

# Encrypting Data in Transit in Cloudera Manager

Date published: 2020-11-30

Date modified: 2021-03-03



# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

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

<b>Encrypting Data in Transit.....</b>	<b>5</b>
TLS/SSL and Its Use of Certificates.....	5
Certificates Overview.....	5
Wildcard Domain Certificates and SAN Certificates Support.....	6
Renew Certificates Before Expiration Dates.....	6
<b>Understanding Keystores and Truststores.....</b>	<b>6</b>
<b>Choosing manual TLS or Auto-TLS.....</b>	<b>9</b>
<b>SAN Certificates.....</b>	<b>12</b>
<b>Configuring TLS Encryption for Cloudera Manager Using Auto-TLS.....</b>	<b>12</b>
Auto-TLS Requirements and Limitations.....	14
Rotating Auto-TLS Certificate Authority and Host Certificates.....	14
Auto-TLS Agent File Locations.....	14
Use case 1: Use Cloudera Manager to generate internal CA and corresponding certificates.....	15
Use case 2: Enabling Auto-TLS with an intermediate CA signed by an existing Root CA.....	17
Certmanager Options - Using CM's GenerateCMCA API.....	18
Use case 3: Enabling Auto-TLS with Existing Certificates.....	21
<b>Manually Configuring TLS Encryption for Cloudera Manager.....</b>	<b>25</b>
Generate TLS Certificates.....	26
On Each Cluster Host:.....	26
On the Cloudera Manager Server Host.....	29
Configure TLS for the Cloudera Manager Admin Console.....	29
Step 1: Enable HTTPS for the Cloudera Manager Admin Console.....	30
Step 2: Specify SSL Truststore Properties for Cloudera Management Services.....	30
Step 3: Restart Cloudera Manager and Services.....	31
Configure TLS for Cloudera Manager Agents.....	31
Step 1: Enable TLS Encryption for Agents in Cloudera Manager.....	31
Step 2: Enable TLS on Cloudera Manager Agent Hosts.....	31
Step 3: Restart Cloudera Manager Server and Agents.....	32
Step 4: Verify that the Cloudera Manager Server and Agents are Communicating.....	32
Enable Server Certificate Verification on Cloudera Manager Agents.....	32
Configure Agent Certificate Authentication.....	33
Step 1: Export the Private Key to a File.....	33
Step 2: Create a Password File.....	34
Step 3: Configure the Agent to Use Private Keys and Certificates.....	34
Step 4: Enable Agent Certificate Authentication.....	34
Step 5: Restart Cloudera Manager Server and Agents.....	35
Step 6: Verify that Cloudera Manager Server and Agents are Communicating.....	35

<b>Manually Configuring TLS Encryption on the Agent Listening Port.....</b>	<b>36</b>
---	-----------

## Encrypting Data in Transit

How to configure TLS/SSL encryption in Cloudera Manager.

Transport Layer Security (TLS) 1.2 is an industry standard set of cryptographic protocols for securing communications over a network. TLS evolved from Secure Sockets Layer (SSL). Because the SSL terminology is still widely used, Cloudera software and documentation refer to TLS as TLS/SSL, but the actual protocol used is TLS. SSL is not used in Cloudera software.

In addition to TLS/SSL encryption, HDFS and HBase transfer data using remote procedure calls (RPCs). To secure this transfer, you must enable RPC encryption.

For instructions on enabling TLS/SSL and RPC encryption, see the following topics:

### Related Information

[Transport Layer Security \(TLS\) 1.2](#)

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

## TLS/SSL and Its Use of Certificates

TLS/SSL provides privacy and data integrity between applications communicating over a network by encrypting the packets transmitted between endpoints (ports on a host, for example). Configuring TLS/SSL for any system typically involves creating a private key and public key for use by server and client processes to negotiate an encrypted connection at runtime. In addition, TLS/SSL can use certificates to verify the trustworthiness of keys presented during the negotiation to prevent spoofing and mitigate other potential security issues.

Setting up Cloudera clusters to use TLS/SSL requires creating private key, public key, and storing these securely in a keystore, among other tasks. Although adding a certificate to the keystore may be the last task in the process, the lead time required to obtain a certificate depends on the type of certificate you plan to use for the cluster.

## Certificates Overview

A certificate is digitally signed, typically by a certificate authority (CA) that indirectly (through a chain of trust) verifies the authenticity of the public key presented during the negotiation. Certificates can be signed in one of the three different ways shown in the table:

Type	Usage Note
Public CA-signed certificates	Recommended. This type of certificate is signed by a public certificate authority (CA), such as Symantec or Comodo. Public CAs are trusted third-parties whose certificates can be verified through publicly accessible chains of trust. Using this type of certificate can simplify deployment because security infrastructure, such as root CAs, are already contained in the Java JDK and its default truststore.
Internal CA-signed certificates	This type of certificate is signed by your organization's internal CA. Organizations using OpenSSL Certificate Authority, Microsoft Active Directory Certificate Service, or another internal CA system can use this type of certificate.
Self-signed certificates	Not recommended for production deployments. Self-signed certificates are acceptable for use in non-production deployments, such as for proof-of-concept setups.

During the process of configuring TLS/SSL for the cluster, you typically obtain a certificate for each host in the cluster, and re-use the certificate obtained in a given format (JKS, PEM) as needed for the various services (daemon roles) supported by the host. For information about converting formats, see “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys”. As an alternative to creating discrete certificates for each host in the cluster, Cloudera cluster components support wildcard domains and SubjectAlternateName certificates.

## Wildcard Domain Certificates and SAN Certificates Support

Cloudera Manager and CDP support the use of wildcard domain certificates and SAN certificates.

A wildcard certificate—a certificate with the common name \*, as in \*.example.com, rather than a specific host name—can be used for any number of first level sub-domains within a single domain name. For example, a wildcard certificate can be used with host-1.example.com, host-2.example.com, host-3.example.com, and so on.

Certificates obtained from public CAs are not free, so using wildcard certificates can reduce costs. Using wildcard certificates also makes it easier to enable encryption for transient clusters and for clusters that need to expand and shrink, since the same certificate and keystore can be re-used.



**Important:** Be aware that using wildcard domain certificates has some security risks. Specifically, because all nodes use the same certificate, a breach of any one machine can result in a breach of all machines.

Wildcard Certificates	Wildcard certificates can be used by all hosts within a given domain. Using wildcard certificates for all hosts in the cluster can reduce costs but also exposes greater potential risk.
SubjectAlternativeName Certificates	SubjectAlternativeName (SAN) certificates are bound to a set of specific DNS names. A single SAN certificate can be used for all hosts or a subset of hosts in the cluster. SAN certificates are used in Cloudera Manager high-availability (HA) configurations.

## Renew Certificates Before Expiration Dates

The signed certificates you obtain from a public CA (or those you obtain from an internal CA) have an expiration date, such as that shown in this excerpt:

```
$ 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:
```

Expired certificates cause most cluster operations to fail. Cloudera Manager Agent hosts, for example, will not be able to validate the Cloudera Manager Server host and will fail to launch the cluster nodes. Administrators should note expiration dates of all certificates when they deploy the certificates to the cluster nodes and setup reminders to allow enough time to renew.



**Tip:** Use OpenSSL to check the expiration dates for certificates already deployed:

```
openssl x509 -enddate -noout -in /opt/cloudera/security/pki/$(hostname
-f)-server.cert.pem
```

## Understanding Keystores and Truststores

Configuring Cloudera Manager Server and cluster components to use TLS/SSL requires obtaining keys, certificates, and related security artifacts.

## Java Keystore and Truststore

All clients in a Cloudera Manager cluster configured for TLS/SSL need access to the truststore to validate certificates presented during TLS/SSL session negotiation. The certificates assure the client or server process that the issuing authority for the certificate is part of a legitimate chain of trust.

The standard Oracle Java JDK distribution includes a default truststore (cacerts) that contains root certificates for many well-known CAs, including Symantec. Rather than using the default truststore, Cloudera recommends using the alternative truststore, jssecacerts. The alternative truststore is created by copying cacerts to that filename (jssecacerts). Certificates can be added to this truststore when needed for additional roles or services. This alternative truststore is loaded by Hadoop daemons at startup.



**Important:** For use with Cloudera clusters, the alternative trust store—jssecacerts—must start as a copy of cacerts because cacerts contains all available default certificates needed to establish the chain of trust during the TLS/SSL handshake. After jssecacerts has been created, new public and private root CAs are added to it for use by the cluster. See “Manually Configuring TLS Encryption for Cloudera Manager” > “On Each Cluster Host” for details.

The private keys are maintained in the keystore.



**Note:** For detailed information about the Java keystore and truststore, see Oracle documentation:

- Keytool—Key and Certificate Management Tool
- JSSE Reference Guide for Java

















Although the keystore and truststore in some environments may comprise the same file, as configured for Cloudera Manager Server and CDP clusters, the keystore and truststore are distinct files. For Cloudera Manager Server clusters, each host should have its own keystore, even if the content is identical while using wildcard certificates. Also, each host should have a truststore file, even if the content of the truststore is identical across hosts. This table summarizes the general differences between keystore and the truststore in Cloudera Manager Server clusters.

Keystore	Truststore
Used by the server side of a TLS/SSL client-server connection.	Used by the client side of a TLS/SSL client-server connection.
Typically contains 1 private key for the host system.	Contains no keys of any kind.
Contains the certificate for the host's private key.	Contains root certificates for well-known public certificate authorities. May contain certificates for intermediary certificate authorities.
Password protected. Use the same password for the key and its keystore.	Password-protection not needed. However, if password has been used for the truststore, never use the same password as used for a key and keystore.
Password stored in a plaintext file read permissions granted to a specific group only (OS filesystem permissions set to 0440, hadoop:hadoop).	Password (if there is one for the truststore) stored in a plaintext file readable by all (OS filesystem permissions set to 0440).
No default. Provide a keystore name and password when you create the private key and CSR for any host system.	For Java JDK, cacerts is the default unless the alternative default jssecacerts is available.
Must be owned by hadoop user and group so that HDFS, MapReduce, YARN can access the private key.	HDFS, MapReduce, and YARN need client access to truststore.

The details in the table above are specific to the Java KeyStore (JKS) format, which is used by Java-based cluster services such as Cloudera Manager Server, Cloudera Management Service, and many (but not all) CDP components and services. See “Certificate Formats (JKS, PEM) and Cluster Components” for information about certificate and key file type used various processes.

