.nodemanager.log-dirs	yarn:yarn
oozie.service.StoreService.jdbc.url (if using Derby)	oozie:oozie
[[database]] name	hue:hue
javax.jdo.option.ConnectionURL	hue:hue

Step 2. Create Principal for Cloudera Manager Server in the Kerberos KDC

Cloudera Manager Server has its own principal to connect to the Kerberos KDC and import user and service principals for use by the cluster.

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.

Note: If an administrator principal to act on behalf of Cloudera Manager cannot be created on the Kerberos KDC for whatever reason, Cloudera Manager will not be able to create or manage principals and keytabs for Cloudera 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.

Creating a Principal in Active Directory

Check your Microsoft documentation for specific details for your Active Directory KDC.

The general process is as follows:

1. Create an Organizational Unit (OU) in your Active Directory KDC service that will contain the principals for use by the Cloudera cluster.
2. Add a new user account to Active Directory, for example, *USERNAME*@YOUR-REALM.EXAMPLE.COM. Set the password for the user to never expire.
3. Use the Delegate Control wizard of Active Directory and grant this new user permission to Create, Delete, and Manage User Accounts.

Creating a Principal in an MIT KDC

For MIT Kerberos, user principals that include the instance name admin designate a user account with administrator privileges. For example:

```
USERNAME/admin@YOUR-REALM.EXAMPLE.COM
```

Create the Cloudera Manager Server principal as shown in one of the examples below, appropriate for the location of the Kerberos instance and using the correct REALM name for your setup.

For MIT Kerberos KDC on a remote host:

```
kadmin: addprinc -pw PASSWORD cloudera-scm/admin@YOUR-REALM.EXAMPLE.COM
```

For MIT Kerberos KDC on a local host:

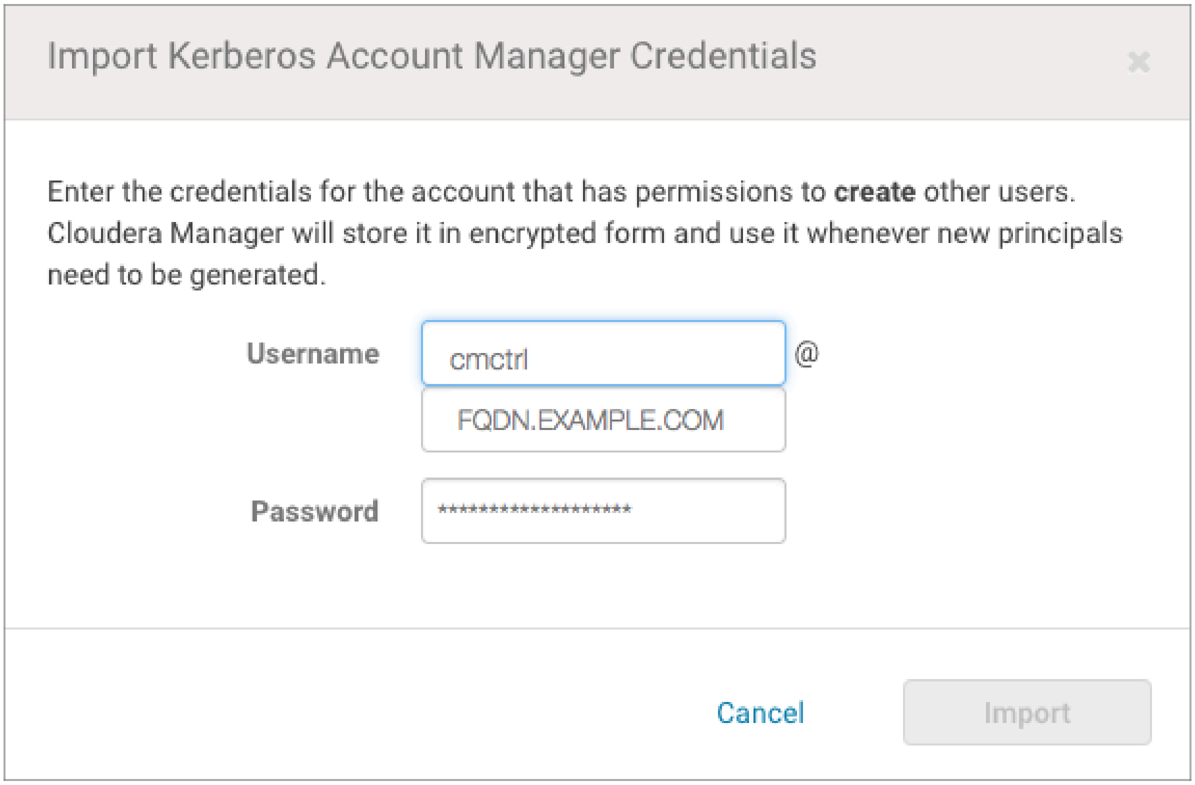
```
kadmin.local: addprinc -pw PASSWORD cloudera-scm/admin@YOUR-REALM.EXAMPLE.C
OM
```

Step 3: Add the Credentials for the Principal to the Cluster

Assuming the principal was successfully added to the Kerberos KDC, it can be added to the cluster as follows:

1. Log in to the Cloudera Manager Admin Console.
2. Select AdministrationSecurity.
3. Click the Kerberos Credentials tab.
4. Click the Import Kerberos Account Manager Credentials button.

5. Enter the credentials for the principal added to the “Kerberos KDC in the previous step”:
   - For Username, enter the primary and realm portions of the Kerberos principal. Enter the realm name in all upper-case only (*YOUR-REALM.EXAMPLE.COM*) as shown below.
   - Enter the Password for the principal.



Import Kerberos Account Manager Credentials

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

Username: cmctrl @ FQDN.EXAMPLE.COM

Password: ********************

6. Click Import.

Cloudera Manager encrypts the username and password into a keytab and uses it to create new principals in the KDC as needed.

Click Close when complete.

Step 4: Identify Default Kerberos Realm for the Cluster

Each host in the cluster must have the default realm property (default_realm) specified in the libdefaults section of its Kerberos configuration file (/etc/krb5.conf).

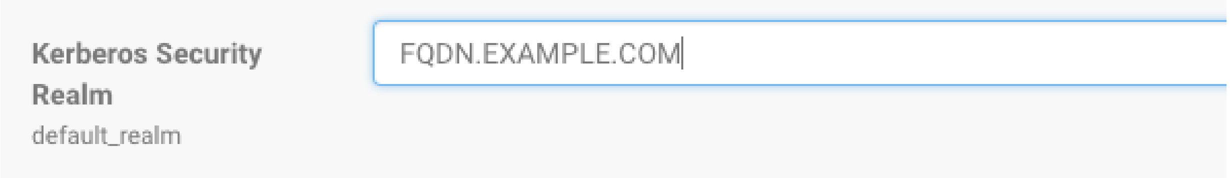
```
[libdefaults]
default_realm = FQDN.EXAMPLE.COM
```

After adding the default realm to the configuration file for all hosts in the cluster, configure the same default realm for Cloudera Manager Server.

In the Cloudera Manager Admin Console:

1. Select AdministrationSettings.
2. Select Kerberos for the Category filter.

3. In the Kerberos Security Realm field, enter the default realm name set in the Kerberos configuration file (/etc/krb 5.conf) of each host in the cluster. For example:



Kerberos Security Realm
default_realm
FQDN.EXAMPLE.COM

4. Click Save Changes.

Step 5: Stop all Services

All service daemons in the cluster must be stopped so that they can be restarted at the same time and start as authenticated services in the cluster. Service daemons running without authenticating to Kerberos first will not be able to communicate with other daemons in the cluster that have authenticated to Kerberos, so they must be shut down and restarted at the end of the configuration process, as a unit.

Note: The requirement to stop all daemons prevents using the rolling upgrade process to enable Kerberos integration on the cluster.

Stop all running services and the Cloudera Management Service as follows:

In the Cloudera Manager Admin Console:

1. Select ClustersCluster-*N*.
2. Click the Actions drop-down menu and select Stop to stop all services on the cluster.
3. Click Stop on the warning message to stop all services on the cluster. The Command Details window displays the progress as each service shuts down. When the message All services successfully stopped displays, close the Command Details window.
4. Select Clusters Cloudera Management Service .
5. Click the Actions drop-down menu and select Stop to stop the Cloudera Management Service. The Command Details window displays the progress as each role instance running on the Cloudera Management Service shuts down. The process is completed when the message Command completed with n/n successful subcommands displays.
6. Click Close.

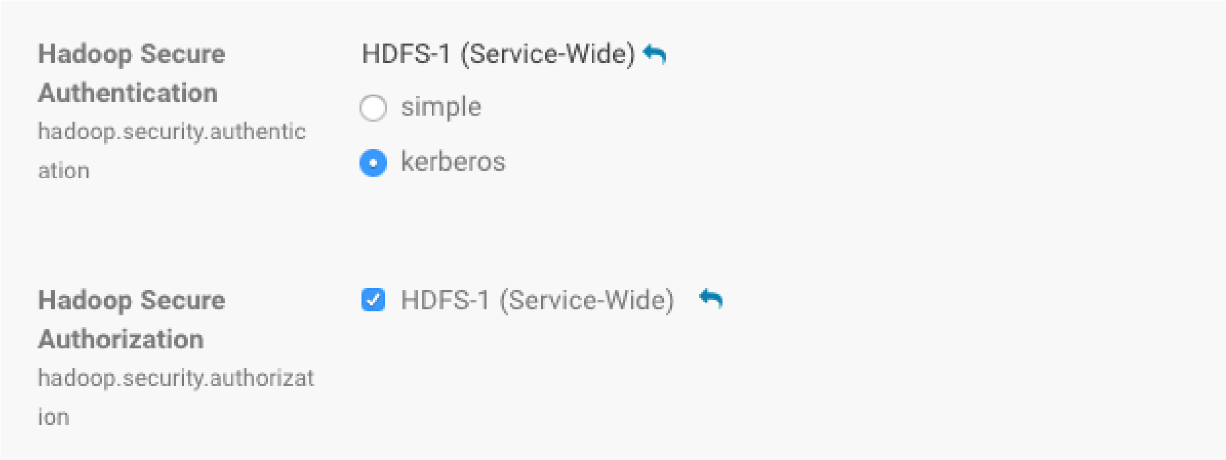
Step 6. Specify Kerberos for Security

Kerberos must be specified as the security mechanism for Hadoop infrastructure, starting with the HDFS service. Enable Cloudera Manager Server security for the cluster on an HDFS service. After you do so, the Cloudera Manager Server automatically enables Hadoop security on the MapReduce and YARN services associated with that HDFS service.

In the Cloudera Manager Admin Console:

1. Select ClustersHDFS-*N*.
2. Click the Configuration tab.
3. Select HDFS-*N* for the Scope filter.
4. Select Security for the Category filter.
5. Scroll (or search) to find the Hadoop Secure Authentication property.

6. Click the kerberos button to select Kerberos:



Hadoop Secure Authentication
hadoop.security.authentication

HDFS-1 (Service-Wide)

○ simple

● kerberos

Hadoop Secure Authorization
hadoop.security.authorization

☑ HDFS-1 (Service-Wide)

7. Click the value for the Hadoop Secure Authorization property and select the checkbox to enable service-level authorization on the selected HDFS service. You can specify comma-separated lists of users and groups authorized to use Hadoop services or perform admin operations using the following properties under the Service-Wide Security section:

- Authorized Users: Comma-separated list of users authorized to use Hadoop services.
- Authorized Groups: Comma-separated list of groups authorized to use Hadoop services.
- Authorized Admin Users: Comma-separated list of users authorized to perform admin operations on Hadoop.
- Authorized Admin Groups: Comma-separated list of groups authorized to perform admin operations on Hadoop.

Important: For Cloudera Manager's Monitoring services to work, the hue user should always be added as an authorized user.

8. In the Search field, type DataNode Transceiver to find the DataNode Transceiver Port property.
9. Click the value for the DataNode Transceiver Port property and specify a privileged port number (below 1024). Cloudera recommends 1004.

