

Key Trustee Server and Key HSM

Date published: 2020-11-30

Date modified: 2024-07-19



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

Key Trustee Server Overview.....	5
Key Trustee Server System Requirements.....	6
Backing up Key Trustee Server and clients.....	7
Back up Key Trustee Server using Cloudera Manager.....	7
Back up Key Trustee Server using the ktbackup.sh script.....	7
Back up Key Trustee Server manually.....	10
Back up Key Trustee Server clients.....	11
Restoring Key Trustee Server.....	12
Restore Key Trustee Server in parcel-based installations.....	12
Restore Key Trustee Server in package-based installations.....	13
Restore Key Trustee Server from ktbackup.sh backups.....	14
Initializing Standalone Key Trustee Server.....	16
Initializing Standalone Key Trustee Server Using Cloudera Manager.....	16
Specifying TLS/SSL Minimum Allowed Version and Ciphers.....	17
Configuring a Mail Transfer Agent for Key Trustee Server.....	17
Verifying Key Trustee Server Operations.....	18
Managing Key Trustee Server Organizations.....	18
Managing Key Trustee Server Certificates.....	21
Generating a New Certificate.....	21
Replacing Key Trustee Server Certificates.....	22
Setting Up Key Trustee Server High Availability.....	24
Configuring Key Trustee Server High Availability Using Cloudera Manager.....	24
Recovering a Key Trustee Server.....	25
Cloudera Navigator Key HSM Overview.....	25
Initializing Navigator Key HSM.....	25
HSM-Specific Setup for Cloudera Navigator Key HSM.....	27

Validating Key HSM Settings.....	30
Verifying Key HSM Connectivity to HSM.....	31
Managing the Navigator Key HSM Service.....	31
Integrating Key HSM with Key Trustee Server.....	33
Integrating custom certificate with Key HSM.....	36
Working with an HSM.....	38
Setting up Luna 7 HSM for KTS and Key HSM.....	38
Configuring encryption algorithms for Luna 7.....	43
Setting up GCP Cloud HSM for KTS and Key HSM.....	44
Setting up CipherTrust HSM for KTS and Key HSM.....	46
Connecting KeySecure HSM to CipherTrust Manager after migration from Key Secure HSM.....	49

Key Trustee Server Overview

An overview of Key Trustee Server and its architecture.

Key Trustee Server Overview

Key Trustee Server is an enterprise-grade virtual safe-deposit box that stores and manages cryptographic keys and other security artifacts. With Key Trustee Server, encryption keys are separated from the encrypted data, ensuring that sensitive data is still protected if unauthorized users gain access to the storage media.

Key Trustee Server protects these keys and other critical security objects from unauthorized access while enabling compliance with strict data security regulations. For added security, Key Trustee Server can integrate with a hardware security module (HSM).

In conjunction with the Ranger KMS, Key Trustee Server can serve as a backing key store for HDFS transparent encryption, providing enhanced security and scalability over the file-based Java KeyStore used by the default Hadoop Key Management Server.

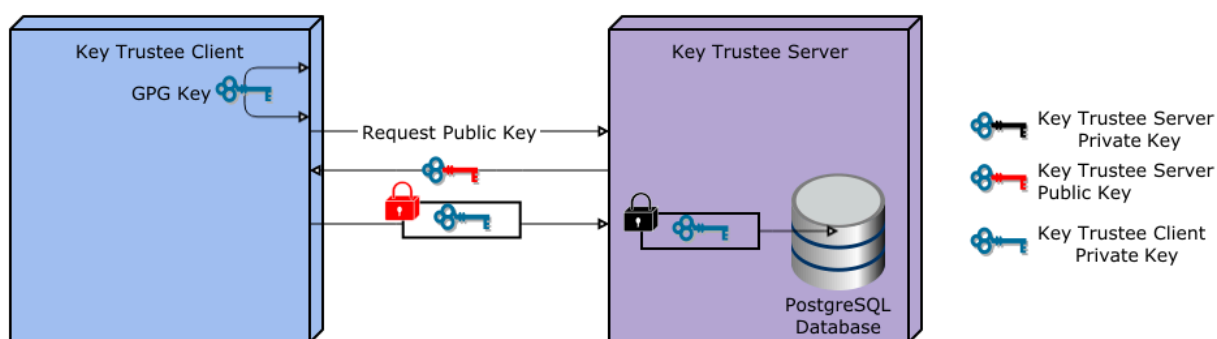
Cloudera Navigator Encrypt also uses Key Trustee Server for key storage and management.

Key Trustee Server Architecture

Key Trustee Server is a secure object store. Clients register with Key Trustee Server, and are then able to store and retrieve objects with Key Trustee Server. The most common use case for Key Trustee Server is storing encryption keys to simplify key management and enable compliance with various data security regulations, but Key Trustee Server is agnostic about the actual objects being stored.

All interactions with Key Trustee Server occur over a TLS-encrypted HTTPS connection.

Key Trustee Server does not generate encryption keys for clients. Clients generate encryption keys, encrypt them with their private key, and send them over a TLS-encrypted connection to the Key Trustee Server. When a client needs to decrypt data, it retrieves the appropriate encryption key from Key Trustee Server and caches it locally to improve performance. This process is demonstrated in the following diagram:



The most common Key Trustee Server clients are Navigator Encrypt and Key Trustee KMS.

When a Key Trustee client registers with Key Trustee Server, it generates a unique fingerprint. All client interactions with the Key Trustee Server are authenticated with this fingerprint. You must ensure that the file containing this fingerprint is secured with appropriate Linux file permissions. The file containing the fingerprint is `/etc/navencrypt/keytrustee/ztrustee.conf` for Navigator Encrypt clients, and `/var/lib/kms-keytrustee/keytrustee/.keytrustee/keytrustee.e.conf` for Key Trustee KMS.

Many clients can use the same Key Trustee Server to manage security objects. For example, you can have several Navigator Encrypt clients using a Key Trustee Server, and also use the same Key Trustee Server as the backing store for Key Trustee KMS (used in HDFS encryption).

Key Trustee Server System Requirements

The system requirements for KTS are described here.

Recommended Hardware and Supported Distributions

Key Trustee Server must be installed on a dedicated server or virtual machine (VM) that is not used for any other purpose. The backing PostgreSQL database must be installed on the same host as the Key Trustee Server, and must not be shared with any other services. For high availability, the active and passive Key Trustee Servers must not share physical resources.

The recommended minimum hardware specifications are as follows:

- Processor: 1 GHz 64-bit quad core
- Memory: 8 GB RAM
- Storage: 20 GB on moderate- to high-performance disk drives

Cloudera Manager Requirements

Installing and managing Key Trustee Server using Cloudera Manager requires Cloudera Manager 7.0.0 and higher. Key Trustee Server does not require Cloudera Navigator Audit Server or Metadata Server.

umask Requirements

Key Trustee Server installation requires the default umask of 0022.

Network Requirements

For new Key Trustee Server installations (5.4.0 and higher) and migrated upgrades, Key Trustee Server requires the following TCP ports to be opened for inbound traffic:

- 11371

Clients connect to this port over HTTPS.

- 11381 (PostgreSQL)

The passive Key Trustee Server connects to this port for database replication.

For upgrades that are not migrated to the CherryPy web server, the pre-upgrade port settings are preserved:

- 80

Clients connect to this port over HTTP to obtain the Key Trustee Server public key.

- 443 (HTTPS)

Clients connect to this port over HTTPS.

- 5432 (PostgreSQL)

The passive Key Trustee Server connects to this port for database replication.