## CDP Services as TLS/SSL Servers and Clients

Cluster services function as a TLS/SSL server, client, or both:

Component	Client	Server
HBase		
HDFS		
Hive		
Hue (Hue is a TLS/SSL client of HDFS, MapReduce, YARN, HBase, and Oozie.)		
MapReduce		
Oozie		
YARN		
ZooKeeper		













Daemons that function as TLS/SSL servers load the keystores when starting up. When a client connects to an TLS/SSL server daemon, the server transmits the certificate loaded at startup time to the client, and the client then uses its truststore to validate the certificate presented by the server.

### Certificate Formats (JKS, PEM) and Cluster Components

Cloudera Manager Server, Cloudera Management Service, and many other CDP services use JKS formatted keystores and certificates. Cloudera Manager Agent, Hue, Key Trustee Server, Impala, and other Python or C++ based services require PEM formatted certificates and keystores rather than Java. Specifically, PEM certificates conform to PKCS #8, which requires individual Base64-encoded text files for certificate and password-protected private key file. The table summarizes certificate types required by several components.

Component	JKS	PEM
HBase		
HDFS		
Hive (Hive clients and HiveServer 2)		



Component	JKS	PEM
Hue		
Impala		
MapReduce		
Oozie		
Solr		
YARN		
ZooKeeper		

For more information, see:

- “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys”
- OpenSSL Cryptography and TLS/SSL Toolkit

### Recommended Keystore and Truststore Configuration

Cloudera recommends the following for keystores and truststores for Cloudera Manager clusters:

- Create a separate keystore for each host. Each keystore should have a name that helps identify it as to the type of host—server or agent, for example. The keystore contains the private key and should be password protected.
- Create a single truststore that can be used by the entire cluster. This truststore contains the root CA and intermediate CAs used to authenticate certificates presented during TLS/SSL handshake. The truststore does not need to be password protected.

The steps included in “Manually Configuring TLS Encryption for Cloudera Manager”>“On Each Cluster Host” follow this approach.

### Related Information

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

[Manually Configuring TLS Encryption for Cloudera Manager](#)

[keytool - Key and Certificate Management Tool](#)

[Java Secure Socket Extension \(JSSE\) Reference Guide](#)

## Choosing manual TLS or Auto-TLS

An explanation of the difference between manual TLS and Auto-TLS in Cloudera Manager.

Wire encryption protects data in motion, and Transport Layer Security (TLS) is the most widely used security protocol for wire encryption. TLS provides authentication, privacy and data integrity between applications communicating over a network by encrypting the packets transmitted between endpoints. Users interact with Hadoop clusters via browser or command line tools, while applications use REST APIs or Thrift.

### Overview of enabling TLS manually

The typical process to enable wire encryption on Cloudera Data Platform Private Cloud clusters is described below.

#### Get Certificates

- Generate a public/private keypair on each host
- Generate the Certificate signing request (CSR) for all the hosts.
- Get the CSR signed by the company's internal Certificate Authority (CA).
- Generate keystore & truststore and deploy them across all the cluster hosts.

#### Cluster configuration

- For each service, enable TLS by setting the keystore and truststore configuration.
- Restart the affected components before proceeding to enable TLS for the next service.
- Make the required changes outside of the cluster manager's UI (like setting up truststore, Enabling Knox SSL, etc.)

#### Ongoing Maintenance

- For new service installation, the keystore and truststore information need to be configured for the service. Restart the impacted services.
- For each new host to be added to the cluster, admins would have to perform the steps from the "Get Certificates" section (only for the new hosts).
- The certificates are rotated before they expire.

### Auto-TLS feature in Cloudera Manager

The process described above can be a significant effort in large deployments, often leading to long deployment times and operational difficulties. The Auto-TLS feature automates all the steps required to enable TLS encryption at a cluster level. Using Auto-TLS, you can let Cloudera manage the Certificate Authority (CA) for all the certificates in the cluster or use the company's existing CA. In most cases, all the necessary steps can be enabled easily via the Cloudera Manager UI. This feature automates the following processes –

When Cloudera Manager is used as a Certificate Authority:

- Creates the root Certificate Authority or a Certificate Signing Request (CSR) for creating an intermediate Certificate Authority to be signed by company's existing Certificate Authority (CA)
- Generates the CSRs for hosts and signs them automatically

The following steps are always performed

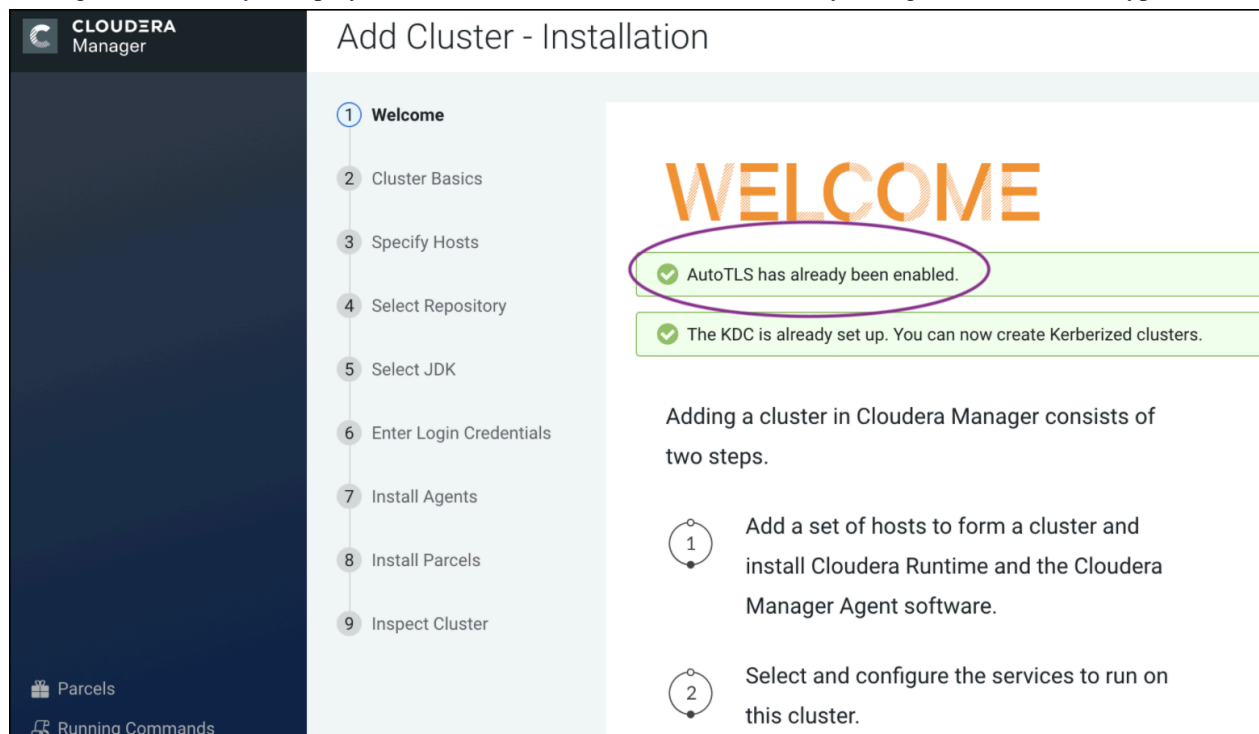
- Creates a keystore and truststore for hosts.
- Deploys the certificates, keystore and truststore to all the hosts in the cluster.
- All the cluster services are then automatically TLS enabled by configuring the keystore and truststore information from a role instance specific directory.
- Enables TLS for Cloudera Manager server and agents.
- After this initial setup, any new service, hosts (or) additional compute clusters setup are automatically TLS enabled by default.
- Provides an automation framework for rotating certificates.

Let us review these options with examples below on a CDP DC 7.1 cluster:

- Use case 1: Using Cloudera Manager to generate an internal CA and corresponding certificates
- Use case 2: Enabling Auto-TLS with an existing Root CA
- Use case 3: Enabling Auto-TLS with existing Certificates

## New Cluster deployment

With either of these options, you can reuse the existing TLS settings when creating a new cluster on a Cloudera Manager with Auto-TLS enabled. When you launch the wizard to create a new cluster you should see the following message. Now, when you deploy the cluster, all services will be automatically configured with wire encryption.



## Summary

The Auto-TLS functionality not only speeds up the initial setup of the wire encryption but also automates future TLS configuration steps for the cluster. The following table summarizes the differences between the options described in this blog.

Steps	HDP/EDH (manual)	CDP Private Cloud use case 1 - Using Cloudera Manager to generate an internal CA and corresponding certificates	CDP Private Cloud use case 2 - Enabling Auto-TLS with an existing Root CA	CDP Private Cloud use case 3 - Enabling Auto-TLS with Existing Certificates
Generate CSR	Manual	Automated	Automated	Manual
CSR Signed by CA	Manual	Automated	One-time	Manual
Deploy certificate to all hosts	Manual	Automated	Automated	Automated
Configuration for each service	Manual	Automated	Automated	Automated
Cluster restarts	Multiple	Once	Once	Once
Configuration steps	Manual	Automated	Automated	Automated
New Service steps	Manual	Automated	Automated	Automated
New Host cert. generation	Manual	Automated	Automated	Manual

The Auto-TLS feature significantly reduces the overhead of TLS management of your cluster, thus providing increased security with reduced operational overhead and helps you stay focused on your customers and their workloads.

### Related Information

[Use case 1: Use Cloudera Manager to generate internal CA and corresponding certificates](#)

[Use case 2: Enabling Auto-TLS with an intermediate CA signed by an existing Root CA](#)

[Use case 3: Enabling Auto-TLS with Existing Certificates](#)

## SAN Certificates

A SubjectAlternativeName (SAN) certificate is a certificate that uses the SubjectAlternativeName extension to associate the resulting certificate with multiple specific host names. SAN certificates are supported while using Auto-TLS only if custom certificates are generated. SAN certificates are not supported with the internal Cloudera Manager Certificate Authority.

## Configuring TLS Encryption for Cloudera Manager Using Auto-TLS

Use Auto-TLS to simplify the process of configuring TLS encryption for Cloudera Manager.