Note: If there is more than one DataNode Role Group, you must specify a privileged port number for each DataNode Transceiver Port property.

10. In the Search field, type DataNode HTTP to find the DataNode HTTP Web UI Port property and specify a privileged port number (below 1024). Cloudera recommends 1006.

Note: The port numbers for the two DataNode properties must be below 1024 to provide part of the security mechanism to make it impossible for a user to run a MapReduce task that impersonates a DataNode. The port numbers for the NameNode and Secondary NameNode can be anything you want, but the default port numbers are good ones to use.

11. In the Search field type Data Directory Permissions to find the DataNode Data Directory Permissions property.
12. Reset the value for the DataNode Data Directory Permissions property to the default value of 700 if not already set to that.
13. Make sure you have changed the DataNode Transceiver Port, DataNode Data Directory Permissions and DataNode HTTP Web UI Port properties for every DataNode role group.
14. Click Save Changes to save the configuration settings.

To enable ZooKeeper security:

1. Go to the ZooKeeper Service Configuration tab and click View and Edit.
2. Click the value for Enable Kerberos Authentication property.
3. Click Save Changes to save the configuration settings.

To enable HBase security:

1. Go to the HBase ServiceConfiguration tab and click View and Edit.

2. In the Search field, type HBase Secure to show the Hadoop security properties (found under the Service-WideSecurity category).
3. Click the value for the HBase Secure Authorization property and select the checkbox to enable authorization on the selected HBase service.
4. Click the value for the HBase Secure Authentication property and select kerberos to enable authorization on the selected HBase service.
5. Click Save Changes to save the configuration settings.

To enable Solr security:

1. Go to the Solr ServiceConfiguration tab and click View and Edit.
2. In the Search field, type Solr Secure to show the Solr security properties (found under the Service-WideSecurity category).
3. Click the value for the Solr Secure Authentication property and select kerberos to enable authorization on the selected Solr service.
4. Click Save Changes to save the configuration settings.

Note: Using Cloudera Manager Admin Console to generate client configuration files after enabling Kerberos does not provide the Kerberos principals and keytabs that users need to authenticate to the cluster. Users must obtain their Kerberos principals from the Kerberos administrator and then run the kinit command themselves.

Credentials Generated

After you enable security for any of the services in Cloudera Manager, a command called Generate Credentials will be triggered automatically. You can watch the progress of the command on the top right corner of the screen that shows the running commands. Wait for this command to finish (indicated by a grey box containing "0" in it).

Step 7: Restart All Services

Start all services on the cluster using the Cloudera Manager Admin Console:

1. Select Clusters Cluster-*N*.
2. Click the Actions drop-down button menu and select Start. The confirmation prompt displays.
3. Click Start to confirm and continue. The Command Details window displays progress. When All services successfully started displays, close the Command Detailswindow.
4. Select Clusters Cloudera Management Service .
5. Click the Actions drop-down menu and select Start. The Command Details window displays the progress as each role instance running on the Cloudera Management Service starts up. The process is completed when the message Command completed with *N*/*N* successful subcommands displays.

Step 8: Deploy Client Configurations

Deploy client configurations for services supported on the cluster using the Cloudera Manager Admin Console:

1. Select Clusters Cluster-*N*.
2. Click the Actions drop-down button menu and select Deploy Client Configuration.

Step 9: Create the HDFS Superuser Principal

To be able to create home directories for users, you will need access to the HDFS superuser account. (Cloudera automatically created the HDFS superuser account on each cluster host during Cloudera installation.) When you enabled Kerberos for the HDFS service, you lost access to the default HDFS superuser account using sudo -u hdfs commands. Cloudera recommends you use a different user account as the superuser, not the default hdfs account.

Designating a Non-Default Superuser Group

To designate a different group of superusers instead of using the default hdfs account, follow these steps:

1. Go to the Cloudera Manager Admin Console and navigate to the HDFS service.
2. Click the Configuration tab.
3. Select ScopeHDFS (Service-Wide).
4. Select CategorySecurity.
5. Locate the Superuser Group property and change the value to the appropriate group name for your environment. For example, *SUPERUSER*.
6. Click Save Changes.
7. Restart the HDFS service.

   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*:

For Active Directory

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

For 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-REALM.EXAMPLE.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-REALM.EXAMPLE.COM
   ```

Step 10: Get or Create a Kerberos Principal for Each User Account

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 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.

To create Kerberos principals for all users:

Active Directory

Add a new AD user account, <username>@EXAMPLE.COM for each Cloudera Manager service that should use Kerberos authentication. The password for these service accounts should be set to never expire.

MIT KDC

1. In the kadmin.local or kadmin shell, use the following command to create a principal for your account by replacing EXAMPLE.COM with the name of your realm, and replacing username with a username:

   ```
   kadmin:  addprinc username@EXAMPLE.COM
   ```
2. When prompted, enter the password twice.

Step 11: Prepare the Cluster for Each User

Before you and other business users can access the cluster, the hosts must be prepared for each user. Perform the following tasks to prepare the hosts for each user:

1. Each host in the cluster must have a Unix user account with the same name as primary component of the user's principal name. For example, the principal jcarlos@YOUR-REALM.EXAMPLE.COM needs the Linux account jcarlos on each host system. Use LDAP (OpenLDAP, Microsoft Active Directory) for this step if possible.

   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 from 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/jcarlos). Change the owner and group of that directory to be the user.

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

Note: The commands above do not include sudo -u hdfs because it is not required with Kerberos configured for the cluster (assuming you created the Kerberos credentials for the HDFS super user as detailed in “Step 9: Create the HDFS Superuser Principal”).

Step 12: Verify Successful Kerberos Integration

To verify that Kerberos has been successfully integrated for the cluster, try running one of the sample MapReduce jobs (sleep or pi, for example) provided at:

```
/usr/lib/hadoop/hadoop-examples.jar
```

This assumes you have the fully-functional cluster as recommended in “Step 1: Verify Requirements and Assumptions” and that the client configure files have been generated as detailed in “Step 8: Deploying Client Configuration Files”.

To verify that Kerberos security is working:

1. Obtain Kerberos credentials for your user account.

   ```
   $ kinit
               YOURUSERID@YOUR-REALM.EXAMPLE.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/hadoop-0.20.2*examples.jar pi 10 10000
   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
```

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)]
```

How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys

Client and server processes require specific file formats for certificates, keys, and other digital artifacts used for TLS/SSL encryption. For example, when TLS is enabled, Cloudera Manager Server presents Java KeyStore (JKS) formatted key and certificate to requesting Cloudera Manager Agent hosts. The Data Explorer client also connects to Cloudera Manager Server, but Data Explorer requires a PEM-formatted key and certificate, rather than JKS. The PEM format used by Cloudera Manager is PKCS #8, which handles certificates and keys as individual Base64-encoded text files.

If you receive binary DER files from your certificate authority, you must convert them to the appropriate format. Since neither Java Keytool nor OpenSSL work directly with PKCS format, many of the configuration tasks detailed in “How to Configure TLS Encryption for Cloudera Manager ” involve converting formats, or extracting keys or certificates from an artifact in one format to another.

Certificates issued by a CA in one format (encoding) can be used to create certificates in a different format using Java Keytool and OpenSSL as detailed below.

Converting DER Encoded Certificates to PEM

OpenSSL can be used to convert a DER-encoded certificate to an ASCII (Base64) encoded certificate. Typically, DER-encoded certificates may have file extension of .DER, .CRT, or .CER, but regardless of the extension, a DER encoded certificate is not readable as plain text (unlike PEM encoded certificate).

A PEM-encoded certificate may also have file extension of .CRT or .CER, in which case, you can simply copy the file to a new name using the .PEM extension:

```
cp hostname.cer hostname.pem
```

To convert a DER-encoded certificate to PEM encoding using OpenSSL:

```
openssl x509 -inform der -in hostname.cer -out hostname.pem
```

For example:

```
openssl x509 -inform der -in /opt/cloudera/security/pki/hostname.cer -out /t
mp/hostname.pem
```

Converting JKS Key and Certificate to PEM

This process uses both Java keytool and OpenSSL (keytool and openssl, respectively, in the commands below) to export the composite private key and certificate from a Java keystore and then extract each element into its own file.

The PKCS12 file created below is an interim file used to obtain the individual key and certificate files.

Replace hostname-keystore, cmhost, hostname, and password with values from your system.

1. Export the private key and certificate command line:

```
$ keytool -importkeystore -srckeystore /opt/cloudera/security/jk
s/hostname-keystore.jks \
-srcstorepass password -srckeypass password -destkeystore /tmp/hostname-
keystore.p12 \
-deststoretype PKCS12 -srcalias hostname -deststorepass password -destke
ypass password
```

2. Extract the certificate file from the resulting PKCS12 file:

```
$ openssl pkcs12 -in /tmp/hostname-keystore.p12 -passin pass:password  -n
okeys \
-out /opt/cloudera/security/pki/hostname.pem
```

This extracted certificate can be used, as is, but you can additionally extract the private key from the keystore as detailed in the next section.

Extracting the Private Key from PKCS Keystore

Use OpenSSL to extract the private key from the PKCS keystore when needed.

The command shown below extracts the key and saves it to a keystore that is protected using the password you provide:

```
   $ openssl pkcs12 -in /tmp/hostname-keystore.p12 -passin pass:password \
   -nocerts -out /opt/cloudera/security/pki/hostname.key -passout pa
ss:password
```

To generate a key without a password, use this version of the command:

```
  $ openssl rsa -in /tmp/hostname-keystore.p12 -passin pass:password \
   -nocerts -out /opt/cloudera/security/pki/hostname.pem
```

Converting PEM Key and Certificate to JKS

Replace hostname in the commands below with the FQDN of the host whose certificate is being imported.

1. Convert the openssl private key and certificate files into a PKCS12 file.

```
$ openssl pkcs12 -export -in /opt/cloudera/security/pki/hostname.pem \
-inkey /opt/cloudera/security/pki/hostname.key -out /tmp/hostname.p12 \
-name hostname -passin pass:password -passout pass:password
```

2. Import the PKCS12 file into the Java keystore.

```

$ keytool -importkeystore -srckeystore /tmp/hostname.p12 -srcstoretype P
KCS12 \
-srcstorepass password -alias hostname -deststorepass password \
-destkeypass password -destkeystore /opt/cloudera/security/jks/hostname-
keystore.jks
```

How To Configure Authentication for Amazon S3

There are several ways to integrate Amazon S3 storage with Cloudera clusters, depending on your use case and other factors, including whether the cluster has been deployed using Amazon EC2 instances and if those instances were deployed using an IAM role, such as might be the case for clusters that have a single-user or small-team with comparable privileges.

Clusters deployed to support many different users with various privilege levels to the Amazon S3 need to use AWS Credentials and have privileges to target data set up in Ranger.

Authentication through the S3 Connector Service

With Cloudera Manager and CDH 5.10+ clusters, the S3 Connector Service automates the authentication process to Amazon S3 for Impala, Hive, and Data Explorer, the components used for business-analytical use cases designed to run on persistent multi-tenant clusters.

The S3 Connector Service transparently and securely distributes AWS credentials needed by the cluster for the Amazon S3 storage. Access to the underlying Impala tables is controlled by Ranger role-based permissions. The S3 Connector Service runs on a secure cluster only, that is, a cluster configured to use:

- Kerberos for authentication, and
- Ranger for role-based authorization.

Note: Only Ranger service and Kerberos-enabled messages are actual requirements. Other messages are for informational purposes only.

The S3 Connector Service setup wizard is launched automatically in Cloudera Manager during the AWS Credential setup process when you select the path to add the S3 Connector Service.

See “Configuring the Amazon S3 Connector” for more information about the S3 Connector Service.