TLS Certificate Requirements

To ensure secure network traffic, Cloudera recommends obtaining Transport Layer Security (TLS) certificates specific to the hostname of your Key Trustee Server. To obtain the certificate, generate a Certificate Signing Request (CSR) for the fully qualified domain name (FQDN) of the Key Trustee Server host. The CSR must be signed by a trusted Certificate Authority (CA). After the certificate has been verified and signed by the CA, the Key Trustee Server TLS configuration requires:

- The CA-signed certificate
- The private key used to generate the original CSR
- The intermediate certificate/chain file (provided by the CA)

Cloudera recommends not using self-signed certificates. If you use self-signed certificates, you must use the `--skip-ssl-check` parameter when registering Navigator Encrypt with the Key Trustee Server. This skips TLS hostname validation, which safeguards against certain network-level attacks.

Backing up Key Trustee Server and clients

In case of failure, you should regularly back up Key Trustee Server databases and configuration files. You must also back up client configuration files and keys for Key Trustee Server clients, such as Navigator Encrypt.

Key Trustee Server high availability applies to read operations only. If either Key Trustee Server fails, the client automatically retries fetching keys from the functioning server. New write operations (for example, creating new encryption keys) are not allowed unless both Key Trustee Servers are operational.

Cloudera strongly recommends regularly backing up Key Trustee Server databases and configuration files. Because these backups contain encryption keys and encrypted deposits, you must ensure that your backup repository is as secure as the Key Trustee Server.

You must also back up client configuration files and keys for Key Trustee Server clients, such as Navigator Encrypt.

Related Information

[Back up Key Trustee Server using Cloudera Manager](#)

[Back up Key Trustee Server using the `ktbackup.sh` script](#)

[Back up Key Trustee Server manually](#)

[Back up Key Trustee Server clients](#)

Back up Key Trustee Server using Cloudera Manager

Cloudera Manager versions 5.8 and higher, when used with Key Trustee Server versions 5.7 and higher, allow for backups of the KT Server.

About this task

The actions executed in this procedure are equivalent to running the `ktbackup.sh` script on the node in question (see “Back up Key Trustee Server using the `ktbackup.sh` script” for additional details).

In addition, when using the HDFS Encryption Wizard in Cloudera Manager 5.8 or higher to install and configure Key Trustee Server versions 5.7 and higher, a cron job is automatically set up to back up the Key Trustee Server on an ongoing basis. See “Initializing Standalone Key Trustee Server” for more detail.

To back up the KT Server service configuration using Cloudera Manager:

Procedure

1. Select the KT Server service configuration that you want to back up.
2. For a KT Server backup, select Create Backup on Active Server (or Create Backup on Passive Server) from the Actions menu.

Results

A successfully completed backup of the KT Server is indicated by the message “Command Create Backup on Active Server finished successfully on service `keytrustee_server`”.

Back up Key Trustee Server using the `ktbackup.sh` script

Key Trustee Server releases 5.7 and higher include a script, `ktbackup.sh`, to simplify and automate backing up Key Trustee Server.

Introduction

When run on a Key Trustee Server host, the script creates a tarball containing the Key Trustee Server private GPG keys and the PostgreSQL database.

To preserve the security of the backup, you must specify a GPG recipient. Because this recipient is the only entity that can decrypt the backup, the recipient must be someone authorized to access the Key Trustee Server database, such as a key administrator.

Creating and Importing a GPG Key for Encrypting and Decrypting Backups

If the key administrator responsible for backing up and restoring Key Trustee Server does not already have a GPG key pair, they can create one using the `gpg --gen-key` command. The following example demonstrates this procedure:



Note: By default, `gpg --gen-key` fails at the password prompt if you have logged in to your user account with the `su` command. You must log in to the SSH session with the user account for which you want to generate the GPG key pair.

```
[john.doe@backup-host ~]$ gpg --gen-key
gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
  (1) RSA and RSA (default)
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
    0 = key does not expire
    <n> = key expires in n days
    <n>w = key expires in n weeks
    <n>m = key expires in n months
    <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.
Real name: John Doe
Email address: john.doe@example.com
Comment: Key Trustee Backup
You selected this USER-ID:
    "John Doe (Key Trustee Backup) <john.doe@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.
can't connect to `/home/john.doe/.gnupg/S.gpg-agent': No such file or directory
gpg-agent[10638]: directory `/home/john.doe/.gnupg/private-keys-v1.d' created
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
```



```

generator a better chance to gain enough entropy.
gpg: /home/john.doe/.gnupg/trustdb.gpg: trustdb created
gpg: key 0936CB67 marked as ultimately trusted
public and secret key created and signed.
gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid: 1  signed: 0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub 2048R/0936CB67 2016-02-10
    Key fingerprint = CE57 FDED 3AFE E67D 2041 9EBF E64B 7D00 0936 CB67
uid                               John Doe (Key Trustee Backup) <john.doe@example.com>
sub 2048R/52A6FC5C 2016-02-10

```

After the GPG key pair is generated, you can export the public key:

```
[john.doe@backup-host ~]$ gpg --armor --output /path/to/johndoe.pub --export 'John Doe'
```

Copy the public key (johndoe.pub in this example) to the Key Trustee Server host or Ranger KMS host, and import it into the service account keyring (keytrustee for Key Trustee Server and kms for Ranger KMS):

- On the Key Trustee Server host:

```
sudo -u keytrustee gpg --import /path/to/johndoe.pub
```

- On the Ranger KMS host:

```
sudo -u kms gpg --import /path/to/johndoe.pub
```

Running the ktbackup.sh Script

You must run ktbackup.sh as the service account. The location of the script depends on the service and installation method. See the following table for the script location and default service account for package- and parcel-based installations for Key Trustee Server.

Table 1: Backup Script Locations

Service	Service Account	Parcel-Based Installation	Package-Based Installation
Key Trustee Server	keytrustee	/opt/cloudera/parcels/KEYTRUSTEE_SERVER/bin/ktbackup.sh	/usr/bin/ktbackup.sh

The following table lists the command options for ktbackup.sh.

Table 2: Command Options for ktbackup.sh

Command Option	Description
-c, --confdir=CONFDIR	Specifies the Key Trustee configuration directory. Defaults to /var/lib/keytrustee/.keytrustee for parcel-based Key Trustee Server. For package-based Key Trustee Server, you must specify this option.
--database-port=PORT	Specifies the Key Trustee Server database port. Defaults to 11381 for parcel-based installations. For package-based Key Trustee Server installations, you must specify this option.
--gpg-recipient=GPG_RECIPIENT	Specifies the GPG recipient. The backup is encrypted with the public key of the specified recipient. The GPG recipient public key must be imported into the service account keyring before running the script.
--cleartext	Outputs an unencrypted tarball. To preserve the security of the cryptographic keys, do not use this option in production environments.

Command Option	Description
<code>--output=DIR</code>	Specifies the output directory for the tarball. Defaults to <code>/var/lib/keytrustee</code> for parcel-based Key Trustee Server. For package-based Key Trustee Server, you must specify this option.
<code>--roll=n</code>	Deletes backups older than the last <i>n</i> backups from the directory specified by the <code>--output</code> parameter. For example, if you have 10 backups, specifying <code>--roll=10</code> creates a new backup (11 backups total) and then delete the oldest backup. Specifying <code>--roll=1</code> creates a new backup and then deletes all other backups.
<code>-q, --quiet</code>	Suppresses console log messages and, if successful, returns only the backup tarball file path. This is useful for automating backups.
<code>--verbose</code>	Outputs additional log messages to the console for debugging.

The following examples demonstrate the command usage for different scenarios:

- To back up a parcel-based Key Trustee Server, specifying the GPG recipient by name:

```
$ sudo -u keytrustee /opt/cloudera/parcels/KEYTRUSTEE_SERVER/bin/ktbackup.sh --gpg-recipient='John Doe'
```

- To back up a package-based Key Trustee Server with the database running on a non-default port (12345 in this example):

```
$ sudo -u keytrustee ktbackup.sh --database-port=12345 --gpg-recipient=john.doe@example.com
```

Automating Backups Using cron

You can schedule automatic backups of Key Trustee Server using the cron scheduling utility.