### About this task

The Auto-TLS feature automates all the steps required to enable TLS encryption at a cluster level. Using Auto-TLS, you can let Cloudera manage the Certificate Authority (CA) for all the certificates in the cluster or use the company's existing CA. In most cases, all the necessary steps can be enabled easily via the Cloudera Manager UI. This feature automates the following processes –

When Cloudera Manager is used as a Certificate Authority:

- Creates the root Certificate Authority or a Certificate Signing Request (CSR) for creating an intermediate Certificate Authority to be signed by company's existing Certificate Authority (CA)
- Generates the CSRs for hosts and signs them automatically

The following steps are always performed

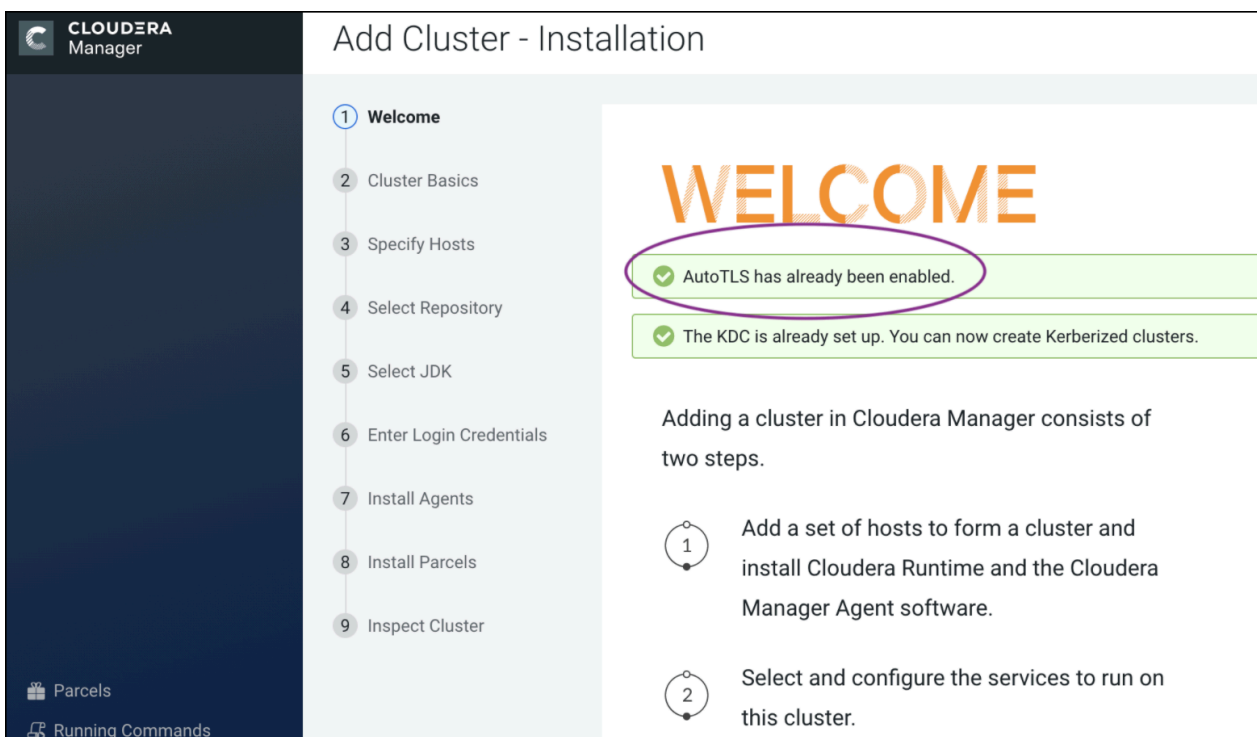
- Creates a keystore and truststore for hosts.
- Deploys the certificates, keystore and truststore to all the hosts in the cluster.
- All the cluster services are then automatically TLS enabled by configuring the keystore and truststore information from a role instance specific directory.
- Enables TLS for Cloudera Manager server and agents.
- After this initial setup, any new service, hosts (or) additional compute clusters setup are automatically TLS enabled by default.
- Provides an automation framework for rotating certificates.

Let us review these options with examples below on a CDP Private Cloud Base 7.1 cluster:

- Use case 1: Using Cloudera Manager to generate an internal CA and corresponding certificates
- Use case 2: Enabling Auto-TLS with an existing Root CA
- Use case 3: Enabling Auto-TLS with existing Certificates

### New Cluster deployment

With either of these options, you can reuse the existing TLS settings when creating a new cluster on a Cloudera Manager with Auto-TLS enabled. When you launch the wizard to create a new cluster you should see the following message. Now, when you deploy the cluster, all services will be automatically configured with wire encryption.



### Summary

The Auto-TLS functionality not only speeds up the initial setup of the wire encryption but also automates future TLS configuration steps for the cluster. The following table summarizes the differences between the available options.

Steps	HDP/EDH (manual)	CDP Private Cloud use case 1 - Using Cloudera Manager to generate an internal CA and corresponding certificates	CDP Private Cloud use case 2 - Enabling Auto-TLS with an existing Root CA	CDP Private Cloud use case 3 - Enabling Auto-TLS with Existing Certificates
Generate CSR	Manual	Automated	Automated	Manual
CSR Signed by CA	Manual	Automated	One-time	Manual
Deploy certificate to all hosts	Manual	Automated	Automated	Automated
Configuration for each service	Manual	Automated	Automated	Automated
Cluster restarts	Multiple	Once	Once	Once
Configuration steps	Manual	Automated	Automated	Automated
New Service steps	Manual	Automated	Automated	Automated
New Host cert. generation	Manual	Automated	Automated	Manual

The Auto-TLS feature significantly reduces the overhead of TLS management of your cluster, thus providing increased security with reduced operational overhead and helps you stay focused on your customers and their workloads.

### Related Information

[Manually Configuring TLS Encryption for Cloudera Manager](#)

## Auto-TLS Requirements and Limitations

Reference information for Auto-TLS requirements, limitations, and component support.

### About this task

For more info, see [Auto-TLS Requirements and Limitations](#).

## Rotating Auto-TLS Certificate Authority and Host Certificates

Your cluster security requirements may require that you rotate the auto-TLS CA and certificates.

### Using an internal CA (Use case 1)

1. Navigate to Administration Security . Click Rotate Auto-TLS Certificates to launch the wizard.
2. Complete the wizard.

### Using a custom CA (Use case 3)

1. Use the `/cm/commands/addCustomCerts` API command to replace the old certificates with new certificates in CMCA directory for each host. You must run this command for each host separately. An example of a curl command to upload the certificates to Cloudera Manager :

```
curl -u admin:admin -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "location": "/opt/cloudera/AutoTLS",
  "interpretAsFileNames": true,
  "hostCerts": [ {
    "hostname": "ccycloud-10.vcdp71.root.hwx.site",
    "certificate":
"/tmp/auto-tls/certs/ccycloud-10.vcdp71.root.hwx.site.pem",
    "key":
"/tmp/auto-tls/certs/ccycloud-10.vcdp71.root.hwx.site.pem"
  } ]
}' 'https://ccycloud-7.vcdp71.root.hwx.site:7183/api/v41/cm/commands/addCustomCerts'
```

In the example above, the "location" should be omitted if Auto-TLS was enabled or rotated after 7.1, and the file paths should point to files on the CM server host.

2. Use CM API `/hosts/{hostId}/commands/generateHostCerts` to deploy the new certificates to each host. You must run this command for each host separately. An example curl command :

```
curl -u admin:admin -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{ "sshPort" : 22, "userName" : "root", "password" : "cloudera" }'
'https://ccycloud-7.vcdp71.root.hwx.site:7183/api/v41/hosts/250e1bb7-8987-419c-a53f-c852c275d299/commands/generateHostCerts'
```

where '250e1bb7-8987-419c-a53f-c852c275d299' in the command above is the hostID.

## Auto-TLS Agent File Locations

### About this task

The certificates, keystores, and password files generated by auto-TLS are stored in `/var/lib/cloudera-scm-agent/agent-cert` on each Cloudera Manager Agent. The filenames are as follows:

**Table 1: Auto-TLS Agent Files**

Filename	Description
cm-auto-global_cacerts.pem	CA certificate and other trusted certificates in PEM format
cm-auto-global_truststore.jks	CA certificate and other trusted certificates in JKS format
cm-auto-in_cluster_ca_cert.pem	CA certificate in PEM format
cm-auto-in_cluster_truststore.jks	CA certificate in JKS format
cm-auto-host_key_cert_chain.pem	Agent host certificate and private key in PEM format
cm-auto-host_cert_chain.pem	Agent host certificate in PEM format
cm-auto-host_key.pem	Agent host private key in PEM format
cm-auto-host_keystore.jks	Agent host private key in JKS format
cm-auto-host_key.pw	Agent host private key password file

## Use case 1: Use Cloudera Manager to generate internal CA and corresponding certificates

Use Cloudera Manager to create and manage its own Certificate Authority.

1. To choose this option, from Cloudera Manager go to Administration Security (Status tab) Enable Auto-TLS . The Enable Auto-TLS page comes up.

The screenshot shows the Cloudera Manager interface. On the left is a dark sidebar with the Cloudera Manager logo and a search bar. Below the search bar are navigation links: Clusters, Hosts, Diagnostics, Audits, Charts, Replication, and Administration (which is highlighted with a blue bar). The main content area is titled 'Security' and has two tabs: 'Status' (selected) and 'Kerberos Credentials'. Under the 'Status' tab, there are three buttons: 'Enable Auto-TLS' (highlighted with a red box), 'TLS Settings', and 'Security Inspector'. Below these buttons is a text description: 'This page allows user to enable TLS (Transport Layer Security) related configuration set by the wizard.' There is also a search bar labeled 'Search by cluster name'. At the bottom, there is a table with columns 'Cluster', 'TLS', and 'Ke'.

2. In the Trusted CA Certificates Location field in the Generate CA section, enter the path to a PEM file on the Cloudera Manager host which contains a list of root CA certificates that should be imported into the truststores of all hosts. This is an optional field.