Authentication through Advanced Configuration Snippets

Before release 5.10 and the introduction of the S3 Connector Service, using Amazon S3 storage with the cluster involved adding the credentials to the core-site.xml configuration file (through Cloudera Manager's Advanced Configuration Snippet mechanism). This approach is not recommended. AWS credentials provide read and write access to data stored on Amazon S3, so they should be kept secure at all times.

- Never share the credentials with other cluster users or services.
- Do not store in cleartext in any configuration files. When possible, use Hadoop's credential provider to encrypt and store credentials in the local JCEK (Java Cryptography Extension Keystore).

- Enable Cloudera sensitive data redaction to ensure that passwords and other sensitive information does not appear in log files.

Important: Cloudera recommends using this approach for single-user clusters on secure networks only—networks that allow access only to authorized users, all of whom are also authorized to use the S3 credentials.

To enable Cloudera services to access Amazon S3, AWS credentials can be specified using the fs.s3a.access.key and fs.s3a.secret.key properties:

```
<property>
    <name>fs.s3a.access.key</name>
    <value>YOUR_ACCESS_KEY</value>
</property>

<property>
    <name>fs.s3a.secret.key</name>
    <value>YOUR_SECRET_KEY</value>
</property>
```

The process of adding AWS credentials is generally the same as that detailed in “Configuring server-side encryption for Amazon S3”, that is, using the Cloudera Manager Admin Console to add the properties and values to the core-sit e.xml configuration file (Advanced Configuration Snippet). However, Cloudera strongly discourages this approach: in general, adding AWS credentials to the core-site.xml is not recommended. A somewhat more secure approach is to use temporary credentials, which include a session token that limits the viability of the credentials to a shorter timeframe within which a key can be stolen and used.

Using Temporary Credentials for Amazon S3

The AWS Security Token Service (STS) issues temporary credentials to access AWS services such as Amazon S3. These temporary credentials include an access key, a secret key, and a session token that expires within a configurable amount of time. Temporary credentials are not currently handled transparently by Cloudera, so administrators must obtain them directly from Amazon STS. For details, see “Temporary Security Credentials” in the “AWS Identity and Access Management Guide”.

Important: Cloudera recommends using this approach only for single-user clusters on secure networks—networks that allow access only to authorized users, all of whom are also authorized to use the S3 credentials.

To connect to Amazon S3 using temporary credentials obtained from STS, submit them as command-line arguments with the Hadoop job. For example:

```
-Dfs.s3a.access.key=YOUR_TEMP_ACCESS_KEY
-Dfs.s3a.secret.key=YOUR_TEMP_SECRET_KEY
-Dfs.s3a.session.token=YOUR_SESSION_TOKEN_FROM_AMAZONSTS
-Dfs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.TemporaryAWSCred
entialsProvider
```

Creating a Table in a Bucket

To create a table in a bucket, a user must have Ranger permissions on the S3 database and bucket URI. Cloudera recommends that you create a database specifically for the purpose of creating tables. Then, you must grant the user's role one of the following options:

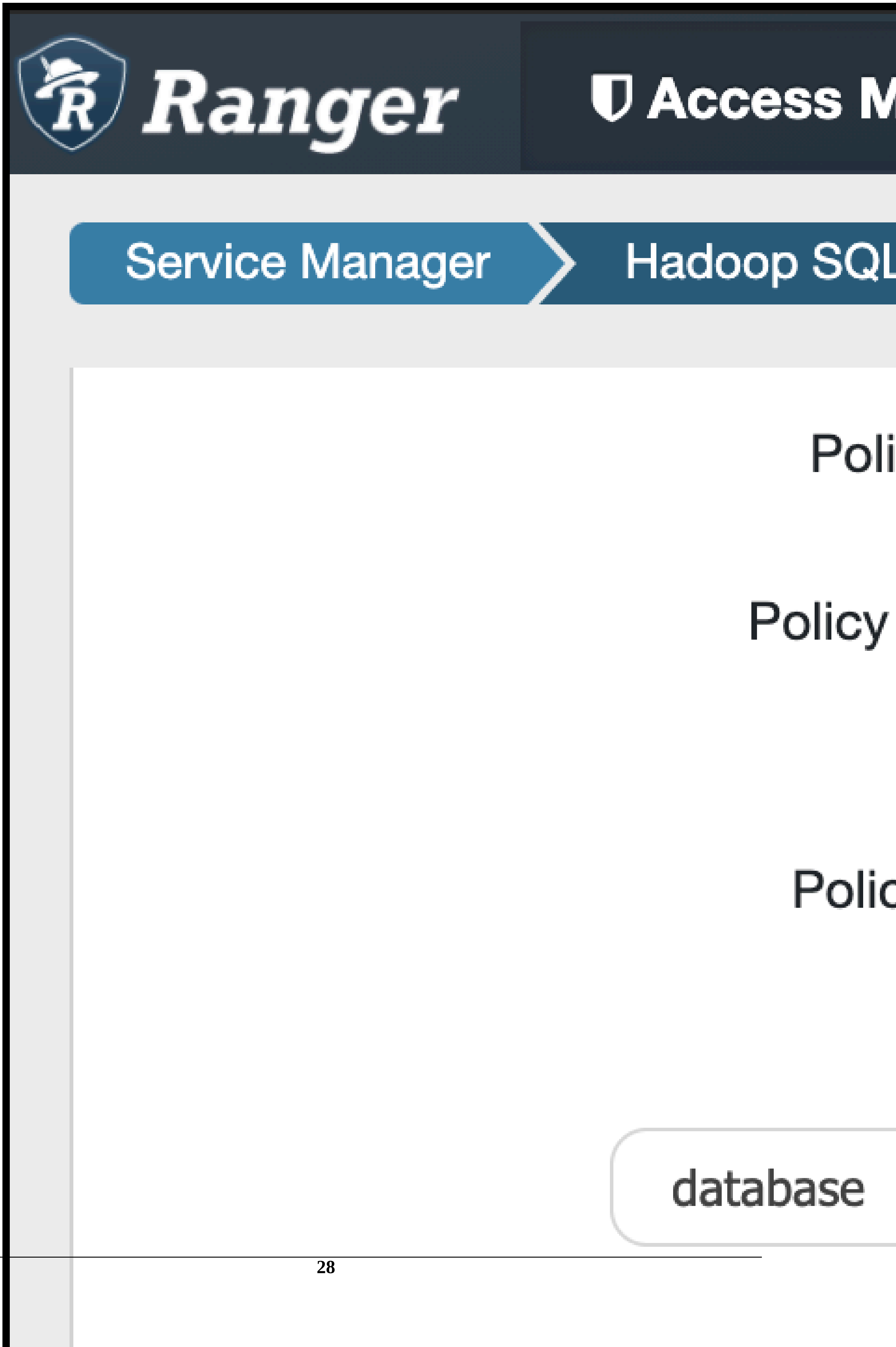
- CREATE on the server and ALL on the URI
- CREATE on the database and ALL on the URI

Note: It is possible to give the user ALL permissions on the server to allow the user to create external tables, but that approach is NOT recommended because it is not secure.

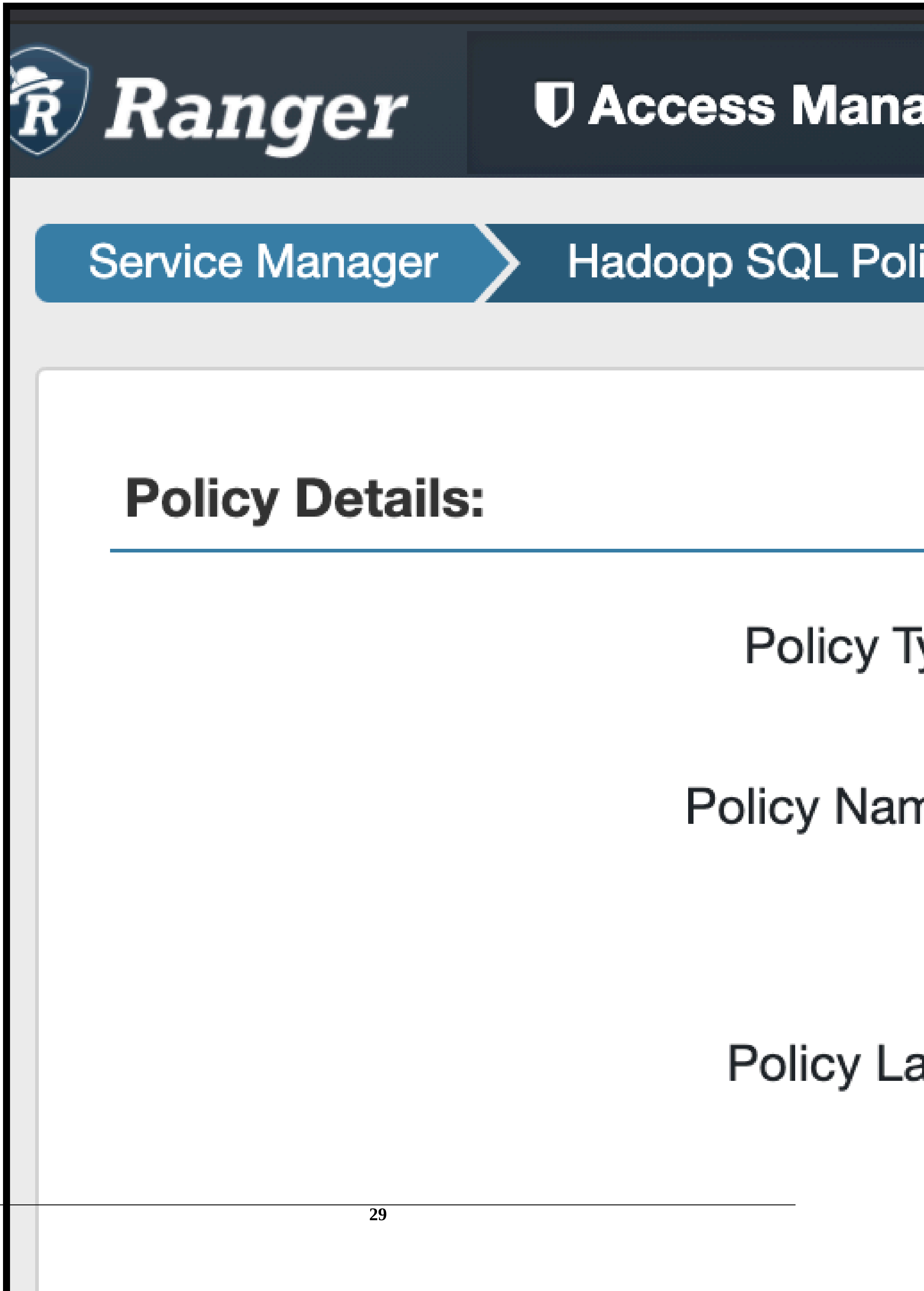
To allow a user to create tables in a bucket, complete the following steps:

1. Create a database where you want to create the tables.

2. Grant CREATE on the database to the user's role, as shown in the following



3. Grant ALL on the bucket URI to the user's role, as shown in the following example:



4. The user must create the table with a reference to the database. For example:

```
CREATE EXTERNAL TABLE
rangers3demo.your_external_s3_table
(name STRING, type STRING) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3a://your-bucket-name/';
```

How to Configure Encryption for Amazon S3

Amazon offers several server-side encryption mechanisms for use with Amazon S3 storage. Cloudera clusters support server-side encryption for Amazon S3 data using either SSE-S3 or SSE-KMS.

Amazon offers several server-side encryption mechanisms for use with Amazon S3 storage. Cloudera clusters support server-side encryption for Amazon S3 data using either SSE-S3 or SSE-KMS. With SSE-S3, keys are completely under the control of Amazon. With SSE-KMS, you have more control over the encryption keys, and can upload your own key material to use for encrypting Amazon S3. With either mechanism, encryption is applied transparently to the Amazon S3 bucket objects after you configure your cluster to use it.