Create a crontab entry using the following commands:

- Edit the crontab by running the following command:

```
sudo -u keytrustee crontab -e
```

- Add the following entry to run the backup script every 30 minutes. This example is for a parcel-based installation of Key Trustee Server. See the “Backup Script Locations” table for the package-based script location.

```
*/30 * * * * /opt/cloudera/parcels/KEYTRUSTEE_SERVER/bin/ktbackup.sh --gpg-recipient='John Doe' --quiet --output=/tmp/backups --roll=10
```

Run `man 5 crontab` to see the crontab man page for details on using cron to schedule backups at different intervals.

Back up Key Trustee Server manually

If you do not wish to back up KTS using Cloudera Manager or the backup script, you can use this procedure for both parcel-based and package-based installations.

About this task

The following procedure references the default database port and location; if you modified these settings during installation, replace the database and port with your values.

Procedure

1. Back up the Key Trustee Server database:

- For Key Trustee Server 3.8:

```
su - postgres
pg_dump -c -p 5432 keytrustee | zip --encrypt keytrustee-db.zip -
```

- For Key Trustee Server 5.4 and higher:

```
su - keytrustee
pg_dump -c -p 11381 keytrustee | zip --encrypt keytrustee-db.zip -
```

The `--encrypt` option prompts you to create a password used to encrypt the zip file. This password is required to decrypt the file.

For parcel-based installations, you must set environment variables after switching to the `keytrustee` user:

```
su - keytrustee
export PATH=$PATH:/opt/cloudera/parcels/KEYTRUSTEE_SERVER/PG_DB/opt/postgres/<postgres-version>/bin
export LD_LIBRARY_PATH=/opt/cloudera/parcels/KEYTRUSTEE_SERVER/PG_DB/opt/postgres/<postgres-version>/lib
pg_dump -c -p 11381 keytrustee | zip --encrypt keytrustee-db.zip -
```



Note: KTS and corresponding PostgreSQL versions:

- KTS 7.x - 7.1.8: PostgreSQL 12.1
- KTS 7.1.9: PostgreSQL 14.2

2. Back up the Key Trustee Server configuration directory (`/var/lib/keytrustee/.keytrustee`): `zip -r --encrypt keytrustee-conf.zip /var/lib/keytrustee/.keytrustee`

The `--encrypt` option prompts you to create a password used to encrypt the zip file. This password is required to decrypt the file.

3. Move the backup files (`keytrustee-db.zip` and `keytrustee-conf.zip`) to a secure location.

Back up Key Trustee Server clients

Cryptographic keys stored in Key Trustee Server are encrypted by clients before they are sent to Key Trustee Server. The primary client for Key Trustee Server is Navigator Encrypt. Cloudera strongly recommends backing up regularly the configuration files and GPG keys for Key Trustee Server clients.

About this task



Warning: Failure to back up these files can result in irretrievable data loss. For example, encryption zone keys used for HDFS Transparent Encryption are encrypted by the KMS before being stored in Key Trustee Server. A catastrophic failure of the KMS with no backup causes all HDFS data stored in encryption zones to become permanently irretrievable.

Procedure

To prevent permanent data loss, regularly back up the following directories on each client that stores objects in Key Trustee Server:

Table 3: Key Trustee Server Client Configuration Directories

Key Trustee Server Client	Directories to Back Up
Navigator Encrypt	/etc/navencrypt

Restoring Key Trustee Server

If a Key Trustee Server fails catastrophically, you must restore it from backup to a new host with the same hostname and IP address as the failed host.

When restoring the Key Trustee Server database from backup, keep in mind that any keys or deposits created after the backup are not restored.

The procedure to restore Key Trustee Server is different for parcel-based than for package-based installations. For more information about parcels, see “Overview of Parcels”.

Related Information

[Restore Key Trustee Server in parcel-based installations](#)

[Restore Key Trustee Server in package-based installations](#)

[Restore Key Trustee Server from ktbackup.sh backups](#)

Restore Key Trustee Server in parcel-based installations

How to restore a parcel-deployed Key Trustee Server database from a backup.

About this task

These instructions apply to Key Trustee Servers deployed using parcels. For package-based deployments, see “Restore Key Trustee Server in package-based installations” for more information.

The following procedures assume the default database port and location; if you modified these settings during installation, replace the database and port with your custom values.

If the Key Trustee Server host has failed completely, remove the host from the cluster and add a new host using Cloudera Manager:

Procedure

1. Remove the failed host from the cluster. See “Deleting Hosts” for instructions.
2. Add a new host with the same hostname and IP address as the failed host to the cluster. See “Adding a Host to a Cluster” for instructions.



Important: Make sure that the replacement host uses the same operating system version as the failed host.

3. Install Key Trustee Server on the new host. Be sure to install the same Key Trustee Server version as the failed host.
4. After you have provisioned a new host and installed Key Trustee Server (or if you are restoring the database or configuration on the original host), restore the database and configuration directory.

Backups created using...

ktbackup.sh script

See “Restore Key Trustee Server from ktbackup.sh backups”

pg_dump command

Continue below

5. If your backups were created manually using the `pg_dump` command, do the following:
 - a) Copy or move the backup files (*keytrustee-db.zip* and *keytrustee-conf.zip*) to the Key Trustee Server host.
 - b) Start the PostgreSQL server:

```
sudo ktadmin db --start --pg-rootdir /var/lib/keytrustee/db --background
```

- c) Restore the Key Trustee Server database:

```
su - keytrustee
export PATH=$PATH:/opt/cloudera/parcels/KEYTRUSTEE_SERVER/PG_DB/opt/postgres/14.2/bin
export LD_LIBRARY_PATH=/opt/cloudera/parcels/KEYTRUSTEE_SERVER/PG_DB/opt/postgres/14.2/lib
unzip -p /path/to/keytrustee-db.zip | psql -p 11381 -d keytrustee
```

If the zip file is encrypted, you are prompted for the password to decrypt the file.

- d) Restore the Key Trustee Server configuration directory:

```
su - keytrustee
cd /var/lib/keytrustee
unzip /path/to/keytrustee-conf.zip
```

If the zip file is encrypted, you are prompted for the password to decrypt the file.

- e) Stop the PostgreSQL server:

```
sudo ktadmin db --stop --pg-rootdir /var/lib/keytrustee/db
```

- f) Start the Key Trustee Server service in Cloudera Manager (Key Trustee Server service Actions Start).
- g) Remove the backup files (*keytrustee-db.zip* and *keytrustee-conf.zip*) from the Key Trustee Server host.

Related Information

[Restore Key Trustee Server in package-based installations](#)

[Deleting Hosts](#)

[Adding a Host to a Cluster](#)

[Restore Key Trustee Server from ktbackup.sh backups](#)

Restore Key Trustee Server in package-based installations

How to restore a package-deployed Key Trustee Server database from a backup.

About this task

These instructions apply to Key Trustee Servers deployed using a package. For parcel-based deployments, see “Restore Key Trustee Server in parcel-based installations” for more information.

The following procedures assume the default database port and location; if you modified these settings during installation, replace the database and port with your custom values.

If the Key Trustee Server host has failed completely, remove the host from the cluster and add a new host using Cloudera Manager:



Important: Make sure to install the same operating system and Key Trustee Server versions as the failed host.

Procedure

1. After you have provisioned a new host and installed Key Trustee Server (or if you are restoring the database or configuration on the original host), restore the database and configuration directory.

Backups created using...**ktbackup.sh script**

See “Restore Key Trustee Server from ktbackup.sh backups”

pg_dump command

Continue below

2. If your backups were created manually using the `pg_dump` command, do the following:
 - a) Copy or move the backup files (*keytrustee-db.zip* and *keytrustee-conf.zip*) to the Key Trustee Server host.
 - b) Change the file ownership on the backup files to `keytrustee:keytrustee`:

```
sudo chown keytrustee:keytrustee /path/to/keytrustee*.zip
```

- c) Restore the Key Trustee Server database:

```
su - keytrustee
unzip -p /path/to/keytrustee-db.zip | psql -p 11381 -d keytrustee
```

If the zip file is encrypted, you are prompted for the password to decrypt the file.