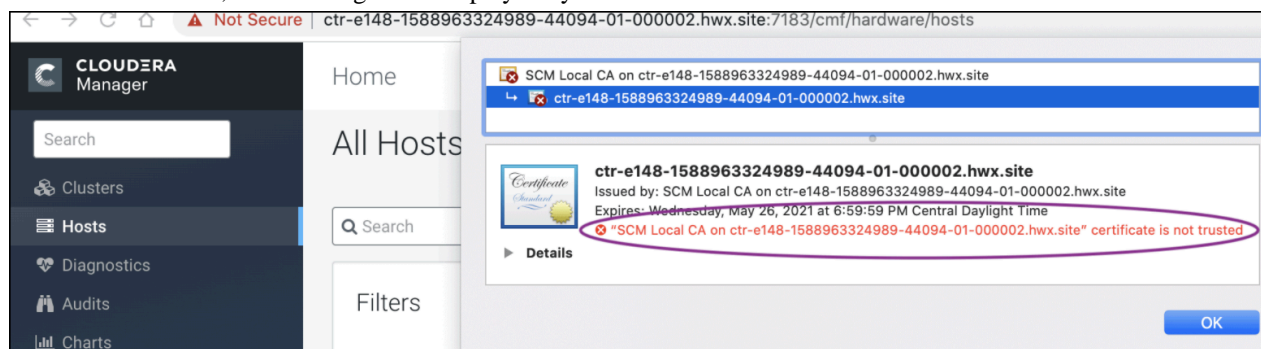
3. From the Enable TLS for: options, select All existing and future clusters to enable Auto-TLS for all existing and future clusters, or select Future clusters only to enable Auto-TLS for future clusters only.
4. Select the required SSH Username option. The available options are root and Another user.



**Note:** If Another user is selected, ensure that you specify the name of a user having passwordless sudo privileges.

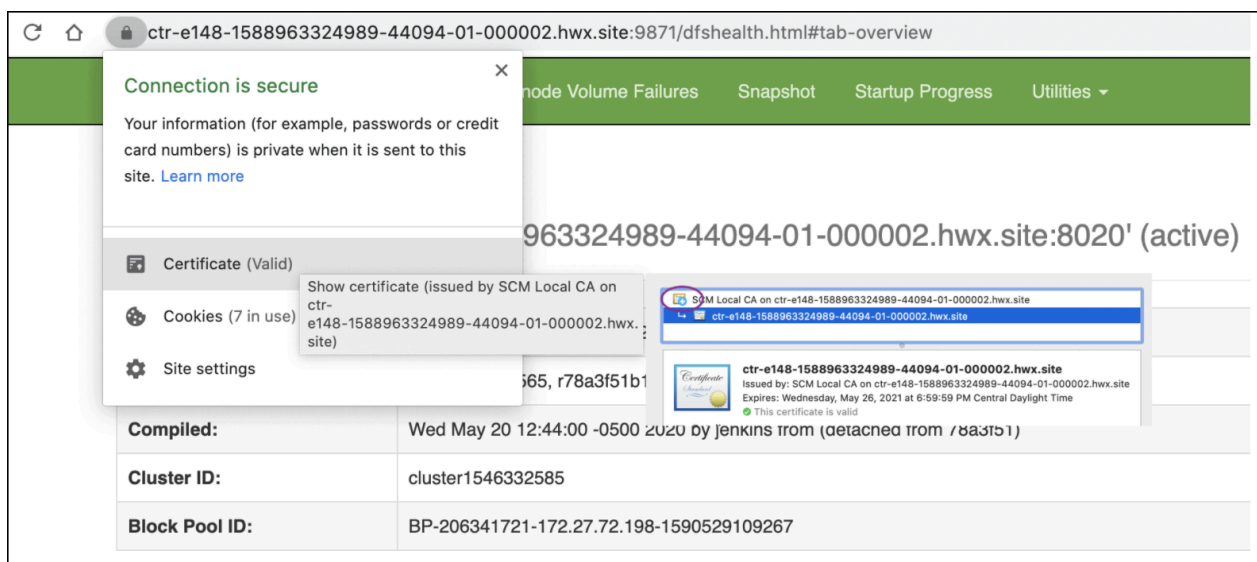
5. Select the required Authentication Method. You can either enable all hosts to accept the same password or you can enable all hosts to accept the same private key.
6. Enter the password in the Password field and verify the SSH Port number.
7. Click Next.

You are prompted to start Cloudera Manager, followed by Cloudera management services and any impacted clusters. When you start the Cloudera Manager server, you should see the UI at the TLS port 7183 by default. The browser displays a self-signed certificate from the SCM Local CA authority, as shown below. The browser displays a warning because it is not aware of the Root CA generated by Cloudera Manager. When the Root CA is imported into the client browser's truststore, this warning is not displayed by the browser.



When you set up the cluster, you should see a message stating that Auto-TLS is already enabled. Continue to install the required services. The whole cluster is TLS encrypted. Any new hosts or services are automatically configured. Here is an example of HDFS service with TLS encryption enabled by default (after trusting the root certificate generated by Cloudera Manager).





While this option is the simplest, it may not be suitable for some enterprise deployments where TLS certificates are issued by the company's existing Certificate Authority (CA) to maintain a centralized chain of trust.

### Rotate Auto-TLS Certificate Authority and Host Certificates

Your cluster security requirements may require that you rotate the Auto-TLS CA and certificates.

1. Navigate to Administration > Security.
2. Click the Rotate Auto-TLS Certificates button to launch the wizard.
3. Complete the wizard.

### Related Information

[Use case 2: Enabling Auto-TLS with an intermediate CA signed by an existing Root CA](#)

[Use case 3: Enabling Auto-TLS with Existing Certificates](#)

## Use case 2: Enabling Auto-TLS with an intermediate CA signed by an existing Root CA

You can make the Cloudera Manager CA an intermediate CA to an existing Root CA.



**Important:** You can apply Use Case 2 only to new Cloudera Manager installations that have not had hosts added or clusters created. If you already added hosts or created clusters, then you can implement only Use case 1 and Use case 3.

This is a three-step process. First, make Cloudera Manager generate a Certificate Signing Request (CSR). Second, have the CSR signed by the company's Certificate Authority (CA). Third, provide the signed certificate chain to continue the Auto-TLS setup. The following example demonstrates these three steps.

1. Initialize the certmanager with `--stop-at-csr` option before starting the Cloudera Manager:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk; /opt/cloudera/cm-agent/bin/certmanager --location /var/lib/cloudera-scm-server/certmanager setup --configure-services --stop-at-csr
```

For more information on the options available for certmanager, and how to use the certmanager's GenerateCMCA API, see "Certmanager Options - Using CM's GenerateCMCA API".

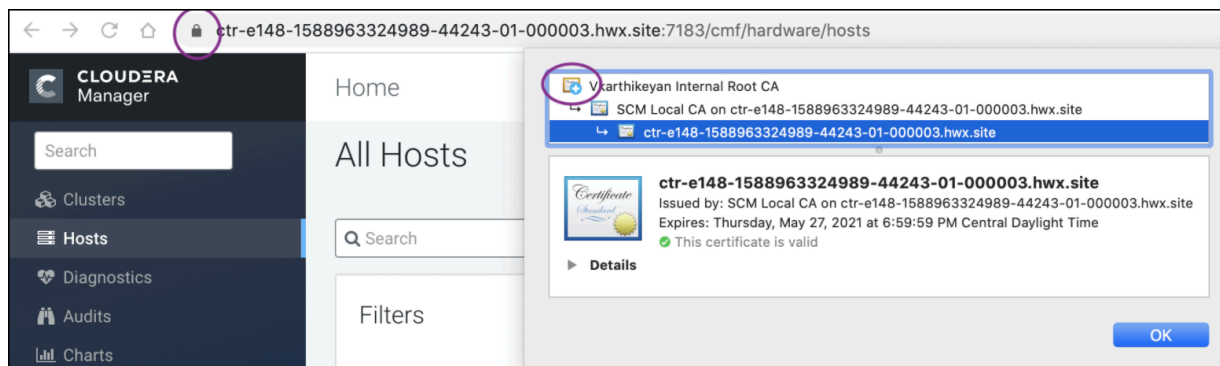
2. This generates a Certificate Signing Request (CSR) file at `/var/lib/cloudera-scm-server/certmanager/CMCA/private/ca_csr.pem`. If you examine the CSR closely, you notice the CSR request the necessary extension X509v3 Key Usage: critical Certificate Sign to sign certificates on its own.

3. Sign the `ca_csr.pem` file with your root CA certificate.
4. After you have the signed certificate, make sure that the certificate has the required extensions – X509v3 Basic Constraints: CA: TRUE and X509v3 Key Usage: Key Cert Sign.
5. Proceed with the installation with the following command:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk;
/opt/cloudera/cm-agent/bin/certmanager --location
/var/lib/cloudera-scm-server/certmanager setup
--configure-services --trusted-ca-certs ca-certs.pem
--signed-ca-cert=cm_cert_chain.pem.
X509v3 Basic Constraints: critical
CA:TRUE, pathlen:0
X509v3 Key Usage: critical
Digital Signature, Certificate Sign, CRL Sign
```

**Note:**

- `cm_cert_chain.pem` is a combination of the root CA certificate and the CA certificate that is generated by Cloudera Manager.
  - `--trusted-ca-certs` is an optional argument, and if it is given, then `ca-certs.pem` should point to a PEM-formatted file containing one or more root CA certificates.
6. Start Cloudera Manager on TLS port 7183. If the signed intermediate certificate is already imported into the client browser's truststore, then you should not see any warnings. In the screenshot below, "Vkarthikeyan Internal Root CA" is the root certificate. This certificate is already trusted by the system and has signed the Cloudera intermediate CA.



**Note:** In this use case, rotation of the Auto-TLS certificate authority is not supported. Cloudera recommends creating an intermediate CA with a long lifetime. The host certificates can be rotated by using the generate HostCerts API.

**Related Information**

[Use case 1: Use Cloudera Manager to generate internal CA and corresponding certificates](#)

[Use case 3: Enabling Auto-TLS with Existing Certificates](#)

[Certmanager Options - Using CM's GenerateCMCA API](#)

**Certmanager Options - Using CM's GenerateCMCA API**

This article describes how to use the certmanager's GenerateCMCA API. It generates the CMCA, which is an OpenSSL self-signed cert and CA directory. It creates and signs the certificate for the CM host. Creates an "internal" truststore with the CA certificate only. If trusted certificates were given, loads them into a "global" truststore. certmanager is part of the CM agent package.

**Name**

certmanager setup- generates CMCA and certificates for the host.

## Description

This command initializes the certmanager and setup the certificates for the CM server to run on this host, using those certificates.

The `--location` option (if any) given to certmanager will decide the location of the directory root where the certmanager will keep its files. This directory will contain sensitive files. It must be backed up and protected accordingly. This directory must not exist prior to calling this command, but must be create-able. (i.e. either its ancestors must exist, or must be create-able).

## Options

- `--config`*config-file-or-dir*

Path for configuration to use. If a directory, read and use all .ini files within it. If a file, must point to a config .ini file.

- `--override`*Section.Property=Value*

Override config file setting.