Amazon S3 also supports TLS/SSL ('wire' or data-in-transit) encryption by default. Configuring data-at-rest encryption for Amazon S3 for use with your cluster involves some configuration both on Amazon S3 and for the cluster, using the Cloudera Manager Admin Console, as detailed below:

Requirements

Using Amazon S3 assumes that you have an Amazon Web Services account and the appropriate privileges on the AWS Management Console to set up and configure Amazon S3 buckets.

In addition, to configure Amazon S3 storage for use with a Cloudera cluster, you must have privileges as the User Administrator or Full Administrator on the Cloudera Manager Admin Console. See “How to Configure AWS Credentials” for details.

Amazon S3 and TLS/SSL Encryption

Amazon S3 uses TLS/SSL by default. Cloudera clusters (release 5.9 and later) include in their default configuration file the boolean property, fs.s3a.connection.ssl.enabled set to true, which activates TLS/SSL. This means that if the cluster has been configured to use TLS/SSL, connections from the cluster to Amazon S3 automatically use TLS wire encryption for the communication. The value of the fs.s3a.connection.ssl.enabled property can be confirmed by running hadoop org.apache.hadoop.conf.Configuration.

⚠ **Important:** Cloudera recommends that clusters always be configured for TLS/SSL. See “How to Configure TLS Encryption for Cloudera Manager” for details.

If the cluster is not configured to use TLS, the connection to Amazon S3 silently reverts to an unencrypted connection.

Amazon S3 and Data at Rest Encryption

Cloudera clusters can use either of these two Amazon S3 mechanisms for data-at-rest encryption:

- Server-side Encryption with AWS KMS-Managed Keys (SSE-KMS), which requires using Amazon Key Management Server (AWS KMS) in conjunction with your Amazon S3. You can have Amazon generate and manage the keys in AWS KMS for you, or you can provide your own key material, but you must configure AWS KMS and create a key before you can use it with your cluster. See “Prerequisites for Using SSE-KMS” for details.

- Server-side Encryption with S3-Managed Encryption Keys (SSE-S3), which is simplest to set up because it uses Amazon-provided and -managed keys and has no requirements beyond setting a single property. See “Configuring the Cluster to Use Server-Side Encryption on Amazon S3” for details.

Enabling the cluster to use Amazon S3 server-side encryption involves using the Cloudera Manager Admin Console to configure the Advanced Configuration Snippet (Safety Valve) as detailed in “Configuring the Cluster to Use Server-Side Encryption on Amazon S3”, below.

The steps assume that your cluster has been set up and that you have set up AWS credentials.

Note: Cloudera clusters can be configured one encryption mode and key at a time for all objects in a given Amazon S3 bucket. However, you can change encryption modes or keys. See “Changing Encryption Modes or Keys” for details.

Prerequisites for Using SSE-KMS

To use SSE-KMS with your Amazon S3 bucket, you must log in to the AWS Management Console using the account you set up in Step 1 of “Getting Started with Amazon Web Services”. For example, the account lab-iam has an IAM user named etl-workload that has been granted permissions on the Amazon S3 storage bucket to be configured using SSE-KMS.

- Select My Security Credentials from the menu.
- Click Encryption keys (bottom left-hand on the AWS Management Console that displays at step 1, above).
- Click the Create key button to start the 5-step key-creation wizard that leads you through entry pages for giving the key an alias and description; adding tags, defining administrator permissions to the key, and defining usage permissions. The last page of the wizard shows you the policy that will be applied to the key before creating the key.

Configuring the Cluster to Use Server-Side Encryption on Amazon S3

Follow the steps below to enable server-side encryption on Amazon S3. To use SSE-KMS encryption, you will need your KMS key ID at step 7. Using SSE-S3 has no pre-requisites—Amazon generates and manages the keys transparently.

To configure the cluster to encrypt data stored on Amazon S3:

1. Log into the Cloudera Manager Admin Console.
2. Select Clusters HDFS.
3. Click the Configuration tab.
4. Select ScopeHDFS (Service-Wide).
5. Select Category Advanced.
6. Locate the Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml property.
7. In the text field, define the properties and values appropriate for the type of encryption you want to use.

   a. To use SSE-S3 encryption:

   ```
   <property>
       <name>fs.s3a.server-side-encryption-algorithm</name>
       <value>AES256</value>
   </property>
   ```

   b. To use SSE-KMS encryption:

   ```
   <property>
       <name>fs.s3a.server-side-encryption-algorithm</name>
       <value>SSE-KMS</value>
   ```

```
</property>
<property>
    <name>fs.s3a.server-side-encryption-key</name>
    <value>YOUR_KMS_KEY_ID</value>
</property>
```

8. Click Save Changes.
9. Restart the HDFS service.

For the value of *YOUR_KMS_KEY_ID* (step 7b., above), you can use any of Amazon's four different key ID formats. Here are some examples:

Format	Example
Key ARN	arn:aws:kms:us-west-1:141229114088:key/c914b724-f191- 41df-934a-6147f6235983
Alias ARN	arn:aws:kms:us-west-1: 141229114088:key/c914b724-f191-41df-934a-6147f6235983: alias/awsCreatedMasterKey
Globally Unique Key ID	141229114088:key/c914b724-f191-41df-934a- 6147f6235983
Alias Name	alias/awsCreatedMasterKey

Changing Encryption Modes or Keys

Cloudera clusters can be configured to use only one type of server-side encryption for Amazon S3 data at a time.

However, Amazon S3 does support using different encryption keys for objects on an Amazon S3 bucket, which means that Amazon S3 continues to use whichever key was initially used to encrypt data (to decrypt on reads and re-encrypt on writes), even after a new mechanism and key exists. Your new key is used on new data written to the Amazon S3 bucket, while your ‘old’ key is used on any existing data that was encrypted with it. To summarize the behavior:

- Changing encryption mechanisms or keys on Amazon S3 has no effect on existing encrypted or unencrypted data.
- Data stored on Amazon S3 without encryption remains unencrypted even after you configure encryption for Amazon S3.
- Any existing encrypted data continues using the original mechanism and key to decrypt data (on reads) and re-encrypt data (on writes).
- After changing encryption mode or key, new objects stored on Amazon S3 from the cluster use the new mode and key.

Effect of Changing Encryption Mode or Key

This table shows the effect on existing encrypted or unencrypted data on Amazon S3 (far left column labeled "Data starts as...," reading down) and the result of "New" and "Existing" data and the keys that would be used after changing encryption-key configuration on the cluster. After changing encryption mode or key, existing data (Existing) and new data (New) use the mode and keys shown in columns 2 ("Unencrypted") through 5 ("Non-SSE Key"):

Data starts as...#	Data results after modifying encryption mode or keys...			
	Unencrypted	SSE-S3	SSE-KMS	Non-SSE Key
Unencrypted	Existing	New	New	~
SSE-S3 encrypted	~	Existing	New	~
SSE-KMS [key1]	~	New	Existing [key1] New [key2]	~
Non-SSE key	~	~	~	Existing

Migrating Encrypted Data to New Encryption Mode or Key

To use a new key with existing data (data stored on Amazon S3 using a different mechanism or key), you must first decrypt the data, and then re-encrypt it using the new key. The process is as follows:

1. Create an Amazon S3 bucket as temporary storage for the unencrypted files.
2. Decrypt the data on the Amazon S3 bucket using the mechanism and key used for encryption (legacy encryption mode or key), moving the unencrypted data to the temporary bucket created in step 1.
3. Configure the Amazon S3 bucket to use the new encryption mechanism and key of your choice (SSE-S3, SSE-KMS).
4. Move the unencrypted data from the temporary bucket back to the Amazon S3 bucket that is now configured using the new mechanism and key.

Deleting an Encryption Key

If you change encryption modes or keys on Amazon S3, do not delete the key. To replace the old key and mode with a completely new mode or key, you must manually migrate the data.

When you delete an encryption key, Amazon puts the key in a Pending Deletion state (as shown in the Status column of the screenshot below) for at least 7 days, allowing you to reinstate a key if you change your mind or realize an error.

The pending time frame is configurable, from 7 up to 30 days. See “AWS Key Management Service Documentation” for complete details.

How to Configure AWS Credentials

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

Amazon S3 (Simple Storage Service) can be used in a Cloudera cluster managed by Cloudera Manager in the following ways:

- As storage for Impala tables
- As a source or destination for HDFS and Hive/Impala replication and for cluster storage
- To enable Cloudera Navigator to extract metadata from Amazon S3 storage
- To browse S3 data using Data Explorer

To provide access to Amazon S3, you configure AWS Credentials that specify the authentication type (role-based, for example) and the access and secret keys. Amazon offers two types of authentication you can use with Amazon S3:

IAM Role-based Authentication

Amazon Identity and Access Management (IAM) can be used to create users, groups, and roles for use with Amazon Web Services, such as EC2 and Amazon S3. IAM role-based access provides the same level of access to all clients that use the role. All jobs on the cluster will have the same level of access to Amazon S3, so this is better suited for single-user clusters, or where all users of a cluster should have the same privileges to data in Amazon S3.

If you are setting up a peer to copy data to and from Amazon S3, using Cloudera Manager Hive or HDFS replication, select this option.

If you are configuring Amazon S3 access for a cluster deployed to Amazon Elastic Compute Cloud (EC2) instances using the IAM role for the EC2 instance profile, you do not need configure IAM role-based authentication for services such as Impala, Hive, or Spark.

Note: IAM role-based authentication does not provide access to Amazon S3 using Cloudera Navigator.

Access Key Credentials

This type of authentication requires an AWS Access Key and an AWS Secret key that you obtain from Amazon and is better suited for environments where you have multiple users or multi-tenancy.

You must enable Kerberos when using the S3 Connector service. Enabling these services allows you to configure selective access for different data paths.

Cloudera Manager stores these values securely and does not store them in world-readable locations. The credentials are masked in the Cloudera Manager Admin console, encrypted in the configurations passed to processes managed by Cloudera Manager, and redacted from the logs.

The client configuration files generated by Cloudera Manager based on configured services do not include AWS credentials. These clients must manage access to these credentials outside of Cloudera Manager. Cloudera Manager uses credentials stored in Cloudera Manager for trusted clients such as the Impala daemon and Data Explorer. For access from YARN, MapReduce or Spark, see Using S3 Credentials with YARN, MapReduce, or Spark.

Note: Isilon storage cannot be used with AWS credentials.

Adding AWS Credentials

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

To add AWS Credentials for Amazon S3:

1. Open the Cloudera Manager Admin Console.
2. Click AdministrationExternal Accounts.
3. Select the AWS Credentials tab.
4. Select one of the following:
   - Add Access Key Credentials

     This authentication mechanism requires you to obtain AWS credentials from Amazon.

     a. Enter a Name of your choosing for this account.
     b. Enter the AWS Access Key ID.
     c. Enter the AWS Secret Key.
   - Add IAM Role-Based Authentication

     a. Enter a name for your IAM Role-based authentication.

     Note: You cannot use IAM Role-based authentication for Cloudera Navigator access.
5. Click Add.

   The Edit S3Guard dialog box displays.

   S3Guard enables a consistent view of data stored in Amazon S3 and requires that you provision a DynamoDB database from Amazon Web Services. S3Guard is optional but can help improve performance and accuracy for certain types of workflows. To configure S3Guard, see Configuring and Managing S3Guard and return to these steps after completing the configuration.

   If you do not want to enable S3Guard, click Save to finish adding the AWS Credential.

   The Connect to Amazon Web Services dialog box displays.

6. Choose one of the following options:

   - Cloud Backup and Restore

     To configure Amazon S3 as the source or destination of a replication schedule (to back up and restore data, for example), click the Replication Schedules link.
   - Cluster Access to S3

     To enable cluster access to S3 using the S3 Connector Service, click the Enable for *CLUSTER NAME* link, which launches a wizard for adding the S3 Connector service.
   - Cloudera Navigator Access to S3

     To give Cloudera Navigator access to Amazon S3, click the Enable for Cloudera Navigator link. Restart the Cloudera Navigator Metadata Server to enable access.

     Note: You cannot enable Cloudera Navigator Access to S3 when using IAM Role-based Authentication.