- d) Restore the Key Trustee Server configuration directory:

```
cd /var/lib/keytrustee
unzip /path/to/keytrustee-conf.zip
```

If the zip file is encrypted, you are prompted for the password to decrypt the file.

- e) Start the Key Trustee Server service:
 - RHEL 6-compatible: `$ sudo service keytrusteed start`
 - RHEL 7-compatible: `$ sudo systemctl start keytrusteed`
- f) Remove the backup files (*keytrustee-db.zip* and *keytrustee-conf.zip*) from the Key Trustee Server host.

Related Information

[Restore Key Trustee Server in parcel-based installations](#)

[Restore Key Trustee Server from ktbackup.sh backups](#)

Restore Key Trustee Server from ktbackup.sh backups

After installing Key Trustee Server on a new host after a failure, or if you need to restore accidentally deleted keys on the same host, use the following procedure to restore Key Trustee Server from backups generated by the `ktbackup.sh` script.

Procedure

1. Decrypt the backup tarball using the private key of the GPG recipient specified in the backup command by running the following command as the GPG recipient user account. The GPG recipient private key must be available on the Key Trustee Server host on which you run this command.

```
gpg -d -o /path/to/decrypted/backup.tar /path/to/encrypted/tarball
```

2. Verify the decrypted tarball using the `tar tvf /path/to/decrypted/backup.tar` command. For example:

```
$ tar tvf kts_bak_kts01_example_com_2016-02-10_11-14-37.tar
drwx----- keytrustee/keytrustee 0 2016-02-09 16:43 var/lib/keytrustee/.
keytrustee/
```

```

-rw----- keytrustee/keytrustee 434 2016-02-09 16:43 var/lib/keytrustee
/.keytrustee/keytrustee.conf
-rw----- keytrustee/keytrustee 1280 2016-02-09 16:43 var/lib/keytrust
ee/.keytrustee/trustdb.gpg
-rw----- keytrustee/keytrustee 4845 2016-02-09 16:43 var/lib/keytrustee
/.keytrustee/secring.gpg
-rw----- keytrustee/keytrustee 600 2016-02-09 16:43 var/lib/keytrust
ee/.keytrustee/random_seed
drwx----- keytrustee/keytrustee 0 2016-02-09 16:40 var/lib/keytrustee
/.keytrustee/.ssl/
-rw----- keytrustee/keytrustee 1708 2016-02-09 16:40 var/lib/keytrustee
/.keytrustee/.ssl/ssl-cert-keytrustee-pk.pem
-rw----- keytrustee/keytrustee 1277 2016-02-09 16:40 var/lib/keytrust
ee/.keytrustee/.ssl/ssl-cert-keytrustee.pem
-rw----- keytrustee/keytrustee 2263 2016-02-09 16:43 var/lib/keytrustee
/.keytrustee/pubring.gpg
-rw-r--r-- keytrustee/keytrustee 457 2016-02-09 16:43 var/lib/keytrustee/
.keytrustee/logging.conf
-rw----- keytrustee/keytrustee 2263 2016-02-09 16:43 var/lib/keytrust
ee/.keytrustee/pubring.gpg~
-rw----- keytrustee/keytrustee 157 2016-02-09 16:40 var/lib/keytrustee
/.keytrustee/gpg.conf
-rw-r--r-- keytrustee/keytrustee 47752 2016-02-10 11:14 var/lib/keytrustee
e/kts_bak_kts01_example_com_2016-02-10_11-14-37.sql

```

3. Restore the files to their original locations, using this command for Key Trustee Server backups:

```
tar xvf /path/to/decrypted/backup.tar -C /
```

4. Drop and re-create the keytrustee PostgreSQL database, and restore the database from the backup.

- For parcel-based installations:

```

$ su - keytrustee
$ source /opt/cloudera/parcels/KEYTRUSTEE_SERVER/meta/keytrustee_env.sh
$ /opt/cloudera/parcels/KEYTRUSTEE_SERVER/PG_DB/opt/postgres/14.2/bin/
psql -p 11381
psql (14.2)
Type "help" for help.
keytrustee=# \list

```

Name	Owner	Encoding	Collate	Ctype	Access privileges
keytrustee	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	=c/keytrustee
template1	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	=c/keytrustee

```

(4 rows)

keytrustee=# \c postgres;
You are now connected to database "postgres" as user "keytrustee".
postgres=# drop database keytrustee;
DROP DATABASE
postgres=# create database keytrustee;
CREATE DATABASE
postgres=# \q

```

```
$ sudo -u keytrustee /opt/cloudera/parcels/KEYTRUSTEE_SERVER/PG
_DB/opt/postgres/14.2/bin/psql -p 11381 -f /var/lib/keytrustee/
kts_bak_kts01_example_com_2016-02-10_11-14-37.sql
```

- For package-based installations:

```
$ su - keytrustee
$ psql -p 11381
psql (14.2)
Type "help" for help.
keytrustee=# \list
```

Name	Owner	Encoding	Collate	Ctype	Access privileges
keytrustee	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	
postgres	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	
template0	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	=c/keytr
template1	keytrustee	UTF8	en_US.UTF-8	en_US.UTF-8	=c/keytr

```
keytrustee=# \c postgres;
You are now connected to database "postgres" as user "keytrustee".
postgres=# drop database keytrustee;
DROP DATABASE
postgres=# create database keytrustee;
CREATE DATABASE
postgres=# \q
$ sudo -u keytrustee psql -p 11381 -f /var/lib/keytrustee/
kts_bak_kts01_example_com_2016-02-10_11-14-37.sql
```

5. Restart Key Trustee Server.

- Using Cloudera Manager: Key Trustee Server service Actions Restart
- Using the Command Line: Run the following command on the Key Trustee Server hosts:

```
sudo service keytrusteed restart      #RHEL 6-compatible
sudo systemctl restart keytrusteed    #RHEL 7-compatible
```

6.

Initializing Standalone Key Trustee Server

How to initialize your non-HA Key Trustee Server.

Initializing Standalone Key Trustee Server Using Cloudera Manager

How to initialize your standalone Key Trustee Server using Cloudera Manager.



Important: If you are using SSH software other than OpenSSH, the initialization fails. To prevent this, pre-create the SSH key before continuing:

```
sudo -u keytrustee ssh-keygen -t rsa -f /var/lib/keytrustee/.ssh/id_rsa
```

For new installations, use the Set up HDFS Data At Rest Encryption wizard and follow the instructions in “Enabling HDFS Encryption Using the Wizard”. When prompted, deselect the Enable High Availability option to proceed in standalone mode.



Important: You must assign the Key Trustee Server and Database roles to the same host. Key Trustee Server does not support running the database on a different host.

For parcel-based Key Trustee Server releases 5.8 and higher, Cloudera Manager automatically backs up Key Trustee Server (using the `ktbackup.sh` script) after adding the Key Trustee Server service. It also schedules automatic backups using cron. For package-based installations, you must manually back up Key Trustee Server and configure a cron job.

Cloudera Manager configures cron to run the backup script hourly. The latest 10 backups are retained in `/var/lib/keytrustee` in cleartext. For information about using the backup script and configuring the cron job (including how to encrypt backups), see “Backing up Key Trustee Server and clients”.

Related Information

[Backing up Key Trustee Server and clients](#)

Specifying TLS/SSL Minimum Allowed Version and Ciphers

Depending on your cluster configuration and the security practices in your organization, you might need to restrict the allowed versions of TLS/SSL used by Key Trustee Server. Older TLS/SSL versions might have vulnerabilities or lack certain features.

Specify one of the following values using the Minimum TLS Support configuration setting:

- `tlsv1`: Allow any TLS version of 1.0 or higher. This setting is the default when TLS/SSL is enabled.
- `tlsv1.1`: Allow any TLS version of 1.1 or higher.
- `tlsv1.2`: Allow any TLS version of 1.2 or higher.