- `--hostname`*dns-or-ip*

Alternate name of local host (for SSL certificates). Only applies if CA type is 'internal'

- `--altname`*alt-name*

Alternate name of to add to generated SSL certificates. Multiple names may be supplied by repeating the option. Only applies if CA type is 'internal'

- `--write-cm-init` / `--skip-cm-init`

Writes a CM init file to set Auto-TLS related parameters. If disabled, only the CA directory will be created, and the init file contents will be printed to stdout.

- `--rotate` / `--no-rotate`

Rotates the CA keys and certificates. If disabled, the command fails if the CA directory already exists.

- `--configure-services` / `--no-configure-services`

Configure new services to use Auto-TLS certificates. If disabled, only agents will use TLS certificates.

- `--trusted-ca-cert`*trusted\_certs.pem*

Path to trusted CA certs bundle in PEM format. These certs will be imported into the truststore for all hosts.

- `--skip-invalid-ca-certs` / `--fail-invalid-ca-cert`

Whether to skip invalid CA certs in the trusted CA certs bundle. If false, setup will fail if any certs are invalid or duplicates.

- `--stop-at-csr` / `--dont-stop-at-csr`

Whether to stop at signing the CSR. If true, continue the setup by running setup and passing in `--signed-ca-cert`.

- `--signed-ca-cert`*signed\_ca\_chain.pem*

Path to signed CA cert chain. Pass this after running setup with the `--stop-at-csr` option.

- `--help`

Show this message and exit.

## How to customize CSR fields



**Important:** If you customize any of the CSR fields by using the “`--override`” option in an Auto-TLS enabled cluster, then post update, you must restart the [Cloudera Manager Server](#), [Cloudera Manager Agents](#), [Cloudera Management Service](#), and [Clusters](#).

You can customize the CSR fields by using the “`--override`” command-line option. The properties available for the `--override` parameter are mentioned below. An example of how to use the API follows this table.

Property	Default Value	Description
email_address	-	Additional subject alternate names to add to the certificate.
subject_suffix	-	The subject suffix to be appended to the CN.
ca_key_algo	rsa	CA Key encryption algorithm to be used. Valid values are "rsa", "dsa", "ec".
ca_key_args	3072	Integer value determining the number of bits for the key.
ca_sig_hash_algo	sha256	The hashing algorithm to use when signing.
ca_dn	-	The Subject DN to add to the CSR.
ca_expiration	5 years	"YYYYMMDD" format date for when certificate expires. Certificate expires at 23:59:59 on the given date at GMT time.
host_expiration	1 year	"YYYYMMDD" format date for when host expires.
ca_name	SCM Local CA on <host_fqdn>	Common Name to be used while generating the subject for the certificate.
host_key_algo	rsa	The asymmetric key algorithm to use to generate a CSR for a host.
host_key_args	3072	The parameter to the key algorithm (# bits, curve name, etc.) to generate a CSR for a host.
host_sig_hash_algo	sha256	The hashing algorithm to use in the signature when using the internal CA to sign the given host's CSR.
key_encryption_algo	aes256	The algo to use to encrypt the private key to generate a CSR for a host.

## Examples

This example shows how you can use `additionalArguments` property to pass any override parameters to the `certmanager` using `generateCMCA` API.

```
curl -X POST "https://tkarkera-1.tkarkera.root.hwx.site:7183/api/v45/cm/commands/generateCmca" -H "accept: application/json" -H "Content-Type: application/json" -d
{
  "sshPort": 22,
  "userName": "...",
  "password": "...",
  "privateKey": "...",
  "passphrase": "...",
  "location": "/opt/cloudera/CMCA",
  "customCA": false,
  "interpretAsFilenames": true,
  "cmHostCert": "host-cert.pem",
  "cmHostKey": "host-key.pem",
  "caCert": "ca-cert.pem",
  "keystorePasswd": "keystore.pw.txt",
  "truststorePasswd": "truststore.pw.txt",
  "trustedCaCerts": "cacerts.pem",
  "additionalArguments": [
    "--override",
    "ca_expiration=301010",
    "--override",
    "ca_key_args=4096",
    "--override",
```

```

    "host_key_args=4096"
  ],
  "hostCerts": [
    {
      "hostname": "...",
      "certificate": "host-cert.pem",
      "key": "host-key.pem",
      "subjectAltNames": [
        "DNS:example.cloudera.com",
        "..."
      ]
    },
    {
      "hostname": "...",
      "certificate": "...",
      "key": "...",
      "subjectAltNames": [
        "...",
        "DNS:example.cloudera.com"
      ]
    }
  ],
  "configureAllServices": true
}

```

### Use case 3: Enabling Auto-TLS with Existing Certificates

You can manually generate the certificates signed by an existing Root CA and upload them to Cloudera Manager

If you have an existing cluster where you need to enable Auto-TLS, or if there is a need to get the host certificates signed individually by the company's existing CA, you can use this option of enabling Auto-TLS with existing certificates. This option adds operational overhead of generating certificates for any new hosts and uploading to Cloudera Manager through an API request. In this option, certificates signed by CA are staged and Auto-TLS is enabled by calling a Cloudera Manager API.

1. Create the Auto-TLS directory `/opt/cloudera/AutoTLS` in the Cloudera Manager server. The directory must be owned by the `cloudera-scm` user.
2. Create a public/private key for each host and generate the corresponding Certificate Signing request (CSR). Have these CSRs signed by the company's Certificate Authority (CA). You can generate private keys and CSRs by using your existing PKI tools and processes, or manually with common utilities like `keytool` or `openssl`. In this example using `openssl`, the private key and CSR files are located under the `/tmp/auto-tls` directory. The password used for the private key is stored in `key.pwd`.

```

openssl req -newkey rsa:4096 -sha256 -days 356 \
-keyout /tmp/auto-tls/keys/host1.example.com-key.pem \
-out /tmp/auto-tls/host1.example.com.csr \
-passout file:/tmp/auto-tls/keys/key.pwd \
-subj '/CN=host1.example.com, O=Cloudera Test, C=US' \
-extensions san -config <( echo '[req]'; echo 'distinguished_name=req';
echo 'req_extensions=san';echo '[san]'; echo 'subjectAltName=DNS:host1.
example.com DNS:loadbalancer.example.com')

```

The same procedure is used for all cluster hosts.

3. Prepare all the certificates signed by the company's CA on the Cloudera Manager server. In this example, all the certificates are located under the `/tmp/auto-tls` directory. The password used for keystore and truststore are present in `key.pwd` and `truststore.pwd` files respectively.

4. Refer the example API given below. Customize this API to match the deployment that has been set up and then execute the API.

```
curl -i -v -uadmin:admin -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' -d '{
  "location" : "/opt/cloudera/AutoTLS",
  "customCA" : true,
  "interpretAsFileNames" : true,
  "cmHostCert" : "/tmp/auto-tls/certs/ccycloud-7.vcdp71.root.hwx.site.pem",
  "cmHostKey" : "/tmp/auto-tls/keys/ccycloud-7.vcdp71.root.hwx.site-key.pem",
  "caCert" : "/tmp/auto-tls/ca-certs/cfssl-chain-truststore.pem",
  "keystorePasswd" : "/tmp/auto-tls/keys/key.pwd",
  "truststorePasswd" : "/tmp/auto-tls/ca-certs/truststore.pwd",
  "trustedCaCerts" : "/tmp/auto-tls/ca-certs.pem", //This is a path to a PEM
  file on the Cloudera Manager host which contains
  a list of CA certificates that should be imported into the truststores of
  all hosts. This is an optional field.
  "hostCerts" : [ {
    "hostname" : "ccycloud-7.vcdp71.root.hwx.site",
    "certificate" : "/tmp/auto-tls/certs/ccycloud-7.vcdp71.root.hwx.site.pem",
    "key" : "/tmp/auto-tls/keys/ccycloud-7.vcdp71.root.hwx.site-key.pem"
  }, {
    "hostname" : "ccycloud-3.vcdp71.root.hwx.site",
    "certificate" : "/tmp/auto-tls/certs/ccycloud-3.vcdp71.root.hwx.site.pem",
    "key" : "/tmp/auto-tls/keys/ccycloud-3.vcdp71.root.hwx.site-key.pem"
  }, {
    "hostname" : "ccycloud-2.vcdp71.root.hwx.site",
    "certificate" : "/tmp/auto-tls/certs/ccycloud-3.vcdp71.root.hwx.site.pem",
    "key" : "/tmp/auto-tls/keys/ccycloud-3.vcdp71.root.hwx.site-key.pem"
  }, {
    "hostname" : "ccycloud-1.vcdp71.root.hwx.site",
    "certificate" : "/tmp/auto-tls/certs/ccycloud-1.vcdp71.root.hwx.site.pem",
    "key" : "/tmp/auto-tls/keys/ccycloud-1.vcdp71.root.hwx.site-key.pem"
  } ],
  "configureAllServices" : "true",
  "sshPort" : 22,
  "userName" : "root",
  "password" : "cloudera"
}' 'http://ccycloud-7.vcdp71.root.hwx.site:7180/api/v41/cm/commands/gener
ateCmca' //This link is valid if you have
not enabled TLS in the Cloudera Manager UI. If you enable TLS for the same
deployment in the Cloudera Manager UI later,
the port number and the protocol changes for the API calls and for acces
sing the link from a browser. In such a scenario,
```

the correct API call is as follows: `https://ccycloud-7.vcdp71.root.hwx.site:7183/api/v41/cm/commands/generateCmca.`

If a new deployment is set up without TLS encryption, the API uses HTTP and port 7180. If you are converting the deployment to an Auto-TLS setup from an existing Manual TLS setup, the Cloudera Manager UI is converted to HTTPS. In such cases, the URL for the API calls has to be modified.



**Note:** If you need to use an SSH private key instead of a password, then replace "password" in the above example with "privateKey" and provide the SSH private key as an argument to that field. The SSH private key must be properly JSON-encoded, including replacing newlines with '\n'. The following awk command will output to the terminal the contents of `~/.ssh/id_rsa` with newlines replaced, and can be used as input to the "privateKey" argument:

```
awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' ~/.ssh/id_rsa
```

**Table 2: JSON file key properties**

Property	Data type	Description
customCA	boolean	Option to generate an internal Cloudera Manager CA (false) or use user-provided certificates (true). When set to true (user-provided certificates), the following other arguments must be given: * cmHostCert * cmHostKey * caCert * keystorePasswd * truststorePasswd
cmHostCert	string	The certificate for the Cloudera Manager host in PEM format. Only used if customCA == true.
cmHostKey	string	The private key for the Cloudera Manager host in PEM format. Only used if customCA == true.
caCert	string	The certificate for the user-provided certificate authority in PEM format. Only used if customCA == true.
trustedCaCerts	string	A list of CA certificates that will be imported into the Auto-TLS truststore and distributed to all hosts.