Managing AWS Credentials

To remove AWS credentials:

Note: You cannot remove AWS credentials if they are in use by a service in the cluster. You might need to edit the connectivity of the credential before removing it.

1. Open the Cloudera Manager Admin Console.
2. Click AdministrationExternal Accounts.
3. Select the AWS Credentials tab.
4. Locate the row with the credentials you want to delete and click ActionsRemove.

To edit AWS Access Key credentials:

1. Open the Cloudera Manager Admin Console.
2. Click AdministrationExternal Accounts.
3. Select the AWS Credentials tab.
4. Locate the row with the Access Key Credentials you want to delete and click ActionsEdit Credential.

   The Edit Credential dialog box displays.
5. Edit the account fields.
6. Click Save.
7. Restart cluster services that use these credentials. If connectivity is for Cloudera Navigator, restart the Cloudera Navigator Metadata server.

To rename the IAM Role-Based Authentication:

1. Open the Cloudera Manager Admin Console.
2. Click AdministrationExternal Accounts.
3. Select the AWS Credentials tab.
4. Locate the row with the IAM Role-Based Authentication you want to rename and click ActionsRename.
5. Enter a new name.
6. Click Save.

   The Connect to Amazon Web Services screen displays.
7. Click the links to change any service connections or click Close to leave them unchanged.

To edit the services connected to an AWS Credentials account:

1. Open the Cloudera Manager Admin Console.
2. Click AdministrationExternal Accounts.

3. Select the AWS Credentials tab.
4. Locate the row with the credentials you want to edit and click ActionsEdit Connectivity.

   The Connect to Amazon Web Services screen displays.
5. Click one of the following options:

   - Choose one of the following options:

     - Cloud Backup and Restore
     - Cluster Access to S3
     - Cloudera Navigator Access to S3

How to Enable Sensitive Data Redaction

Redaction is a process that obscures data. It helps organizations comply with government and industry regulations, such as PCI (Payment Card Industry) and HIPAA, by making personally identifiable information (PII) unreadable except to those whose jobs require such access. For example, in simple terms, HIPAA legislation requires that patient PII is available only to appropriate medical professionals (and the patient), and that any medical or personal information exposed outside the appropriate context cannot be used to associate an individual's identity with any medical information. Data redaction can help ensure this privacy, by transforming PII to meaningless patterns—for example, transforming U.S. social security numbers to XXX-XX-XXXX strings.

Data redaction works separately from Cloudera data encryption techniques. Data encryption alone does not preclude administrators with full access to the cluster from viewing sensitive user data. Redaction ensures that cluster administrators, data analysts, and others cannot see PII or other sensitive data that is not within their job domain. At the same time, it does not prevent users with appropriate permissions from accessing data to which they have privileges.

Cloudera clusters implement some redaction features by default, while some features are configurable and require administrators to specifically enable them. The details are covered below:

Related Information

Set up and Configure the Cloudera Manager Database

Syntax for scm_prepare_database.sh

Disabling Redaction of sensitive information when using the Cloudera Manager API

Sensitive Data Redaction

Cloudera Manager and Passwords

Passwords are not in cleartext in the Cloudera Manager Admin Console or the configuration files on disk. Passwords managed by Cloudera Manager and Cloudera Navigator are redacted internally, with the following results:

- In the Cloudera Manager Admin Console:

  - In the Processes page for a given role instance, passwords in the linked configuration files are replaced by *******.
  - Advanced Configuration Snippet (Safety Valve) parameters, such as passwords and secret keys, are visible to users (such as admins) who have edit permissions on the parameter, while those with read-only access see redacted data. However, the parameter name is visible to anyone. (Data to be redacted from these snippets is identified by a fixed list of key words: password, key, aws, and secret.)
- On all Cloudera Manager Server and Cloudera Manager Agent hosts:

  - Passwords in the configuration files in /var/run/cloudera-scm-agent/process are replaced by ********.

Cloudera Manager Server Database Password Handling

Unlike the other passwords that are redacted or encrypted by Cloudera Manager, the password used for the Cloudera Manager Server database is stored in plaintext in the configuration file, /etc/cloudera-scm-server/db.properties, as shown in this example:

```
# Auto-generated by scm_prepare_database.sh on Mon Jan 30 05:02:18 PST 2017
#
# For information describing how to configure the Cloudera Manager Server
# to connect to databases, see the "Cloudera Manager Installation Guide."
#
com.cloudera.cmf.db.type=mysql
com.cloudera.cmf.db.host=localhost
com.cloudera.cmf.db.name=cm
com.cloudera.cmf.db.user=cm
com.cloudera.cmf.db.setupType=EXTERNAL
com.cloudera.cmf.db.password=PASSWORD
```

Instead of using a cleartext password, you can use a script or other executable that uses stdout to return a password for use by the system.

During installation of the database, you can pass the script name to the scm_prepare_database.sh script with the --scm-password-script parameter. See “Step 5: Set up and Configure the Cloudera Manager Database” and “Syntax for scm_prepare_database.sh” for details.

You can also replace an existing cleartext password in /etc/cloudera-scm-server/db.properties by replacing the com.cloudera.cmf.db.password setting with com.cloudera.cmf.db.password_script and setting the name of the script or executable:

Cleartext Password (5.9 and prior)	Script (5.10 and higher)
com.cloudera.cmf.db.password=*PASSWORD*	com.cloudera.cmf.db.password_script=*SCRIPT_NAME_HERE*

At runtime, if /etc/cloudera-scm-server/db.properties does not include the script identified by com.cloudera.cmf.db.password_script, the system looks for the value of com.cloudera.cmf.db.password.

Cloudera Manager API Redaction

Cloudera Manager API has redaction enabled by default. If you use the API to export the configuration, the output may contain passwords and other sensitive information. The Cloudera Manager API automatically redacts the sensitive items returned from API calls.

You can disable redaction of the sensitive items by specifying a JVM parameter for Cloudera Manager. For more information, see “Disabling Redaction of sensitive information when using the Cloudera Manager API”.

Log and Query Redaction

Cloudera Manager provides a configurable log and query redaction feature that lets you redact sensitive data in the Cloudera cluster as it's being written to the log files (see the Cloudera Engineering Blog “Sensitive Data Redaction” post for a technical overview), to prevent leakage of sensitive data. Redaction works only on data, not metadata—that is, sensitive data inside files is redacted, but the name, owner, and other metadata about the file is not.

Redaction is enabled for the entire cluster through the Cloudera Manager Admin Console, which also lets you define rules to target sensitive data in SQL data and log files. After enabling data redaction, the following contain replacement strings (such as a series of Xs) for the sensitive data items you define in your rules:

- Logs in HDFS and any dependent cluster services.
- Audit data sent to Cloudera Navigator.
- SQL query strings displayed by Data Explorer, Hive, and Impala.

Note: Log redaction is not available in Isilon-based clusters.

See “Enabling Log and Query Redaction Using Cloudera Manager” (below) for information about how to enable and define rules for sensitive data redaction for your cluster's logs and SQL queries (Hive, Data Explorer, Impala).

How Redaction Rules Work

Cloudera's redaction process (redactor) uses regular expressions to target data for redaction. Common regular expression patterns for sensitive data include social security numbers, credit card numbers, email addresses, and dates, for example. The redaction rules are specified using the following elements:

- Search - Regular expression to compare against the data. For example, the regular expression \d{4}[^\w]\d{4}[^\w]\d{4}[^\w]\d{4} searches for a credit card number pattern. Segments of data that match the regular expression are redacted using the Replace string.
- Replace - String used to redact (obfuscate) data, such as a pattern of Xs to replace digits of a credit card number: XXXX-XXXX-XXXX-XXXX.
- Trigger - Optional simple string to be searched before applying the regular expression. If the string is found, the redactor searches for matches using the Search regular expression. Using the Trigger field improves performance: simple string matching is faster than regular expression matching.

You can use the following preconfigured redaction rules on your cluster. Rules are applied in the order listed in the table.

<table>
<tr><th>Rule</th><th>Regex Pattern</th><th>Replacement</th></tr>
<tr><td>Credit Card numbers (with separator)</td><td>\d{4}[^\w]\d{4}[^\w]\d{4}[^\w]\d{4}</td><td>XXXX-XXXX-XXXX-XXXX</td></tr>
<tr><td>Social Security numbers (with separator)</td><td>\d{3}[^\w]\d{2}[^\w]\d{4}</td><td>XXX-XX-XXXX</td></tr>
<tr><td>Email addresses</td><td>\b([A-Za-z0-9]|[A-Za-z0-9][A-Za-z0-9\-\._] \ *[A-Za-z0-9])@(([A-Za-z0-9]|[A-Za-z] \ [A-Za-z0-9\-]*[A-Za-z0-9])\.)+([A-Za-z0-9] \ |[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])\b</td><td>email@redacted.host</td></tr>
<tr><td>Hostnames</td><td>\b(([A-Za-z]|[A-Za-z][A-Za-z0-9\-] \ *[A-Za-z0-9])\.)+([A-Za-z0-9] \ |[A-Za-z0-9][A-Za-z0-9\-]*[A-Za-z0-9])\b</td><td>HOSTNAME.REDACTED</td></tr>
</table>

Ways to optimize regular expressions

Regular expressions are used to redact logs and queries. Regular expressions are powerful tools for pattern matching and string manipulation, but their performance impact can vary significantly depending on how they are used. Cloudera recommends that you use the log and query redaction feature and construct regex patterns thoughtfully. Be aware of the implications of different regex constructs on performance. Following are a few ways to optimize regex performance:

- Simplify the pattern as much as possible
- Use non-capturing groups if you do not need to extract data.
- Avoid unnecessary backtracking.
- Consider the specificities of the regex engine you are using.
- Test the performance with realistic data sets.

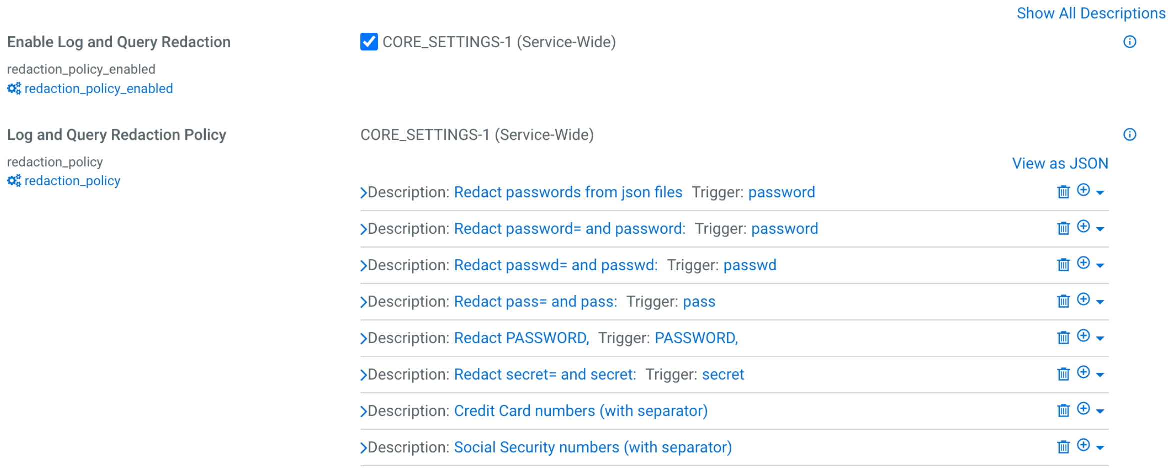
Enabling Log and Query Redaction Using Cloudera Manager

To enable log and query redaction in Cloudera Manager:

1. Login to the Cloudera Manager Admin Console.
2. Select Clusters CORE_SETTINGS .
3. Click the Configuration tab.

4. In the Search box, type redaction to find the redaction property settings:

- Enable Log and Query Redaction
- Log and Query Redaction Policy list of rules for redacting sensitive information from log files and query strings. Choose a preconfigured rule or add a custom rule. See "How Redaction Rules Work" for more information about rule pattern definitions.