Note: The pyOpenSSL version on the Key Trustee Server cluster should be updated to 16.2 before changing the TLS version to 1.2. If pyOpenSSL is not updated, then the following error appears when the Key Trustee Server service attempts to restart:

```
keytrustee-server: Error in setting the protocol to the value TLSv1.2.
This usually means there is no support for the entered value.
Python Error: 'module' object has no attribute 'OP_NO_TLSv1_1'
```

Along with specifying the version, you can also specify the allowed set of TLS ciphers using the Supported Cipher Configuration for SSL configuration setting. The argument to this option is a list of keywords, separated by colons, commas, or spaces, and optionally including other notation.

```
AES256:CAMELLIA256-SHA
```

By default, the cipher list is empty, and Key Trustee Server uses the default cipher list for the underlying platform. See the output of `man ciphers` for the full set of keywords and notation allowed in the argument string.

Configuring a Mail Transfer Agent for Key Trustee Server

The Key Trustee Server requires a mail transfer agent (MTA) to send email. Cloudera recommends Postfix, but you can use any MTA that meets your needs.

To configure Postfix for local delivery, run the following commands:

```
export KEYTRUSTEE_SERVER_PK="/var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee-pk.pem"
export KEYTRUSTEE_SERVER_CERT="/var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee.pem"
export KEYTRUSTEE_SERVER_CA="/var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee-ca.pem"
export KEYTRUSTEE_SERVER_HOSTNAME="$(hostname -f)" # or adjust as required
postconf -e 'mailbox_command ='
postconf -e 'smtpd_sasl_local_domain ='
postconf -e 'smtpd_sasl_auth_enable = yes'
postconf -e 'smtpd_sasl_security_options = noanonymous'
postconf -e 'broken_sasl_auth_clients = yes'
postconf -e 'smtpd_recipient_restrictions = permit_sasl_authenticated,permit_my networks,reject_unauth_destination'
postconf -e 'inet_interfaces = localhost'
postconf -e 'smtp_tls_security_level = may'
postconf -e 'smtpd_tls_security_level = may'
```

Start the Postfix service and ensure that it starts at boot:

```
service postfix restart
sudo chkconfig --level 235 postfix on
```

For information on installing Postfix or configuring a relay host, see the Postfix documentation.

Related Information

[Postfix Documentation](#)

Verifying Key Trustee Server Operations

To ensure that your Key Trustee Server is working correctly, verify its operations.

Verify that the installation was successful by running the following command on all Key Trustee Servers.

```
curl -k https://keytrustee.example.com:11371/?a=fingerprint
```

Replace `keytrustee.example.com` with the fully qualified domain name (FQDN) of each Key Trustee Server you are validating. This command outputs a fingerprint similar to the following:

```
4096R/4EDC46882386C827E20DEEA2D850ACA33BEDB0D1
```

Managing Key Trustee Server Organizations

Organizations allow you to configure Key Trustee for use in a multi-tenant environment. Using the `keytrustee-orgtool` utility, you can create organizations and administrators for multiple organizations. Organization administrators can then approve or deny the registration of clients, depending on the registration method.

The `keytrustee-orgtool` Utility

`keytrustee-orgtool` is a command-line utility for administering organizations. The `keytrustee-orgtool` command must be run as the root user.

The following table explains the various `keytrustee-orgtool` commands and parameters. Run `keytrustee-orgtool --help` to view this information at the command line.

Table 4: Usage for keytrustee-orgtool

Operation	Usage	Description
Add	<code>keytrustee-orgtool add [-h] -n <i>name</i> -c <i>contacts</i></code>	Adds a new organization and administrators for the organization.
List	<code>keytrustee-orgtool list</code>	Lists current organizations, including the authorization secret, all administrators, the organization creation date, and the organization expiration date.
Disable client	<code>keytrustee-orgtool disable-client [-h] --fingerprint <i>fingerprint</i></code>	Disables a client that has already been activated by the organization administrator.
Enable client	<code>keytrustee-orgtool enable-client [-h] --fingerprint <i>fingerprint</i></code>	Enables a client that has requested activation but has not yet been approved by the organization administrator.
Set authorization Code	<code>keytrustee-orgtool set-auth [-h] -n <i>name</i> -s <i>secret</i></code>	Sets the authorization code to a new string, or to blank to allow automatic approvals without the code.

Create Organizations

Each new Key Trustee tenant needs its own organization. You can create new organizations using the `keytrustee-orgtool add` command. For example, to create a new organization for the Disaster Recovery group and add two administrators, Finn and Jake:

```
keytrustee-orgtool add -n disaster-recov -c finn@example.com,jake@example.com
```

When adding organizations:

- Do not use spaces or special characters in the organization name. Use hyphens or underscores instead.
- Do not use spaces between email addresses (when adding multiple administrators to an organization). Use a comma to separate email addresses, without any space (as shown in the example above).

Each contact email address added when creating the organization receives a notification email, as detailed below.

Once an organization exists, use the `keytrustee-orgtool add` command to add new administrators to the organization. For example, to add an administrator to the disaster-recov organization:

```
keytrustee-orgtool add -n disaster-recov -c marceline@example.com
```



Note: You cannot remove contacts from an organization with the `keytrustee-orgtool` utility.

List Organizations

After creating an organization, verify its existence with the `keytrustee-orgtool list` command. This command lists details for all existing organizations. The following is the entry for the disaster-recov organization created in the example:

```
{
  "disaster-recov": {
    "auth_secret": "/qFiICsyYqMLhdTznNY3Nw==",
    "contacts": [
      "finn@example.com",
      "jake@example.com"
    ],
    "creation": "2013-12-02T09:55:21",
    "expiration": "9999-12-31T15:59:59",
    "key_info": null,
    "name": "disaster-recov",
  }
}
```

```

    "state": 0,
    "uuid": "xY3Z8xCwMuKZMiTYJa0mZOdHmVdxhyCUOc6vSNc9I8X"
  }

```

Change the Authorization Code

When an organization is created, an authorization code is automatically generated. When you run the `keytrustee-orgtool list` command, the code is displayed in the `auth_secret` field. To register with a Key Trustee Server, the client must have the authorization code along with the organization name. To set a new `auth_secret`, run the following command:

```
keytrustee-orgtool set-auth -n disaster-recov -s ThisISAs3cr3t!
```

Run the `keytrustee-orgtool list` command again, and confirm the updated `auth_secret` field:

```

"disaster-recov": {
  "auth_secret": "ThisISAs3cr3t!",
  "contacts": [
    "finn@example.com",
    "jake@example.com"
  ],
  "creation": "2013-12-02T09:55:21",
  "expiration": "9999-12-31T15:59:59",
  "key_info": null,
  "name": "disaster-recov",
  "state": 0,
  "uuid": "xY3Z8xCwMuKZMiTYJa0mZOdHmVdxhyCUOc6vSNc9I8X"
}

```

If you do not want to use an authorization code, set the `auth_secret` field to an empty string:

```
keytrustee-orgtool set-auth -n disaster-recov -s ""
```

Cloudera recommends requiring an authorization code.

Notification Email and GPG Keys

Whenever an administrator is added to an organization, the Key Trustee Server sends an automated email message (subject: “KeyTrustee Contact Registration”) to the newly added administrator:

```

Hello, this is an automated message from your Cloudera keytrustee Server.

Welcome to Cloudera keytrustee! You have been listed as an administrator con
tact
for keytrustee services at your organization [test-org]. As an administrato
r,
you may be contacted to authorize the activation of new keytrustee clients.

We recommend that you register a GPG public key for secure administration of
your clients. To do so, visit the link below and follow the instructions.

https://keytrustee01.example.com:11371/?q=CnRV6u0nbm7zB07BQEpXCXsN0QJFBz684u
C0lcHMoWL

This link will expire in 12 hours, at Thu Sep  3 00:08:25 2015 UTC.