- When this API returns successfully, you should see the recent command run as follows.

Recent Commands				
Enable Auto-TLS		Show Failed Only		30m 1h 2h 6h 12h 1d 7d 30d
Command	Context	Start Time	Duration	Actions
Enable Auto-TLS		May 24, 7:46:20 PM	16.02s	
1 - 1 of 1				

- When this API is executing you can check `/var/log/cloudera-scm-server/cloudera-scm-server.log` for API logs.
- Restart the Cloudera Manager service. Then restart the Cloudera Manager agents on all cluster servers.
- The Cloudera Manager UI/API is now available on the TLS port. Now restart all the Cloudera Manager management services.
- Restart the Cluster services. Now all the services are configured for wire encryption.

10. When adding new hosts to this cluster, the following additional steps need to be performed to upload the CA signed host certificates to Cloudera Manager.
- The add hosts wizard will prompt the following screen with instructions to upload the certificates.

### Add Hosts to Cluster: base

- 1 Setup Auto-TLS
- 2 Specify Hosts
- 3 Select Repository
- 4 Select JDK
- 5 Enter Login Credentials
- 6 Install Agents
- 7 Install Parcels
- 8 Inspect Hosts for Correctness
- 9 Select Host Template
- 10 Command Details

#### Setup Auto-TLS

You have already successfully set up the certificate manager for Auto-TLS.

If you used Cloudera Manager to generate an internal Certificate Authority and its corresponding certificates when you initially enabled Auto-TLS, click **Continue** to proceed. The certificates and keys will be created automatically.

If you used an existing Certificate Authority and its corresponding certificates when you initially enabled Auto-TLS, you must add certificates using the Cloudera Manager API. The filenames must include the full path to the files on the Cloudera Manager server. The cloudera-scm user must have read access to those paths. Here is an example command:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "location": "",
  "interpretAsFilenames": true,
  "hostCerts": [
    {
      "hostname": "host1",
      "certificate": "/var/certs/host1-cert.pem",
      "key": "/var/certs/host1-key.pem"
    },
    {
      "hostname": "...",
      "certificate": "...",
      "key": "..."
    }
  ]
}'
```

- Upload the certificates to Cloudera Manager using the following example command:

```
curl -u admin:admin -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{
  "location": "/opt/cloudera/AutoTLS",
  "interpretAsFilenames": true,
  "hostCerts": [ {
    "hostname": "ccycloud-10.vcdp71.root.hwx.site",
    "certificate":
"/tmp/auto-tls/certs/ccycloud-10.vcdp71.root.hwx.site.pem",
    "key":
"/tmp/auto-tls/certs/ccycloud-10.vcdp71.root.hwx.site.pem"
  } ]
}' 'https://ccycloud-7.vcdp71.root.hwx.site:7183/api/v41/cm/commands/addCustomCerts'
```

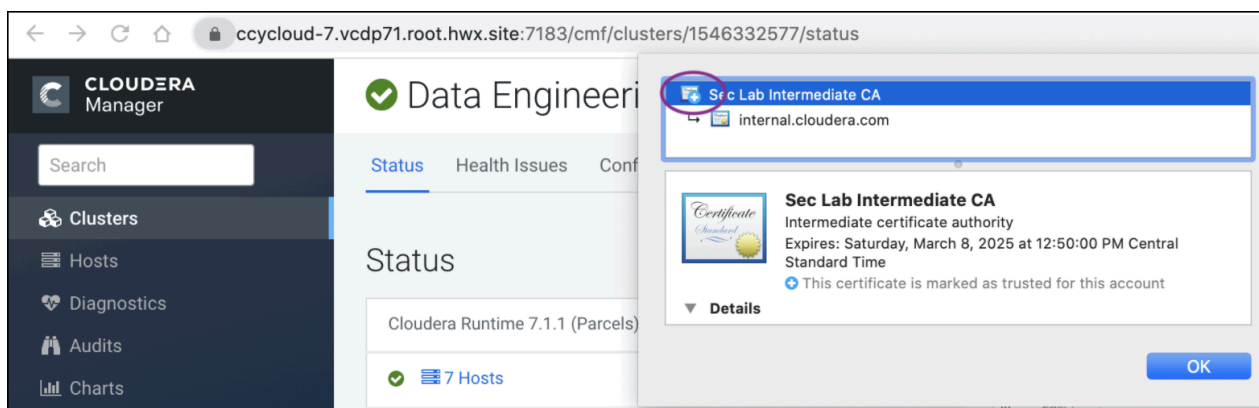
In the curl command example above, the "location" should be omitted if Auto-TLS was enabled or rotated after 7.1, and the file paths should point to files on the CM server host.

- Continue to add hosts.

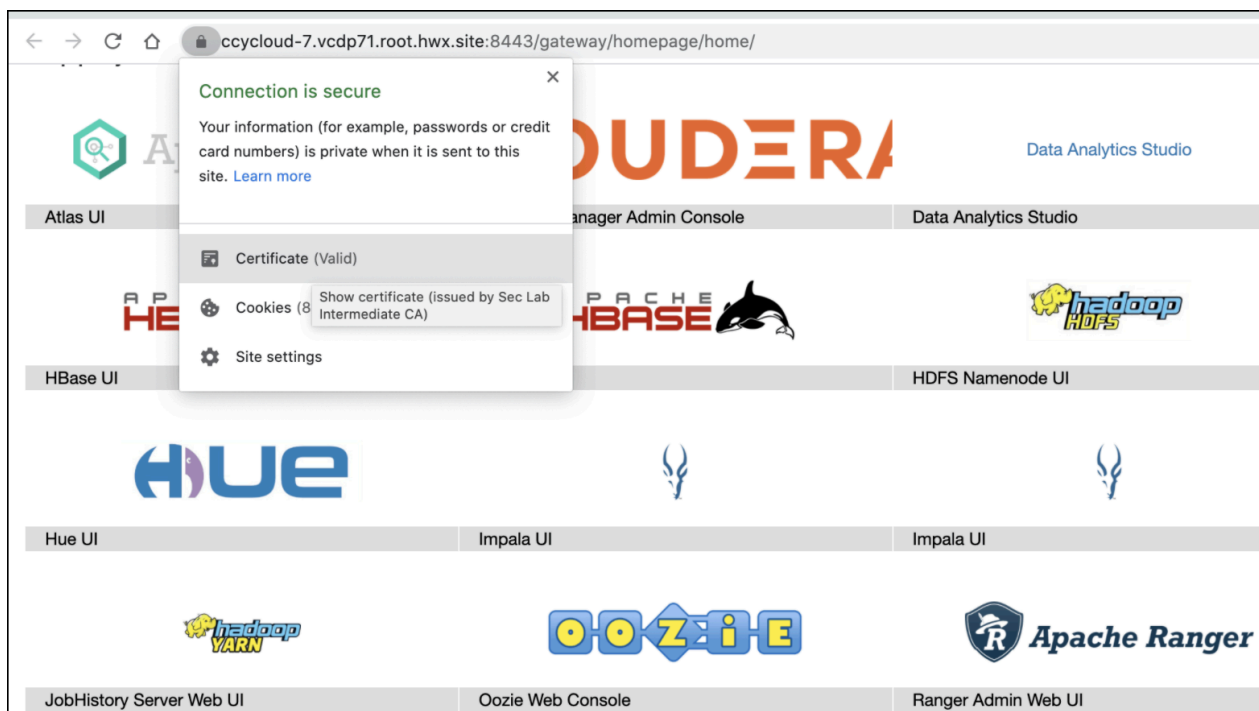
In this example, the CA used to sign all the certificates is Sec Lab Intermediate CA which can be found in the screenshot below:

Cloudera Manager UI:





Knox UI:



### Rotate Auto-TLS Certificate Authority and Host Certificates

After the certificate files in the specified paths have been replaced with the new certificates, run the API calls that were used when enabling Auto-TLS. Refer Step 4 in this use case to run the Cloudera Manager API. You do not need to run the `addCustomCerts` API if you are performing the steps given in this use case.

### Related Information

Use case 1: Use Cloudera Manager to generate internal CA and corresponding certificates

Use case 2: Enabling Auto-TLS with an intermediate CA signed by an existing Root CA

## Manually Configuring TLS Encryption for Cloudera Manager

How to manually enable TLS encryption and certificate authentication for Cloudera Manager.

### About this task



**Note:** Cloudera recommends using auto-TLS to configure TLS encryption for Cloudera Manager and CDP components.

The Auto-TLS feature automates all the steps required to enable TLS encryption at a cluster level. Using Auto-TLS, you can let Cloudera manage the Certificate Authority (CA) for all the certificates in the cluster or use the company's existing CA. In most cases, all the necessary steps can be enabled easily via the Cloudera Manager UI. This feature automates the following processes –

For instructions on enabling auto-TLS, see “Configuring TLS Encryption for Cloudera Manager Using Auto-TLS”.

When you configure authentication and authorization on a cluster, Cloudera Manager Server sends sensitive information over the network to cluster hosts, such as Kerberos keytabs and configuration files that contain passwords. To secure this transfer, you must configure TLS encryption between Cloudera Manager Server and all cluster hosts.

TLS encryption is also used to secure client connections to the Cloudera Manager Admin Interface, using HTTPS.

Cloudera Manager also supports TLS authentication. Without certificate authentication, a malicious user can add a host to Cloudera Manager by installing the Cloudera Manager Agent software and configuring it to communicate with Cloudera Manager Server. To prevent this, you must install certificates on each agent host and configure Cloudera Manager Server to trust those certificates.

This guide shows how to configure and enable TLS encryption and certificate authentication for Cloudera Manager. The provided examples use an internal certificate authority (CA) to sign all TLS certificates, so this guide also shows you how to establish trust with the CA. (For certificates signed by a trusted public CA, establishing trust is not necessary, because the Java Development Kit (JDK) already trusts them.)

### Related Information

[Configuring TLS Encryption for Cloudera Manager Using Auto-TLS](#)

## Generate TLS Certificates

### About this task

The following procedure assumes that an internal certificate authority (CA) is used, and shows how to establish trust for that internal CA. If you are using a trusted public CA (such as Symantec, GeoTrust, Comodo, and others), you do not need to explicitly establish trust for the issued certificates, unless you are using an older JDK and a newer public CA. Older JDKs might not trust newer public CAs by default.

### On Each Cluster Host:

#### About this task

Complete the following steps on each cluster host, including the Cloudera Manager Server host.

#### Procedure

1. Configure your environment to set JAVA\_HOME to the Oracle JDK. For example:

```
export JAVA_HOME=/usr/java/jdk1.8.0_141-cloudera
```

If you log out of the host before completing this procedure, make sure to set JAVA\_HOME again when you log in to complete the steps.

2. Create the `/opt/cloudera/security/pki` directory:

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

If you choose to use a different directory, make sure you use the same directory on all cluster hosts to simplify management and maintenance.

3. Use the `keytool` utility to generate a Java keystore and certificate signing request (CSR). Replace the OU, O, L, ST, and C entries with the values for your environment. When prompted, use the same password for the keystore password and key password. Cloudera Manager does not support using different passwords for the key and keystore.