Show All Descriptions

Enable Log and Query Redaction
redaction_policy_enabled
redaction_policy_enabled

☑ CORE_SETTINGS-1 (Service-Wide)

Log and Query Redaction Policy
redaction_policy
redaction_policy

CORE_SETTINGS-1 (Service-Wide)

View as JSON

›Description: Redact passwords from json files Trigger: password
›Description: Redact password= and password: Trigger: password
›Description: Redact passwd= and passwd: Trigger: passwd
›Description: Redact pass= and pass: Trigger: pass
›Description: Redact PASSWORD, Trigger: PASSWORD,
›Description: Redact secret= and secret: Trigger: secret
›Description: Credit Card numbers (with separator)
›Description: Social Security numbers (with separator)

Test your rules:

- Enter sample text into the Test Redaction Rules text box
- Click Test Redaction.

5. Click Save Changes.
6. Restart the cluster.

If no rules match, the text you entered displays in the Results field, unchanged.

Using Cloudera Navigator Data Management for Data Redaction

Important: This approach has been supplanted by Cloudera Manager's cluster-wide log and query redaction feature ("Enabling Log and Query Redaction Using Cloudera Manager"), and is not recommended.

You can specify credit card number patterns and other PII to be masked in audit events, in the properties of entities displayed in lineage diagrams, and in information retrieved from the Audit Server database and the Metadata Server persistent storage. Redacting data other than credit card numbers is not supported by default with the Cloudera Navigator property. You can use regular expressions to redact social security numbers or other PII. Masking applies only to audit events and lineage entities generated after enabling a mask.

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

1. Log into Cloudera Manager Admin Console.
2. Select Clusters Cloudera Management Service .
3. Click the Configuration tab.
4. Expand the Navigator Audit Server Default Group category.
5. Click the Advanced category.
6. Configure the PII Masking Regular Expression property with a regular expression that matches the credit card number formats to be masked. The default expression is:

```
(4[0-9]{12}(?:[0-9]{3})?)|(5[1-5][0-9]{14})|
(3[47][0-9]{13})|(3(?:0[0-5]|[68][0-9])[0-9]{11})|
(6(?:011|5[0-9]{2})[0-9]{12})|((?:2131|1800|35\\d{3})\\d{11})
```

which consolidates these regular expressions:

- Visa - (4[0-9]{12}(?:[0-9]{3})?)
- MasterCard - (5[1-5][0-9]{14})
- American Express - (3[47][0-9]{13})
- Diners Club - (3(?:0[0-5]|[68][0-9])[0-9]{11})
- Discover - (6(?:011|5[0-9]{2})[0-9]{12})
- JCB - ((?:2131|1800|35\\d{3})\\d{11})

If the property is left blank, PII information is not masked.

7. Click Save Changes.

How to Log a Security Support Case

Fastpath: Cloudera customers can get help at the support site and use the Cloudera Knowledge Base.

Before logging a support case, collect as much information about your issue as possible, as detailed below:

- If possible, gather a diagnostic bundle.
- Note specific details about the issue, including these:
  - Is this a new install, or did you upgrade a working cluster? For upgrades, identify release numbers ("trying to upgrade from Cloudera Manager 5.7.0 to Cloudera Manager 6.0.0," for example).
  - Is this a production deployment, development environment, or sandbox environment?
  - Note the severity of the impact and whether it is causing a production outage.
  - Capture error messages (screenshots, command-line capture, and so on) and attach to the case.
  - Note the commands executed and the results, captured to a file if possible.
  - Itemize all changes made to your cluster configuration after the issue arose (possibly, as attempts to resolve the issue).

The following are some additional details to gather when contacting support:

- Kerberos Issues on page 40
- LDAP Issues on page 41
- TLS/SSL Issues on page 41

Related Information

How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys

How to Add Root and Intermediate CAs to Truststore for TLS/SSL

Kerberos Issues

- For Kerberos issues, your krb5.conf and kdc.conf files are valuable for support to be able to understand your configuration.
- If you are having trouble with client access to the cluster, provide the output for klist -ef after kiniting as the user account on the client host in question. Additionally, confirm that your ticket is renewable by running kinit -R after successfully kiniting.
- Specify if you are authenticating (kiniting) with a user outside of the Hadoop cluster's realm (such as Active Directory, or another MIT Kerberos realm).
- If using AES-256 encryption, ensure you have the Unlimited Strength JCE Policy Files deployed on all cluster and client nodes.

LDAP Issues

- Specify the LDAP service in use (Active Directory, OpenLDAP, one of Oracle Directory Server offerings, OpenDJ, etc)
- Provide a screenshot of the LDAP configuration screen you are working with if you are troubleshooting setup issues.
- Be prepared to troubleshoot using the ldapsearch command (requires the openldap-clients package) on the host where LDAP authentication or authorization issues are being seen.

TLS/SSL Issues

- Specify whether you are using a private/commercial CA for your certificates, or if they are self-signed. Note that Cloudera strongly recommends against using self-signed certificates in production clusters.
- Clarify what services you are attempting to setup TLS/SSL for in your description.
- When troubleshooting TLS/SSL trust issues, provide the output of the following openssl command:

```
openssl s_client -connect host.fqdn.name:port
```

How To Obtain and Deploy Keys and Certificates for TLS/SSL

Cloudera recommends obtaining certificates from one of the trusted public certificate authorities (CA) such as Symantec or Comodo for TLS/SSL encryption for the cluster. This How To provides a brief overview of the tools used to create the various artifacts followed by the steps in the process. Plan ahead to allow time to receive signed certificates from whichever CA you use.

Note: Always check with your selected public CA before creating any CSRs and follow the CA-specific processes.

If your organization uses its own internal CA, follow your internal process for creating and submitting the CSRs.

Tools Overview

Java Keytool and OpenSSL are key management tools that let you create the security artifacts needed for TLS/SSL. See “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys” for more information beyond the two short overviews below.

Java Keytool

Oracle Java keytool is a utility included with the Oracle JDK for creating and managing cryptographic keys and certificates. During configuring the Cloudera Manager Cluster for TLS/SSL, you create the private key pairs, keystore, certificate signing requests, and create a truststore for specific use by the cluster using this software tool, as detailed in the steps throughout this guide.

Java Keytool Requirements for Cloudera Manager TLS/SSL Configuration

For any steps using the Java Keytool, be sure to:

- Use the Oracle Java keytool rather than tools such as OpenJDK.

- Use the JDK downloaded from Oracle or the Cloudera-provided Oracle JDK located in this default path on a Cloudera Manager Server host:

  ```
  {JAVA_HOME}/jre/lib/security
  ```

- Use the same version of the Java keytool for all steps. If the host has multiple JDKs installed, set the PATH variable so that the Oracle JDK is invoked first, as in this example:

  ```
  $ export JAVA_HOME= [path to your installed JDK]
  ...
  $ export JAVA_HOME=/usr/lib/jvm/java-openjdk
  ```

- Use the same *PASSWORD* for the -keypass and -storepass in any commands that invoke these two options. Cloudera Manager requires the same password for a key and its keystore.

OpenSSL

OpenSSL is an open source cryptography and TLS/SSL toolkit that has been widely used since its inception ~ 1999. Just as with Java Keytool, OpenSSL lets you create private keys, certificate requests, and keystores, and it provides options for verifying certificates.

Cloudera Manager Agent hosts act as clients of a Cloudera Manager Server host during RPC client and server communications. The Agent hosts, Data Explorer, Impala and other Python-based services require PEM-formatted keys and certificates (PKCS #8), which is why the steps below include converting some of the JKS artifacts using this tool. See “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys” for more information.

Step 1: Create Directory for Security Artifacts

Distributing the certificates, keys, truststore—in short, all security artifacts used for TLS/SSL intra-cluster communications—is part of this and some subsequent processes. To keep things organized, Cloudera recommends that you create the directory and distribute and store artifacts when you receive them, even though they may not be immediately needed.

The following table shows an example directory structure for the security artifacts created in the steps in this and subsequent sections. Use different names if you like, but for ease of deployment, use the same directory names across all hosts in the cluster.

Example	Description
cmhost.sec.example.com	FQDN of an example Cloudera Manager Server host.
/opt/cloudera/security	Base path for security-related files.
/opt/cloudera/security/pki	Path for all security artifacts associated with TLS/SSL, including keys, keystores (keystore.jks), CSR, and root- and intermediate-CA certificates.
JAVA_HOME= [path to your installed JDK]{JAVA_HOME}/jre/lib/security/jssecacerts. **Note:** If you do not have the $JAVA_HOME variable set, replace it with the JAVA_HOME path to the JDK. For example, the default path to the Java JDK on a Cloudera Manager Server host is:/usr/java/<jdk-server> or OpenJDK /usr/lib/jvm	Path to the default alternative Java truststore on a Cloudera Manager Server host system.

1. On the Cloudera Manager Server host, create the /opt/cloudera/security/pki directory:

   ```
   sudo mkdir -p /opt/cloudera/security/pki
   ```

2. This directory must be owned by cloudera-scm:cloudera-scm and have its executable bit set correctly so that Cloudera Manager can access the keystore at runtime:

```
sudo chown -R cloudera-scm:cloudera-scm /opt/cloudera/security/jks
umask 022
cd /opt/cloudera/security/jks
```

Directories and artifacts persist during system upgrades. For security purposes, for any host you remove from a cluster, remove any directories you create and more importantly, remove any security artifacts (keys, certificates, and so on) they may contain.

Step 2: Create the Java Truststore

On the Cloudera Manager Server host, copy the JDK cacerts file to jssecacerts:

```
sudo cp $JAVA_HOME/jre/lib/security/cacerts $JAVA_HOME/jre/lib/security/jsse
cacerts
```

If you do not have the $JAVA_HOME variable set, replace it with the path to the Oracle JDK. For example, the default path to the Java JDK on a Cloudera Manager Server host is:

/usr/java/<jdk-version>-cloudera/

Step 3: Generate Server Key and CSR

1. On the Cloudera Manager Server host, use the keytool utility to generate a keypair and a keystore named for the server. Replace the OU, O, L, ST, and C entries with the values for your organization name, location, and country code (US, in the example):

```
$ sudo keytool -genkeypair -alias $(hostname -f)-server -keyalg RSA -key
store \
/opt/cloudera/security/pki/$(hostname -f)-server.jks -keysize 2048 -dname
 \
"CN=$(hostname -f),OU=DEPT,O=EXAMPLE.COM,L=CITY,ST=STATE,C=US" \
-storepass PASSWORD -keypass PASSWORD -ext san=dns:$(hostname -f)
```

Use the same password for the key and its keystore (-keypass and -storepass, respectively): Cloudera Manager does not support using different passwords for the key and keystore.

Note: The key size of 2048 bits shown here is just an example. You can also create a certificate with a larger key size, such as 4096 bits, by updating the -keysize parameter accordingly.

2. Generate a certificate signing request (CSR) for the public key (contained in the keystore as a result of the command above). In this command below, enter the password that you set for the key and the keystore in the command above:

```
$ sudo keytool -certreq -alias $(hostname -f)-server \
-keystore /opt/cloudera/security/pki/$(hostname -f)-server.jks \
-file /opt/cloudera/security/pki/$(hostname -f)-server.csr -storepas
s PASSWORD \
-keypass PASSWORD -ext san=dns:$(hostname -f) -ext EKU=serverAuth,clientA
uth
```

Step 4: Submit the CSR to the CA

1. Submit the CSR file to your certificate authority using the process and means required by the CA, for example, email or web submission. For the certificate format, specify PEM (base64 ASCII) (see Step 5 below for an example of PEM formatted certificate heading and closing structure).
2. The public CA will request specific details from you, to verify that you own the domain name contained in the CSR, before they issue the certificate.
3. When you receive the signed certificate from the CA, you can proceed with Step 5.

Step 5: Verify the Certificate

From the public (or internal) CA, you receive the certificate for the server signed by the CA and several other digital artifacts, including root CA and possibly one (or more) intermediate CA certificates. Before distributing these to the hosts, make sure the certificate is in PEM format. A PEM formatted certificate file looks like this:

```
-----BEGIN CERTIFICATE-----
<The encoded certificate is represented by multiple lines of exactly 64 char
acters, except
for the last line, which can contain 64 characters or fewer.>
-----END CERTIFICATE-----
```

If your issued certificate is in binary (DER) format, convert it to PEM format before continuing. See “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys” for details.

To modify the truststore (jssecacerts) to explicitly trust certificates or certificate chain (as you might for certificates signed by an internal CA), follow the steps in “How to Add Root and Intermediate CAs to Truststore for TLS/SSL”.

Step 6: Import the Certificate into the Keystore

Copy the signed certificate you receive from the CA to the Cloudera Manager Server host. To identity the certificate's functionality, include a suffix such as "-server.cert.pem" in the target name for the copy command, as shown in this example:

```
cp CERT-FILE-RECD /opt/cloudera/security/pki/$(hostname -f)-server.cert.pem
```

Import the certificate into the keystore.

```
$ sudo keytool -importcert -alias $(hostname -f)-server \
-file /opt/cloudera/security/pki/$(hostname -f)-server.cert.pem \
-keystore /opt/cloudera/security/pki/$(hostname -f)-server.jks
```

Assuming the certificate was obtained from a public CA, you can safely disregard this message about trust, and enter yes to continue:

```
... is not trusted. Install reply anyway? [no]:  yes
```

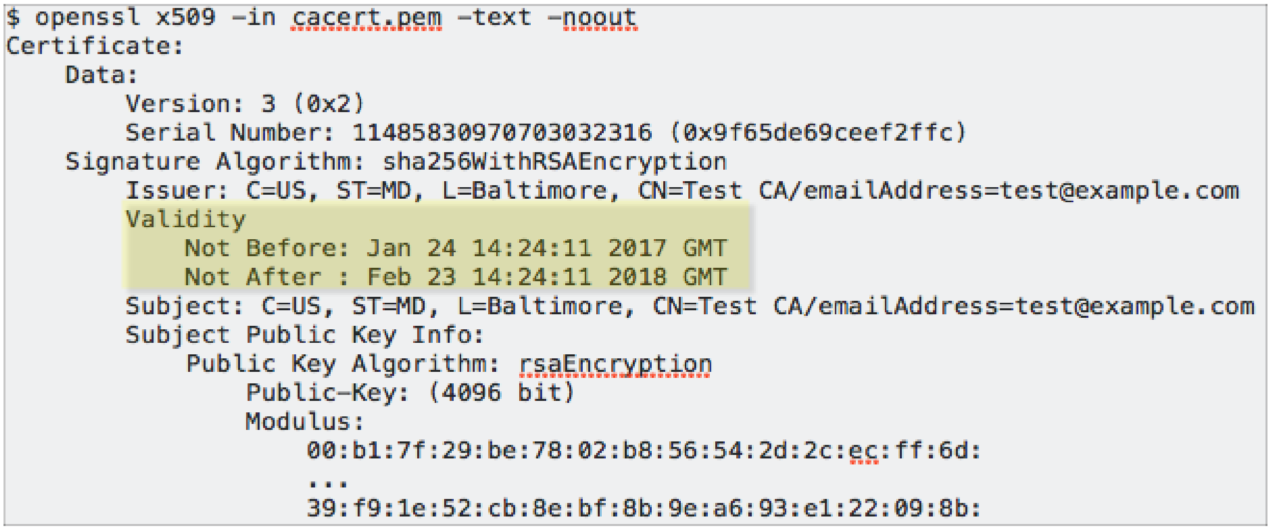
You must see the following response confirming that the certificate has been properly imported and can verify the private key that it certifies.

```
Certificate reply was installed in keystore
```

If you do not see this response, double-check all your steps up to this point: are you working in the correct path? Do you have the proper certificate? and so on. Contact Cloudera Support for assistance.

How To Renew and Redistribute Certificates

Certificates installed throughout TLS/SSL-configured clusters have a defined lifetime. That lifetime begins on a certain date and ends on another date, typically a year or two later as specified in the certificate's validity attribute. For example, the following certificate is valid between January 24, 2017 and February 23, 2018 only:



```
$ openssl x509 -in cacert.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 11485830970703032316 (0x9f65de69ceef2ffc)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, ST=MD, L=Baltimore, CN=Test CA/emailAddress=test@example.com
        Validity
            Not Before: Jan 24 14:24:11 2017 GMT
            Not After : Feb 23 14:24:11 2018 GMT
        Subject: C=US, ST=MD, L=Baltimore, CN=Test CA/emailAddress=test@example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:b1:7f:29:be:78:02:b8:56:54:2d:2c:ec:ff:6d:
                    ...
                    39:f9:1e:52:cb:8e:bf:8b:9e:a6:93:e1:22:09:8b:
```

If any certificates used for TLS/SSL in the Cloudera cluster expire, the cluster can fail completely or can exhibit various issues related to connectivity between components. For example, an expired server certificate on the Cloudera Manager Server host will be rejected by the Cloudera Manager Agent host and prevent the cluster node from launching.

Warning: Replace certificates before they expire to avoid issues. If certificates expire and the cluster or any nodes fail to start as a result, see “Replacing Certificates After a Certificate Expires” below for steps to resolve.

Obtaining certificates with new expiration dates takes just as much time as it did to obtain them in the first place. This guide steps you through the process.

Note: Keep track of the expiration dates for all certificates installed in Cloudera clusters. For example, add certificate expiration dates to your calendar with notifications set to alter you at least 1 month in advance of the expiration date.

Related Information

Configuring TLS Encryption for Cloudera Manager Using Auto-TLS

How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys

Understanding Keystores and Truststores

How To Obtain and Deploy Keys and Certificates for TLS/SSL

How to Add Root and Intermediate CAs to Truststore for TLS/SSL

Assumptions and Requirements

The steps below assume that the cluster has been configured for TLS/SSL using certificates obtained previously as detailed in “How to Configure TLS Encryption for Cloudera Manager”, and that the cluster has been operational using those certificates. This table recaps recommended paths to various security artifacts:

Example	Description
/opt/cloudera/security	Base path for security-related files.
/opt/cloudera/security/pki	Path for all security artifacts associated with TLS/SSL, including keys, keystores (keystore.jks), CSR, and root- and intermediate-CA certificates.
JAVA_HOME= [path to your installed JDK]{JAVA_HOME}/jre/lib/security/jssecacerts. **Note:** If you do not have the $JAVA_HOME variable set, replace it with the JAVA_HOME path to the JDK. For example, the default path to the Java JDK on a Cloudera Manager Server host is:/usr/java/<jdk-server> or OpenJDK /usr/lib/jvm	Path to the default alternative Java truststore on a Cloudera Manager Server host system.

This guide assumes that the Cloudera Manager Server host uses the jssecacerts truststore and includes all CA certs from cacerts and any intermediate CA certificates needed to enable successful chain of trust traversal during handshake.

```
sudo cp $JAVA_HOME/jre/lib/security/cacerts $JAVA_HOME/jre/lib/security/jsse
cacerts
```

Use Cloudera Manager Admin Console to check TLS/SSL configuration details for the cluster and services, and to verify paths to keys and keystores, certificates, and trust stores configured for each service. These do not need to be re-enabled or changed (unless you replace existing keys with new ones as part of this process), but you can note all paths and names of all TLS-related security artifacts before you begin.

Certificates and keys may have been converted from one format to another (as detailed in “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys”). That means that a CSR may have been used to obtain a JKS formatted certificate for one service that was then converted to PEM for use by another service (or services) running on the same node of the cluster as needed.

Note: Use the same names for certificates to avoid having to reconfigure the TLS/SSL settings in Cloudera Manager Admin Console.

Check Certificate Expiration Dates

Cloudera Manager 6.3 and higher alerts you 60 days before the Cloudera Manager Server TLS certificate expires. You can view and modify the threshold values by searching for Expiry in the Cloudera Management Service configuration page (Cloudera Management Service Configuration).

Note: The TLS certificate expiry alert applies only to the Cloudera Manager Server certificate. It does not alert on upcoming certificate expirations for certificates used by other services. Use the instructions provided below to check the other host certificates.

If you do not know the expiration dates for certificates installed on the cluster, use OpenSSL (for PEM-formatted certificates) or use Java Keytool (for JKS-formatted certificates) to determine certificate expiration dates.

PEM-formatted certificates (PKCS #8) are used by Cloudera Manager Agent hosts, Data Explorer, Impala and other Python-based services, while JKS-formatted certificates are used by HDFS, MapReduce, and YARN, for example. See “Understanding Keystores and Truststores” for more information.

Using OpenSSL to Obtain Certificate Details

To check the expiration dates of PEM-formatted certificates using openssl and the certificate by name:

```
openssl x509 -enddate -noout -in /OPT/CLOUDERA/SECURITY/PKI/$(HOSTNAME -F)-
SERVER.CERT.PEM
```

To check expiration dates by querying the active listener ports for any TLS-enabled services from the command line, use OpenSSL as in this example of querying the Cloudera Manager TLS listener port (7183):

```
echo | openssl s_client -connect FQDN.EXAMPLE.COM:7183 2>/dev/null | open
ssl x509 -noout -subject -dates
```

The subject and dates should display, as shown in this example:

```

subject=/C=US/ST=California/L=Los Angeles/O=Internet Corporation for Assign
ed Names
and Numbers/OU=Technology/CN=www.example.org
notBefore=Nov 3 00:00:00 2015 GMT
notAfter=Nov 28 12:00:00 2018 GMT
```

Using Java Keytool to Obtain Certificate Details

To obtain all information about a certificate:

```
keytool -list -v -keystore KEYSTORE_NAME.jks
```

Renewing and Replacing Certificates Before Expiration

There are many different ways to proceed depending on your specific needs. Renewing a certificate really means obtaining a new certificate, one with a new start and end date, and updating the key in the keystore with the certificate so it can being presented during the TLS handshake. That means you can replace the key at the same time, or re-use the existing key but apply a new certificate to it. If you have the original Certificate Signing Request (CSR), you can resubmit that to the commercial Certificate Authority (CA) as was done when TLS/SSL was configured for the cluster. Here is a summary of the various approaches:

- Use the existing key to generate a new CSR for a certificate to replace the one currently associated with the key, as detailed in “Step 1: Generate a New CSR from the Key” below. This requires knowing the key/keystore password.
- Create a new private key and public key to replace those already in use and generate a new CSR to obtain a completely new certificate to use with the public key. To use this approach, follow steps in “How To Obtain and Deploy Keys and Certificates for TLS/SSL”.
- Submit the original CSR to the CA and obtain a new certificate. If you have the CSR that matches the key used by the service, you can skip Step 1 and start with “Step 2: Submit the CSR to the CA” below.

Step 1: Generate a New CSR from the Key

You must know the password for the key and keystore to generate a new signing request from it. Use the same alias used for the key when it was created.

1. Use keytool to generate the CSR as follows:

   ```
   $ sudo keytool -certreq -keystore KEYSTORE.JKS -file NEWCERT.CSR -alias
   <cert_alias>
   Enter keystore password:
   ```

Note: The certificate needs to contain an alias (which could be ,but not necessarily the hostname) and also a private key.

Step 2: Submit the CSR to the CA

1. Submit the CSR file to your certificate authority using the process and means required by the CA, for example, email or web submission. For the certificate format, specify PEM (base64 ASCII) (see “Step 3: Verify the Certificate” below for an example of PEM formatted certificate heading and closing structure).
2. The public CA will request specific details from you, to verify that you own the domain name contained in the CSR, before they issue the certificate.
3. When you receive the signed certificate from the CA, you can proceed with Step 3.