```
$JAVA_HOME/bin/keytool -genkeypair -alias $(hostname -f) -keyalg
RSA -keystore /opt/cloudera/security/pki/$(hostname -f).jks -keysiz
e 2048 -dname "CN=$(hostname -f),OU=Engineering,O=Cloudera,L=Palo
Alto,ST=California,C=US" -ext san=dns:$(hostname -f)
```

```
$JAVA_HOME/bin/keytool -certreq -alias $(hostname -f) -keystore /opt/clo
udera/security/pki/$(hostname -f).jks -file /opt/cloudera/security/pki/$
(hostname -f).csr -ext san=dns:$(hostname -f) -ext EKU=serverAuth,client
Auth
```



**Note:** You must ensure that your Issuing Authority will issue the certificates with the extensions CDP requires.

4. Submit the CSR files (for example, `cm01.example.com.csr`) to your certificate authority to obtain a server certificate.

For security purposes, many commercial CAs ignore requested extensions in a CSR. Make sure that you inform the CA that you require certificates with both server and client authentication options.

If possible, obtain the certificate in PEM (Base64 ASCII) format. The certificate file is in PEM format if it looks similar to this (some lines omitted):

```
-----BEGIN CERTIFICATE-----
MIIDAzCCAesCAQAwY0xCzAJBgNVBAYTAlVTMRMwEQYDVQQIEWpDYWxpZm9ybm1h
MRIwEAYDVQQHEwlQYWxvIEFsdG8xETAPBgNVBAoTCENsb3VhZm9ybm1hMRQwEgYDVQQL
...
tudY0C32LjGjW0g5ALlIn9Oy1u2xRKGAVfapbzAZ2rchtlCZc7mtaT6BXgW8S+Db
0HhuObn1/8TL4Ho9G+KlJB3MWik2oEbOvQt0rBidMr9qaNX86m0i7pouXZelZ5c5
UnDPtrhW6A==
-----END CERTIFICATE-----
```

If your issued certificate is in binary (DER) format, convert it to PEM format.

5. After you have received the signed certificate, copy the signed certificate to the following location:

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

6. Inspect the signed certificate to verify that both server and client authentication options are present, as well as the subject alternative name:

```
openssl x509 -in /opt/cloudera/security/pki/$(hostname -f).pem -noout -t
ext
```

Look for output similar to the following to verify the server and client authentication options:

```
X509v3 Extended Key Usage:
```

```
ion          TLS Web Server Authentication, TLS Web Client Authentica
```

Look for output similar to the following to validate the subject alternative name:

```
X509v3 Subject Alternative Name:
      DNS:hostname.example.com
```



**Important:**

If the certificate does not have the DNS field, re-submit the CSR to the CA, and request that they generate a certificate that keeps the Subject Alternative Name field intact.

If the certificate does not have both TLS Web Server Authentication and TLS Web Client Authentication listed in the X509v3 Extended Key Usage section, re-submit the CSR to the CA, and request that they generate a certificate that can be used for both server and client authentication.

7. Copy the root and intermediate CA certificates to `/opt/cloudera/security/pki/rootca.pem` and `/opt/cloudera/security/pki/intca.pem` on each host. If you have a concatenated file containing the root CA and an intermediate CA certificate, split the file along the `END CERTIFICATE/BEGIN CERTIFICATE` boundary into individual files. If there are multiple intermediate CA certificates, use unique file names such as `intca-1.pem`, `intca-2.pem`, and so on.
8. Copy the JDK cacerts file to `jssecacerts` as follows:

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



**Note:** The default password for the cacerts file is `changeit`. The same applies to the `jssecacerts` file if you copied it from the cacerts before changing the password. Cloudera recommends changing these passwords by running the following commands:

```
$JAVA_HOME/bin/keytool -storepasswd -keystore $JAVA_HOME/jre/lib/sec
urity/cacerts
```

```
$JAVA_HOME/bin/keytool -storepasswd -keystore $JAVA_HOME/jre/lib/sec
urity/jssecacerts
```

The Oracle JDK uses the `jssecacerts` file for its default truststore if it exists. Otherwise, it uses the `cacerts` file. Creating the `jssecacerts` file allows you to trust an internal CA without modifying the `cacerts` file that is included with the JDK.



**Note:** If you upgrade your JDK, make sure to copy your existing `jssecacerts` file to the new JDK (under `$JAVA_HOME/jre/lib/security`).

9. Import the root CA certificate into the JDK truststore.

```
sudo $JAVA_HOME/bin/keytool -importcert -alias rootca -keystore $JAVA_HO
ME/jre/lib/security/jssecacerts -file /opt/cloudera/security/pki/rootca.
pem
```

If you see a message like the following, enter `yes` to continue:

```
Trust this certificate? [no]: yes
```

You must see the following response verifying that the certificate has been properly imported:

```
Certificate was added to keystore
```

10. Perform these steps to replace the self-signed cert with CA issued cert.

```
cd /opt/cloudera/security/pki/; mv $(hostname -f).jks $(hostname -f)-orig.jks
```

```
keytool -importkeystore -srcstoretype JKS -deststoretype PKCS12 -srckeystore $(hostname -f)-orig.jks -srcalias $(hostname -f) -destkeystore $(hostname -f).p12
```

```
openssl pkcs12 -in $(hostname -f).p12 -nodes -nocerts -out $(hostname -f)-pk.pem
```

```
openssl pkcs12 -export -in $(hostname -f).pem -inkey $(hostname -f)-pk.pem -name $(hostname -f) -out $(hostname -f).pk12
```

```
keytool -importkeystore -deststoretype JKS -srcstoretype PKCS12 -srckeystore $(hostname -f).pk12 -destkeystore $(hostname -f).jks
```

11. Append the intermediate CA certificate to the signed host certificate, and then import it into the keystore. Make sure that you use the append operator (>>) and not the overwrite operator (>):

```
sudo cat /opt/cloudera/security/pki/intca.pem >> /opt/cloudera/security/pki/$(hostname -f).pem
```

12. Create symbolic links (symlink) for the certificate and keystore files:

```
sudo ln -s /opt/cloudera/security/pki/$(hostname -f).pem /opt/cloudera/security/pki/agent.pem
```

This allows you to use the same /etc/cloudera-scm-agent/config.ini file on all agent hosts rather than maintaining a file for each agent.

## On the Cloudera Manager Server Host

### About this task

On the Cloudera Manager Server host, create an additional symlink for the keystore file:

```
sudo ln -s /opt/cloudera/security/pki/$(hostname -f).jks /opt/cloudera/security/pki/server.jks
```

## Configure TLS for the Cloudera Manager Admin Console

### About this task

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

Use the following procedure to enable TLS encryption for the Cloudera Manager Server admin interface. Make sure you have generated the host certificate as described in [On Each Cluster Host](#): on page 26.

## Step 1: Enable HTTPS for the Cloudera Manager Admin Console

### Procedure

1. Log in to the Cloudera Manager Admin Console.
2. Select Administration Settings .
3. Select the Security category.
4. Configure the following TLS settings:

Property	Description
Cloudera Manager TLS/SSL Server JKS Keystore File Location	The complete path to the keystore file. For example:  <code>/opt/cloudera/security/pki/server.jks</code>
Cloudera Manager TLS/SSL Server JKS Keystore File Password	The password for the /opt/cloudera/security/jks/server.jks keystore.
Use TLS Encryption for Admin Console	Check this box to enable TLS encryption for Cloudera Manager.

5. Enter a Reason for Change, then click Save Changes to save the settings.

## Step 2: Specify SSL Truststore Properties for Cloudera Management Services

### About this task

When enabling TLS for the Cloudera Manager Server admin interface, you must set the Java truststore location and password in the Cloudera Management Services configuration. Otherwise, roles such as Host Monitor and Service Monitor cannot connect to Cloudera Manager Server and will not start.

Configure the path and password for the \$JAVA\_HOME/jre/lib/security/jssecacerts truststore that you created earlier. Make sure that you have created this file on all hosts, including the Cloudera Management Service hosts, as instructed in [On Each Cluster Host](#): on page 26.

### Procedure

1. Open the Cloudera Manager Administration Console and go to the Cloudera Management Service service.
2. Click the Configuration tab.
3. Select Scope Cloudera Management Service (Service-Wide) .
4. Select Category Security .
5. Edit the following TLS/SSL properties according to your cluster configuration.

Property	Description
TLS/SSL Client Truststore File Location	The path to the client truststore file used in HTTPS communication. This truststore contains certificates of trusted servers, or of Certificate Authorities trusted to identify servers. For this example, set the value to:  <code>&lt;JAVA_HOME&gt;/jre/lib/security/jssecacerts</code>  Replace <JAVA_HOME> with the path to the Oracle JDK.
Cloudera Manager Server TLS/SSL Certificate Trust Store Password	The password for the truststore file.

6. Enter a Reason for Change, then click Save Changes to save the settings.

## Step 3: Restart Cloudera Manager and Services

### About this task

You must restart both Cloudera Manager Server and the Cloudera Management Service for TLS encryption to work. Otherwise, the Cloudera Management Services (such as Host Monitor and Service Monitor) cannot communicate with Cloudera Manager Server.

### Procedure

1. Restart the Cloudera Manager Server by running the following command on the Cloudera Manager Server host:

- RHEL 7 compatible:

```
sudo systemctl restart cloudera-scm-server
```

- RHEL 6 compatible, SLES, Ubuntu:

- ```
sudo service cloudera-scm-server restart
```

2. After the restart completes, connect to the Cloudera Manager Admin Console using the HTTPS URL (for example: <https://cm01.example.com:7183>). If you used an internal CA-signed certificate, you must configure your browser to trust the certificate. Otherwise, you will see a warning in your browser any time you access the Cloudera Manager Administration Console. By default, certificates issued by public commercial CAs are trusted by most browsers, and no additional configuration is necessary if your certificate is signed by one of them.
3. Restart the Cloudera Management Service ( Cloudera Management Service Actions Restart ).

## Configure TLS for Cloudera Manager Agents

### About this task

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

Use the following procedure to encrypt the communication between Cloudera Manager Server and Cloudera Manager Agents:

## Step 1: Enable TLS Encryption for Agents in Cloudera Manager

### About this task

Configure the TLS properties for Cloudera Manager Agents.

### Procedure

1. Log in to the Cloudera Manager Admin Console.
2. Select Administration Settings .
3. Select the Security category.
4. Select the Use TLS Encryption for Agents option.
5. Enter a Reason for Change, then click Save Changes to save the settings.

## Step 2: Enable TLS on Cloudera Manager Agent Hosts

### About this task

To enable TLS between the Cloudera Manager agents and Cloudera Manager, you must specify values for the TLS properties in the `/etc/cloudera-scm-agent/config.ini` configuration file on all agent hosts.

### Procedure

- On each agent host (including the Cloudera Manager Server host, which also has an agent), open the `/etc/cloudera-scm-agent/config.ini` configuration file and set the `use_tls` parameter in the `[Security]` section as follows:

```
use_tls=1
```

Alternatively, you can edit the `config.ini` file on one host, and then copy it to the other hosts because this file by default does not contain host-specific information. If you have modified properties such as `listening_hostname` or `listening_ip` address in `config.ini`, you must edit the file individually on each host.

## Step 3: Restart Cloudera Manager Server and Agents

### Procedure

- Restart the Cloudera Manager Server by running the following command on the Cloudera Manager Server host:

- RHEL 7 compatible:

```
sudo systemctl restart cloudera-scm-server
```

- RHEL 6 compatible, SLES, Ubuntu:

- ```
sudo service cloudera-scm-server restart
```

- On each agent host (including the Cloudera Manager Server host), restart the Cloudera Manager agent service:

- RHEL 7 compatible:

```
sudo systemctl restart cloudera-scm-agent
```

- RHEL 6 compatible, SLES, Ubuntu:

- ```
sudo service cloudera-scm-agent restart
```

## Step 4: Verify that the Cloudera Manager Server and Agents are Communicating

### About this task

In the Cloudera Manager Admin Console, go to **Hosts All Hosts**. If you see successful heartbeats reported in the **Last Heartbeat** column after restarting the agents, TLS encryption is working properly.

## Enable Server Certificate Verification on Cloudera Manager Agents

### About this task

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

If you have completed the previous sections, communication between Cloudera Manager server and the agents is encrypted, but the certificate authenticity is not verified. For full security, you must configure the agents to verify the Cloudera Manager server certificate. If you are using a server certificate signed by an internal certificate authority (CA), you must configure the agents to trust that CA:



### Procedure

1. On each agent host (including the Cloudera Manager Server host), open the `/etc/cloudera-scm-agent/config.ini` configuration file, and then uncomment and set the following property:

```
verify_cert_file=/opt/cloudera/security/pki/rootca.pem
```

Alternatively, you can edit the `config.ini` file on one host, and then copy it to the other hosts because this file by default does not contain host-specific information. If you have modified properties such as `listening_hostname` or `listening_ip` address in `config.ini`, you must edit the file individually on each host.

2. Restart the Cloudera Manager agents. On each agent host (including the Cloudera Manager Server host), run the following command:

- RHEL 7 compatible:

```
sudo systemctl restart cloudera-scm-agent
```

- RHEL 6 compatible, SLES, Ubuntu:

- ```
sudo service cloudera-scm-agent restart
```

3. Verify that the Cloudera Manager server and agents are communicating. In the Cloudera Manager Admin Console, go to **Hosts All Hosts**. If you see successful heartbeats reported in the **Last Heartbeat** column after restarting the agents and management service, TLS verification is working properly. If not, check the agent log (`/var/log/cloudera-scm-agent/cloudera-scm-agent.log`) for errors.

## Configure Agent Certificate Authentication

### About this task



**Important:** Perform this procedure on each agent host, including the Cloudera Manager Server host, which also has an agent.

Without certificate authentication, a malicious user can add a host to Cloudera Manager by installing the Cloudera Manager agent software and configuring it to communicate with Cloudera Manager Server. To prevent this, you must configure Cloudera Manager to trust the agent certificates.

### Step 1: Export the Private Key to a File

#### About this task

On each Cloudera Manager Agent host, use the `keytool` utility to export the private key and certificate to a PKCS12 file, which can then be split into individual key and certificate files using the `openssl` command:

#### Procedure

1. Export the private key and certificate:

```
sudo $JAVA_HOME/bin/keytool -importkeystore -srckeystore /opt/cloudera/security/pki/$(hostname -f).jks -destkeystore /opt/cloudera/security/pki/$(hostname -f)-key.p12 -deststoretype PKCS12 -srcalias $(hostname -f)
```

2. Use the `openssl` command to export the private key into its own file:

```
sudo openssl pkcs12 -in /opt/cloudera/security/pki/$(hostname -f)-key.p12 -nocerts -out /opt/cloudera/security/pki/$(hostname -f).key
```

3. Create a symbolic link for the .key file:

```
sudo ln -s /opt/cloudera/security/pki/$(hostname -f).key /opt/cloudera/security/pki/agent.key
```

This allows you to use the same `/etc/cloudera-scm-agent/config.ini` file on all agent hosts rather than maintaining a file for each agent.

## Step 2: Create a Password File

### About this task

The Cloudera Manager agent obtains the password from a text file, not from a command line parameter or environment variable. The password file allows you to use file permissions to protect the password. For example, run the following commands on each Cloudera Manager Agent host, or run them on one host and copy the file to the other hosts:

Create and secure the file containing the password used to protect the private key of the Agent:

### Procedure

1. Use a text editor to create a file called `/etc/cloudera-scm-agent/agentkey.pw` that contains the password.
2. Change ownership of the file to root:

```
sudo chown root:root /etc/cloudera-scm-agent/agentkey.pw
```

3. Change the permissions of the file:

```
sudo chmod 440 /etc/cloudera-scm-agent/agentkey.pw
```

## Step 3: Configure the Agent to Use Private Keys and Certificates

### About this task

On a Cloudera Manager Agent, open the `/etc/cloudera-scm-agent/config.ini` configuration file, uncomment and edit the following properties:

Property	Example Value	Description
<code>client_key_file</code>	<code>/opt/cloudera/security/pki/agent.key</code>	Path to the private key file.
<code>client_keypw_file</code>	<code>/etc/cloudera-scm-agent/agentkey.pw</code>	Path to the private key password file.
<code>client_cert_file</code>	<code>/opt/cloudera/security/pki/agent.pem</code>	Path to the client certificate file.

Copy the file to all other cluster hosts. If you have modified properties such as `listening_hostname` or `listening_ip` address in `config.ini`, you must edit the file individually on each host.

## Step 4: Enable Agent Certificate Authentication

### Procedure

1. Log in to the Cloudera Manager Admin Console.
2. Select Administration Settings .
3. Click the Security category.

#### 4. Configure the following TLS settings:

Setting	Description
Use TLS Authentication of Agents to Server	Select this option to enable TLS authentication of agents to the server.
Cloudera Manager TLS/SSL Certificate Trust Store File	Specify the full filesystem path to the jssecacerts file located on the Cloudera Manager Server host. For this example, set the value to:  <pre>&lt;JAVA_HOME&gt;/jre/lib/security/jssecacerts</pre> Replace <JAVA_HOME> with the path to the Oracle JDK.
Cloudera Manager TLS/SSL Certificate Trust Store Password	Specify the password for the jssecacerts truststore.

#### 5. Enter a Reason for Change, then click Save Changes to save the settings.

## Step 5: Restart Cloudera Manager Server and Agents

### Procedure

#### 1. On the Cloudera Manager server host, restart the Cloudera Manager server:

- RHEL 7 compatible:

```
sudo systemctl restart cloudera-scm-server
```

- RHEL 6 compatible, SLES, Ubuntu:

```
sudo service cloudera-scm-server restart
```

#### 2. On every agent host, restart the Cloudera Manager agent:

- RHEL 7 compatible:

```
sudo systemctl restart cloudera-scm-agent
```

- RHEL 6 compatible, SLES, Ubuntu:

```
sudo service cloudera-scm-agent restart
```

## Step 6: Verify that Cloudera Manager Server and Agents are Communicating

### About this task

In the Cloudera Manager Admin Console, go to **Hosts All Hosts** . If you see successful heartbeats reported in the **Last Heartbeat** column after restarting the agents and server, TLS certificate authentication is working properly. If not, check the agent log (/var/log/cloudera-scm-agent/cloudera-scm-agent.log) for errors.

For example, you might see the following error:

```
WrongHost: Peer certificate commonName does not match host, expected 192.0.2.155, got cdh-1.example.com
[02/May/2018 15:04:15 +0000] 4655 MainThread agent ERROR Heartbeat
ing to 192.0.2.155:7182 failed
```

For this scenario, make sure that your DNS and /etc/hosts file are configured correctly, and that your server\_host parameter in /etc/cloudera-scm-agent/config.ini uses the Cloudera Manager Server hostname, and not IP address.

# Manually Configuring TLS Encryption on the Agent Listening Port

The agent listening port (TCP Port 9000) of a Cloudera Manager Agent can be secured with TLS. This port is used for retrieving diagnostic and log information.

## About this task

The requirements for a Cloudera Manager Agent to enable the agent listening port are as follows:

- The following properties must be defined in the config.ini file of the Cloudera Manager Agent: use\_tls=1, verify\_cert\_file, client\_cert\_file, client\_keypw\_file.
- An encryption key must be configured.
- A certificate must be configured.

The main requirement for the Cloudera Manager Server to connect with TLS to the agent listening port is as follows:

## Procedure

- The Cloudera Manager TLS/SSL Client Trust Store File property must be configured to specify the CA certificate using which all the agent certificates are signed.



**Note:** If either the Cloudera Manager Agent or the Cloudera Manager Server is not configured properly, the various diagnostic capture features in Cloudera Manager could fail.

To verify whether the agent listening port is secured with TLS, run the following command:

```
openssl s_client -connect <hostname>:9000
```

If the output of this command includes a server certificate in PEM format, then the port is secured with TLS.