Step 3: Verify the Certificate

From any Certificate Authority (CA), you should receive a certificate signed by the CA attesting to the validity of the key. Before distributing these to the hosts, make sure the certificate is in PEM format. A PEM formatted certificate file looks like this:

```
-----BEGIN CERTIFICATE-----
<The encoded certificate is represented by multiple lines of exactly 64 char
acters, except
for the last line, which can contain 64 characters or fewer.>
-----END CERTIFICATE-----
```

If your issued certificate is in binary (DER) format, convert it to PEM format before continuing. See “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys” for details.

Along with the certificate, the CA also provides several other digital artifacts, including root CA and possibly one (or more) intermediate CA certificates. The intermediate certificates may need to be added to the trust store in the following situations:

For certificates obtained from a public (commercial) CA—Intermediate or root certificates do not need to be installed in the trust stores for the cluster. The JDK trust store already includes the certificates needed to establish chain of trust for certificates obtained from commercial CAs.

For certificates obtained from an internal CA—To modify the truststore (jssecacerts) to explicitly trust certificates or certificate chain (as you might for certificates signed by an internal CA), follow the steps in “How to Add Root and Intermediate CAs to Truststore for TLS/SSL”.

Step 4: Import the Certificate into the Keystore

Copy the signed certificate you receive from the CA to the Cloudera Manager Server host. To identity the certificate's functionality, include a suffix such as "-server.cert.pem" in the target name for the copy command, as shown in this example:

```
cp CERT-FILE-RECD /opt/cloudera/security/pki/$(hostname -f)-server.cert.pem
```

Import the certificate into the keystore.

```
$ sudo keytool -importcert -alias $(hostname -f)-server \
-file /opt/cloudera/security/pki/$(hostname -f)-server.cert.pem \
-keystore /opt/cloudera/security/pki/$(hostname -f)-server.jks
```

Assuming the certificate was obtained from a public CA, you can safely disregard this message about trust, and enter yes to continue:

```
... is not trusted. Install reply anyway? [no]:  yes
```

You must see the following response confirming that the certificate has been properly imported and can verify the private key that it certifies.

```
Certificate reply was installed in keystore
```

If you do not see this response, double-check all your steps up to this point: are you working in the correct path? Do you have the proper certificate? and so on. Contact Cloudera Support for assistance.

Step 5: Restart related services

Restart related services, so that those services pick up the new certificate.

Replacing Certificates After a Certificate Expires

If the cluster cannot start up because of an expired certificate, then perform the steps in “Renewing and Replacing Certificates Before Expiration” to resolve the issue.

If the replacement certificate has not been obtained yet, you can use a self-signed certificate for the short term, until you receive the CA signed certificate.

How to Set Up a Gateway Host to Restrict Access to the Cluster

The steps below configure a firewall-protected Hadoop cluster that allows access only through the host configured as the gateway. Clients access the cluster through the gateway using the REST API, for example, using HttpFS (which provides REST access to HDFS) or using Oozie, which allows REST access for submitting and monitoring jobs.

Installing and Configuring the Firewall and Gateway

Follow these steps:

1. Choose a cluster host to be the gateway host.
2. Following the instructions in “Adding a Service” or “Role Instances”, add the roles and services that you want to be available. For example, you can add the HttpFS role to the HDFS service, or add the Oozie service to your cluster.
3. If you have not already done so, add the Oozie service to your cluster, following the instructions in “Adding a Service”. Assign the Oozie Server role to your selected gateway host.
4. Configure your firewall to block access to all cluster nodes from outside the cluster, with the exception of the gateway host. Open the ports to the gateway host for the services you want to make available. In general, Cloudera recommends that you only allow access to the gateway host from clients that require access. For more information on the ports used by Cloudera software, see “Ports”.
5. If you have not already done so, enable security for your cluster, including Kerberos authentication, Apache Ranger authorization, and encryption for data at rest or in transit. For more information, see “Cloudera Security Overview”.

Accessing HDFS

With the Hadoop client:

All of the standard `hadoop fs` commands work; just make sure to specify -fs webhdfs://HOSTNAME:14000. For example (where *GATEWAYHOST* is the hostname of the gateway machine):

```
$ hadoop fs -fs webhdfs://GATEWAYHOST:14000 -cat /user/me/myfile.txt
Hello World!
```

Without the Hadoop client:

You can run all of the standard `hadoop fs` commands by using the WebHDFS REST API and any program that can do `GET` , `PUT`, `POST`, and `DELETE` requests; for example:

```
$ curl "http://GATEWAYHOST:14000/webhdfs/v1/user/me/myfile.txt?op=OPEN&use
r.name=me"
Hello World!
```

Important: The user.name parameter is valid only if security is disabled. In a secure cluster, you must a initiate a valid Kerberos session.

In general, the command will look like this:

```
curl "http://GATEWAYHOST/webhdfs/v1/PATH?[user.name=USER&]op=…"
```

You can find a full explanation of the commands in the WebHDFS REST API documentation (see "APIs for accessing HDFS").

Submitting and Monitoring Jobs

The Oozie REST API supports the direct submission of jobs for MapReduce, Pig, and Hive; Oozie automatically creates a workflow with a single action. For any other action types, or to run anything more complicated than a single job, you must create an actual workflow. Required files (JAR files, input data) must already exist on HDFS; if they do not, you can use HttpFS to upload the files.

With the Oozie client:

All of the standard Oozie commands will work. You can find a full explanation of the commands in the documentation for the command-line utilities (see "Accessing the Oozie server with the Oozie Client)".

Without the Oozie client:

You can run all of the standard Oozie commands by using the REST API and any program that can do GET, PUT, and POST requests. You can find a full explanation of the commands in the Oozie Web Services API documentation.

Related Information

Adding a Service

Role Instances

Security Overview

Ports

APIs for accessing HDFS

Accessing the Oozie server with the Oozie Client

How To Set Up Access to Cloudera EDH (Microsoft Azure Marketplace)

The Cloudera Enterprise Data Hub (EDH) and Cloudera Altus Director bundles that are available on the Microsoft Azure Marketplace only support SSH access (port 22). To access Cloudera Manager (port 7180), Altus Director (port 7189), or other services, you can:

- Set up a SOCKS ("Sockets Secure protocol") proxy on your client machine. Cloudera recommends that you use this option.
- Add inbound rules to the Network Security Group in the Azure instance after you deploy EDH or Director to Azure.

Related Information

Sockets Secure protocol

SSH tunnel to endpoints in Azure VNet from Windows

Microsoft Azure Portal

Configure the SOCKS Proxy

The SOCKS5 protocol is implemented as a client and server process that enables traversal of IP network firewalls. After you configure the SOCKS proxy, your browser resolves DNS lookups using the Microsoft Azure network (through the proxy server), and lets you connect to services using internal FQDNs or private IP address.

With this approach, you complete the following tasks:

- Set up a single SSH tunnel to one of the hosts on the network and create a SOCKS proxy on the host.
- Change the browser configuration to perform all lookups through the SOCKS proxy host.

Network Prerequisites

Verify the following prerequisites before you connect to your cluster with a SOCKS proxy:

- You must be able to reach the host that you want to proxy to from the public internet or from the network that you're connecting from.
- The host that you proxy to must be on the same network as the Cloudera services that you're connecting to. For example, if you’re using the Cloudera EDH offering, tunnel to the Cloudera Manager host.

Find the Public IP of the Host

For the Cloudera EDH offering, use the public IP of the 0th master node VM: [dnsName]-mn0.

Start the SOCKS Proxy

On Linux

To start a SOCKS proxy over SSH, run the following command:

```
ssh -i your-key-file.pem -CND 1080
the_username_you_specified@publicIP_of_VM
```

The command uses the following parameters:

- -i your-key-file.pem Specifies the path to the private key needed to SSH to the Cloudera EDH server. Omit if using SSH passwords.
- C Sets up compression.
- N Suppresses any command execution once established.
- D Sets up the SOCKS proxy on a port.
- 1080 The port to set the SOCKS proxy locally.

On Windows

Follow the instructions on Microsoft's website (see “SSH tunnel to endpoints in Azure VNet from Windows”).

Configure Google Chrome to Use the Proxy

By default, Google Chrome uses system-wide proxy settings on a per-profile basis. To start Chrome without these settings, open Chrome through the command line and specify the following:

- The SOCKS proxy port. This must be the same port that you used when you started the proxy.
- The profile. This examples below create a new profile.

Use one of the following commands to create a profile and launch a new instance of Chrome that does not conflict with any currently running Chrome instances.

Linux

```
/usr/bin/google-chrome \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:1080"
```

Mac OS X

```
"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" \
--user-data-dir="$HOME/chrome-with-proxy" \
--proxy-server="socks5://localhost:1080"
```

Microsoft Windows

```
"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" ^
--user-data-dir="%USERPROFILE%\chrome-with-proxy" ^
--proxy-server="socks5://localhost:1080"
```

In this Chrome session, you can use the private IP address or internal FQDN to connect to any host that is accessible by Cloudera EDH.

Network Security Group

Warning: This method is not recommended for any purpose other than a Proof of Concept. If the data is not carefully locked down, it will be accessible to hackers and malicious entities.

On the “Microsoft Azure Portal”, find the Network Security Group(s) and add inbound rules for the various services. You might have to create these rules for the services.

How to Use Self-Signed Certificates for TLS

Self-signed certificates should not be used for production deployments. Self-signed certificates are created and stored in the keystore specified during the key-generation process, and should be replaced by a signed certificate. Using self-signed certificates requires generating and distributing the certificates and establishing explicit trust for the certificate.

However, using self-signed certificates lets you easily obtain certificates for TLS/SSL configuration and may be appropriate for non-production or test setups. See “Manually Configuring TLS Encryption for Cloudera Manager” for more information.

Replace paths, file names, aliases, and other examples in the commands below for your system.

1. Create the directory for the certificates:

   ```
   mkdir -p /opt/cloudera/security/x509/ /opt/cloudera/security/jks/
   ```

   Give Cloudera Manager access to the directory, set the correct permissions, and then change to the directory:

   ```
   sudo chown -R cloudera-scm:cloudera-scm /opt/cloudera/security/jks
   sudo umask 0700
   cd /opt/cloudera/security/jks
   ```

2. Generate the key pair and self-signed certificate, storing everything in the keystore with the same password for keystore and storepass, as shown below. Use the FQDN of the current host for the CN to avoid raising a

java.io.IOException: HTTPS hostname wrong exception. Replace values for OU, O, L, ST, and C with entries appropriate for your environment:

```
keytool -genkeypair -alias cmhost -keyalg RSA -keysize 2048 -dname "cn=c
m01.example.com, ou=Department,
o=Company, l=City, st=State, c=US" -keypass password -keystore example.jks
 -storepass password
```

3. Copy the default Java truststore (cacerts) to the alternate system truststore (jssecacerts):

```
sudo cp $JAVA_HOME/jre/lib/security/cacerts $JAVA_HOME/jre/lib/security/
jssecacerts
```

4. Export the certificate from the keystore (example.jks).

```
keytool -export -alias cmhost -keystore example.jks -rfc -file selfsigne
d.cer
```

5. Copy the self-signed certificate (selfsigned.cer) to the /opt/cloudera/security/x509/ directory.

```
cp selfsigned.cer /opt/cloudera/security/x509/cmhost.pem
```

6. Import the public key into the alternate system truststore (jssecacerts), so that any process that runs with Java on this machine will trust the key. The default password for the Java truststore is changeit. Do not use the password created for the keystore in step 2.

```
$ keytool -import -alias cmhost -file /opt/cloudera/security/jks/selfsig
ned.cer
-keystore $JAVA_HOME/jre/lib/security/jssecacerts -storepass changeit
```

⚠ **Important:** Repeat this process on each host in the cluster.

7. Rename the keystore:

```
mv /opt/cloudera/security/jks/example.jks /opt/cloudera/security/jks/cmh
ost-keystore.jks
```

You can also delete the certificate because it was copied to the appropriate path in step 5.

```
rm /opt/cloudera/security/jks/selfsigned.cer
```

The self-signed certificate set up is complete.

Related Information

Manually Configuring TLS Encryption for Cloudera Manager