```

Cloudera recommends that an organization's administrators:

- Register the GPG public key by following the link contained in the notification email. Registering the GPG public key is optional, but if you choose to register your public key:
 - Complete the process within 12 hours, before the link expires.
 - Upload the entire key, including the BEGIN and END strings, as shown here:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.2.1 (GNU/Linux)
Comment: For info see http://www.gnupg.org

mQGIBDkHP3URBACKWGsYh43pkXU9wj/XlG67K8/DSr185r7dNtHNfLL/ewill10k2
q8saWJn26QZPsDVqdUJModHfJ6kQTAt9NzQbgcVrxLYNfgeBsvkHF/POtnYcZRgL
tZ6syBBWs8JB4xt5V09iJSGAMPUQE8Jpdn2aRXPapdoDw179LM8Rq6r+gwCg5ZZa
. . .
-----END PGP PUBLIC KEY BLOCK-----
```

- Import the Key Trustee Server's public GPG key to verify that the server is the sender.

The organization's administrators are notified by email when new clients are registered to the Key Trustee Server using the mail transfer agent (as discussed in “Configuring a Mail Transfer Agent for Key Trustee Server”). However, if the server does not have access to email, you can use a local system mail address, such as `username@hostname`, where `hostname` is the system hostname and `username` is a valid system user. If you use a local system email address, be sure to regularly monitor the email box.

Related Information

[Configuring a Mail Transfer Agent for Key Trustee Server](#)

Managing Key Trustee Server Certificates

Transport Layer Security (TLS) certificates are used to secure communication with Key Trustee Server. By default, Key Trustee Server generates self-signed certificates when it is first initialized. Cloudera strongly recommends using certificates signed by a trusted Certificate Authority (CA).

Generating a New Certificate

Generating a new certificate for Key Trustee Server.

Procedure

1. Generate a new certificate signing request (CSR):

```
openssl req -new -key keytrustee_private_key.pem -out new.csr
```

Replace `keytrustee_private_key.pem` with the filename of the private key. You can reuse the existing private key or generate a new private key in accordance with your company policies. For existing auto-generated self-signed certificates, the private key file is located at `/var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee-pk.pem`.

2. Generate a new certificate from the CSR:

- For a CA-signed certificate, submit the CSR to the CA, and they will provide a signed certificate.
- To generate a new self-signed certificate, run the following command:

```
$ openssl x509 -req -days 365 -in new.csr -signke
y keytrustee_private_key.pem \
-out new_keytrustee_certificate.pem
```

Related Information

[Replacing Key Trustee Server Certificates](#)

Replacing Key Trustee Server Certificates

Use the following procedure if you need to replace an existing certificate for the Key Trustee Server. For example, you can use this procedure to replace the auto-generated self-signed certificate with a CA-signed certificate, or to replace an expired certificate.

About this task



Note: Key Trustee Server supports password-protected private keys, but not password-protected certificates.

Procedure

1. After “Generating a New Certificate”, back up the original certificate and key files:

```
sudo cp -r /var/lib/keytrustee/.keytrustee/.ssl /var/lib/keytrustee/.keytrustee/.ssl.bak
```

2. (CA-Signed Certificates Only) Provide either the root or intermediate CA certificate:



Important: If you have separate root CA and intermediate CA certificate files, then you must concatenate them into a single file. If you need to convert from a JKS with combined certificate and private key file to a PEM file and separate private key file, refer to “How to Convert File Encodings (DER, JKS, PEM) for TLS/SSL Certificates and Keys”.

```
sudo mv /path/to/rootca.pem /var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee-ca.pem
```

3. Make sure that the certificate and key files are owned by the keytrustee user and group, with file permissions set to 600:

```
sudo chown keytrustee:keytrustee /path/to/new/certificate.pem /path/to/new/private_key.pem
sudo chmod 600 /path/to/new/certificate.pem /path/to/new/private_key.pem
```

4. Update the Key Trustee Server configuration with the location of the new certificate and key files:

- Using Cloudera Manager:
 - a. Go to the Key Trustee Server service.
 - b. Click the Configuration tab.
 - c. Select Category Security .
 - d. Edit the following properties to specify the location of the new certificate and key files. If the private keys are not password protected, leave the password fields empty.
 - Active Key Trustee Server TLS/SSL Server Private Key File (PEM Format)
 - Active Key Trustee Server TLS/SSL Server Certificate File (PEM Format)
 - Active Key Trustee Server TLS/SSL Private Key Password
 - Passive Key Trustee Server TLS/SSL Server Private Key File (PEM Format)
 - Passive Key Trustee Server TLS/SSL Server Certificate File (PEM Format)
 - Passive Key Trustee Server TLS/SSL Private Key Password
 - e. Enter a Reason for change, and then click Save Changes to commit the changes.
- Using the command line:
 - a. Edit `/var/lib/keytrustee/.keytrustee/keytrustee.conf` on the active and passive Key Trustee Server hosts and modify the `SSL_CERTIFICATE` and `SSL_PRIVATE_KEY` parameters as follows:

```
"SSL_CERTIFICATE": "/path/to/new/certificate.pem",
"SSL_PRIVATE_KEY": "/path/to/new/private_key.pem"
```

If the private key is password protected, add the following entry:

```
"SSL_PRIVATE_KEY_PASSWORD_SCRIPT": "/path/to/password_script
[arguments]"
```

Replace `/path/to/password_script [arguments]` with the path to a script (and any necessary command arguments) that returns the password for the private key file. If you do not want to create a script, you can use a simple command, such as `echo -n password`. For example:

```
"SSL_PRIVATE_KEY_PASSWORD_SCRIPT": "/bin/echo -n password"
```

Keep in mind that this method can expose the private key password in plain text to anyone who can view the `/var/lib/keytrustee/.keytrustee/keytrustee.conf` file.

5. Restart Key Trustee Server:

- Using Cloudera Manager: Restart the Key Trustee Server service (Key Trustee Server service Actions Restart).
- Using the Command Line: Restart the Key Trustee Server daemon:
 - RHEL 6-compatible: `$ sudo service keytrusteed restart`
 - RHEL 7-compatible: `$ sudo systemctl restart keytrusteed`

6. If you are using Key HSM, update the Key Trustee Server and Key HSM configuration:

- a) Run the `keyhsm trust` command, using the path to the new certificate:

```
sudo keyhsm trust /path/to/new/key_trustee_server/cert
```

- b) Run the `ktadmin keyhsm` command, using the `--client-certfile` and `--client-keyfile` options to specify the location of the new certificate file and private key:

```
sudo ktadmin keyhsm --server https://keyhsm01.example.com:9090 --client-
certfile /path/to/new/key_trustee_server/cert --client-keyfile /path/to/
new/key_trustee_server/private_key
```

Related Information

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

Setting Up Key Trustee Server High Availability

Key Trustee Server high availability applies to read operations only. If either Key Trustee Server fails, the KeyProvider automatically retries fetching keys from the functioning server. New write operations (for example, creating new encryption keys) are not allowed unless both Key Trustee Servers are operational.

If a Key Trustee Server fails, the following operations are impacted:

- HDFS Encryption
 - You cannot create new encryption keys for encryption zones.
 - You can write to and read from existing encryption zones, but you cannot create new zones.
- Cloudera Navigator Encrypt
 - You cannot register new Cloudera Navigator Encrypt clients.
 - You can continue reading and writing encrypted data, including creating new mount points, using existing clients.

Cloudera recommends monitoring both Key Trustee Servers. If a Key Trustee Server fails catastrophically, restore it from backup to a new host with the same hostname and IP address as the failed host. Cloudera does not support PostgreSQL promotion to convert a passive Key Trustee Server to an active Key Trustee Server.

Depending on your cluster configuration and the security practices in your organization, you might need to restrict the allowed versions of TLS/SSL used by Key Trustee Server.

Configuring Key Trustee Server High Availability Using Cloudera Manager

About this task

Procedure

1. Pick one:
 - a) For new installations, use the Set up HDFS Data At Rest Encryption wizard and follow the instructions in “Enabling HDFS Encryption Using the Wizard”. When prompted, make sure that the Enable High Availability option is selected.
 - b) If you already have a Key Trustee Server service, and want to enable high availability, see “Adding a Role Instance” for the Key Trustee Server service instead to add the Passive Key Trustee Server and Passive Database roles.



Note: You must assign the Key Trustee Server and Database roles to the same host. Assign the Active Key Trustee Server and Active Database roles to one host, and the Passive Key Trustee Server and Passive Database roles to a separate host.

After completing the Add Role Instances wizard, the Passive Key Trustee Server and Passive Database roles fail to start. Complete the following manual actions to start these roles:

2. Stop the Key Trustee Server service (Key Trustee Server service Actions Stop).
3. Run the Set Up Key Trustee Server Database command (Key Trustee Server service Actions Set Up Key Trustee Server Database).

4. Run the following command on the Active Key Trustee Server:

```
sudo rsync -zcav --exclude .ssl /var/lib/keytrustee/.keytrustee  
root@keytrustee02.example.com:/var/lib/keytrustee/.
```

Replace *keytrustee02.example.com* with the hostname of the Passive Key Trustee Server.

5. Run the following command on the Passive Key Trustee Server:

```
sudo ktadmin init
```

6. Start the Key Trustee Server service (Key Trustee Server service Actions Start).



Important: Starting or restarting the Key Trustee Server service attempts to start the Active Database and Passive Database roles. If the Active Database is not running when the Passive Database attempts to start, the Passive Database fails to start. If this occurs, manually restart the Passive Database role after confirming that the Active Database role is running.

7. Enable synchronous replication (Key Trustee Server service Actions Setup Enable Synchronous Replication in HA mode).
8. Restart the Key Trustee Server service (Key Trustee Server service Actions Restart).

Recovering a Key Trustee Server

If a Key Trustee Server fails, restore it from backup as soon as possible. If the Key Trustee Server hosts fails completely, make sure that you restore the Key Trustee Server to a new host with the same hostname and IP address as the failed host.

For more information, see “Backing up Key Trustee Server and clients” and “Restoring Navigator Key Trustee Server”.

Related Information

[Backing up Key Trustee Server and clients](#)

[Restoring Key Trustee Server](#)

Cloudera Navigator Key HSM Overview

Cloudera Navigator Key HSM allows Cloudera Navigator Key Trustee Server to seamlessly integrate with a hardware security module (HSM). Key HSM enables Key Trustee Server to use an HSM as a root of trust for cryptographic keys, taking advantage of Key Trustee Server’s policy-based key and security asset management capabilities while at the same time satisfying existing, internal security requirements regarding treatment of cryptographic materials.

For instructions on installing Key HSM, see [Installing Cloudera Navigator Key HSM](#).

Initializing Navigator Key HSM

Key HSM is initialized using a series of CLI commands and prompts. The setup information you enter is dependent on which type of HSM you are using with Navigator Key HSM.

Before you begin

Before initializing Navigator Key HSM, verify that the HSM is properly configured and accessible from the Key HSM host, and that the HSM client libraries are installed on the Key HSM host:

- SafeNet Luna

Install the SafeNet Luna client. No additional configuration is needed.

- SafeNet KeySecure

Extract the KeySecure client tarball in the Key HSM library directory (/usr/share/keytrustee-server-keyhsm/).

- Thales

Install the Thales client service. Copy nCipherKM.jar, jcetools.jar, and rsaprivenc.jar from the installation media (usually located in opt/nfast/java/classes relative to the installation media mount point) to the Key HSM library directory (/usr/share/keytrustee-server-keyhsm/).

- AWS CloudHSM

Install the AWS CloudHSM client. No additional configuration is needed.

See your HSM product documentation for more information on installing and configuring your HSM and client libraries.



Note: These steps need to be performed for Ranger KMS KTS with KeyHSM with runtime JDK 17

1. Open the keyhsm startup script.

```
vim /usr/share/keytrustee-server-keyhsm/start.sh
```

2. Export java home.

```
export JAVA_HOME=/usr/lib/jvm/jdk1.17.0-openjdk
```

3. Export JAVA options mentioned below.

```
export _JAVA_OPTIONS="--add-opens=java.base/sun.security.rsa=ALL-UNNAMED --add-opens=java.base/com.sun.crypto.provider=ALL-UNNAMED"
```

4. Change the java command at the end accordingly

```
${JAVA_HOME}/bin/java
```

This is an example of a start.sh script.

```
KEYHSM_HOME_DIR=/usr/share/keytrustee-server-keyhsm

export JAVA_HOME=/usr/lib/jvm/jdk1.17.0-openjdk

export _JAVA_OPTIONS="--add-opens=java.base/sun.security.rsa=ALL-UNNAMED --add-opens=java.base/com.sun.crypto.provider=ALL-UNNAMED"

PROPERTIES=${KEYHSM_HOME_DIR}/application.properties

maxHeap=`grep "^keyhsm.jvm.heap.mx.gb" ${PROPERTIES} | awk -F= '{print $NF}' | tr -d '[:space:]'`
if [ -z "$maxHeap" ]; then
    maxHeap=2
fi

${JAVA_HOME}/bin/java -classpath "*/usr/safenet/lunaclient/jsp/lib/*:/opt/nfast/java/classes/*:/opt/cloudhsm/java/*:/usr/share/keytrustee-server-keyhsm/conf/" -Djava.library.path=/usr/safenet/lunaclient/jsp/lib/*:/opt/cloudhsm/lib/ -Dlogback.configurationFile=/usr/share/keytrustee-server-keyhsm/conf/logback.xml -Djdk.tls.trustNameService=true -Xms1g -Xmx${maxHeap}g com.cloudera.app.run.Program $@
```



Note: When using an HSM with Key Trustee Server and Navigator Encrypt, encrypting many block devices may exceed the capacity of the HSM. A key is created in the HSM for each encrypted block device, so be sure that your HSM can support your encryption requirements.

For Luna v7, the keyhsm user must be added to the hsmusers group with the following command:

```
sudo usermod -a -G hsmusers keyhsm
```

Procedure

1. To initialize Key HSM, use the service keyhsm setup command in conjunction with the name of the target HSM distribution:

```
sudo service keyhsm setup [keysecure|thales|luna|cloudhsm]
```

For all HSM distributions, you are prompted for the IP address and port number that Key HSM listens on.



Important: If you have implemented Key Trustee Server high availability, initialize Key HSM on each Key Trustee Server.

Cloudera recommends using the loopback address (127.0.0.1) for the listener IP address and 9090 as the port number:

```
-- Configuring keyHsm General Setup --
Cloudera Recommends to use 127.0.0.1 as the listener port for Key HSM
Please enter Key HSM SSL listener IP address: [127.0.0.1]127.0.0.1
Will attempt to setup listener on 127.0.0.1
Please enter Key HSM SSL listener PORT number: 9090
validate Port:                               :[ Successful ]
```

2. If the setup utility successfully validates the listener IP address and port, you are prompted for additional information specific to your HSM. For HSM-specific instructions, continue to the [HSM-Specific Setup for Cloudera Navigator Key HSM](#) on page 27 section for your HSM.
3. The Key HSM keystore defaults to a strong, randomly-generated password. However, you can change the keystore password in the application.properties file:

```
keyhsm.keystore.password.set=yes
```

Next, run the service keyhsm setup command with the name of the HSM to which the keystore password applies. You will be prompted to enter the new keystore password, which must be a minimum of six characters in length:

```
sudo service keyhsm setup [keysecure|thales|luna|cloudhsm]
```

4. After initial setup, the configuration is stored in the /usr/share/keytrustee-server-keyhsm/application.properties file, which contains human-readable configuration information for the Navigator Key HSM server.



Important: The truststore file created during Key HSM initialization must be stored at /usr/share/keytrustee-server-keyhsm/. There is no way to change the default location.

For additional details about keystores and truststores, see [Understanding Keystores and Truststores](#).

HSM-Specific Setup for Cloudera Navigator Key HSM

Additional HSM-specific setup information for Key HSM.

SafeNet KeySecure



Note: KeySecure was previously named DataSecure, but the Key HSM configuration process is the same for both.

Prerequisites

Before setting up SafeNet KeySecure, be sure to:

- Set the protocol to NAE-XML
- Set Allow Key and Policy Configuration Operations to enabled
- Set Password Authentication to required
- Disable Client Certificate Authentication
- Set KeySecure Crypto Operations to activated

For additional details about SafeNet KeySecure settings, see the SafeNet KeySecure product documentation.

After entering the Key HSM listener IP address and port, the HSM setup for SafeNet KeySecure prompts for login credentials, the IP address of the KeySecure HSM, and the port number:

```
-- Ingrian HSM Credential Configuration --
Please enter HSM login USERNAME: keyhsm
Please enter HSM login PASSWORD: *****

Please enter HSM IP Address or Hostname: 172.19.1.135
Please enter HSM Port number: 9020
```

If the connection is successful, the following message is displayed:

```
Valid address:                               :[ Successful ]
```

The KeySecure setup utility then prompts you whether to use SSL:

```
Use SSL? [Y/n] Y
```

If you choose to use SSL, Key HSM attempts to resolve the server certificate, and prompts you to trust the certificate:

```
[0]          Version: 3
          SerialNumber: 0
          IssuerDN: C=US,ST=TX,L=Austin,O=ACME,OU=Dev,
CN=172.19.1.135,E=webadmin@example.com
          Start Date: Thu Jan 29 09:55:57 EST 2015
          Final Date: Sat Jan 30 09:55:57 EST 2016
          SubjectDN: C=US,ST=TX,L=Austin,O=ACME,OU=Dev,
CN=172.19.1.135,E=webadmin@example.com
          Public Key: RSA Public Key
          modulus: abe4a8dcef92e145984309bd466b33b35562c7f875
                  1dlc406b1140e0584890272090424eb347647ba04b
                  34757cacc79652791427d0d8580a652c106bd26945
                  384b30b8107f8e15d2deba8a4e868bf17bb0207383
                  7cffe0ef16d5b5da5cfb4d3625c0affbda6320daf
                  7c6b6d8adfcfb563960fcd1207c059300feb6513408
                  79dd2d929a5b986517531be93f113c8db780c92ddf
                  30f5c8bf2b0bea60359b67be306c520358cc0c3fc3
                  65500d8abeeac99e53cc2b369b2031174e72e6fca1
                  f9a4639e09240ed6d4a73073885868e814839b09d5
                  6aa98a5ale230b46cdb4818321f546ac15567c5968
                  33be47ef156a73e537fd09605482790714f4a276e5
                  f126f935
          public exponent: 10001

          Signature Algorithm: SHA256WithRSAEncryption
```

```

Signature: 235168c68567b27a30b14ab443388039ff12357f
          99ba439c6214e4529120d6ccb4a9b95ab25f81b4
          7deb9354608df45525184e75e80eb0948eae3e15
          c25c1d58c4f86cb9616dc5c68dfe35f718a0b6b5
          56f520317eb5b96b30cd9d027a0e42f60de6dd24
          5598dlfcea262b405266f484143a74274922884e
          362192c4f6417643da2df6dd1a538d6d5921e78e
          20a14e29calbb82b57c02000fa4907bd9f3c890a
          bdae380c0b4dc68710deeaef41576c0f767879a7
          90f30a4b64a6afb3alace0f3ced17ael42ee6f18
          5eff64e8b710606b28563dd99e8367a0d3cbab33
          2e59c03cadce3a5f4e0aaa9d9165e96d062018f3
          6a7e8e3075c40a95d61ebc8db43d77e7

Extensions:
          critical(false) BasicConstraints: isCa(true)
          critical(false) NetscapeCertType: 0xc0
Trust this server? [y/N] Y
Trusted server:                               :[ Successful ]

```

CloudHSM

Before completing the CloudHSM setup, verify that the CloudHSM connection is properly configured by running either the `cloudhsm_mgmt_util` or `key_mgmt_util` tool and verifying that the tool connects to the HSM successfully. After validating this connection, you can exit the tool:

```

$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.
cfg
Connecting to the server(s), it may take time
depending on the server(s) load, please wait...

Connecting to server '<hsm ip>': hostname '<hsm ip>', port 2225...
Connected to server '<hsm ip>': hostname '<hsm ip>', port 2225.
aws-cloudhsm>quit

```

After entering the Key HSM listener IP address and port identifier, the HSM setup for CloudHSM prompts you for: the crypto user name and crypto user password.



Note: You can configure different user types in the CloudHSM client (for example, the crypto user type manages key operations). Be sure to enter the user type crypto (CU) here.

```

-- Configuring CloudHSM --
Please enter the crypto user name: <username>
Please enter the crypto user password (input suppressed):
Configuration saved in 'application.properties' file
Configuration stored in: 'application.properties'. (Note: You can also use
keyhsm settings to quickly view your current configuration)

```

Thales HSM

By default, the Thales HSM client process listens on ports 9000 and 9001. The Cloudera Manager agent also listens on port 9000. To prevent a port conflict, you must change the Thales client ports. Cloudera recommends using ports 11500 and 11501.

To change the Thales client ports, run the following commands:

```

sudo /opt/nfast/bin/config-serverstartup --enable-tcp --enable-privileged-tc
p --port=11500 --privport=11501
sudo /opt/nfast/sbin/init.d-ncipher restart

```

To configure Key HSM to use the modified port, edit the `/usr/share/keytrustee-server-keyhsm/start.sh` file and add the `-DNFAST_SERVER_PORT=11500` Java system property. For example:

```
java -classpath "*/usr/safenet/lunaclient/jsp/lib/*:/opt/nfast/java/classes/*" -Djava.library.path=/usr/safenet/lunaclient/jsp/lib/ -DNFAST_SERVER_PORT=11500 com.cloudera.app.run.Program $@
```

Before completing the Thales HSM setup, run the `nfkminfo` command to verify that the Thales HSM is properly configured:

```
$ sudo /opt/nfast/bin/nfkminfo
World generation 2
state          0x17270000 Initialised Usable Recovery !PINRecovery !Existing
Client
                RTC  NVRAM FTO !AlwaysUseStrongPrimes SEEDebug
```

If state reports `!Usable` instead of `Usable`, configure the Thales HSM before continuing. See the Thales product documentation for instructions.

After entering the Key HSM listener IP address and port, the HSM setup for Thales prompts for the OCS card password:

```
Please enter the OCS Card Password (input suppressed):

Configuration saved in 'application.properties' file
Configuration stored in: 'application.properties'. (Note: You can also use
service keyHsm settings to quickly view your current configuration)
```

Luna HSM



Important: If you have implemented Key Trustee Server high availability, ensure that the Luna client on each Key Trustee Server is configured with access to the same partition. See the Luna product documentation for instructions on configuring the Luna client.

Before completing the Luna HSM setup, run the `vtl verify` command (usually located at `/usr/safenet/lunaclient/bin/vtl`) to verify that the Luna HSM is properly configured.

After entering the Key HSM listener IP address and port, the HSM setup for Luna prompts for the slot number and password:

```
-- Configuring SafeNet Luna HSM --
Please enter SafeNetHSM Slot Number: 1
Please enter SafeNet HSM password (input suppressed):
Configuration stored in: 'application.properties'. (Note: You can also use s
ervice keyHsm settings to quickly view your current configuration)
Configuration saved in 'application.properties' file
```

See the Luna product documentation for instructions on configuring your Luna HSM if you do not know what values to enter here.

Validating Key HSM Settings

After you finish setting up Navigator Key HSM, you can check the configuration settings and verify that Key HSM is properly connected to your HSM.

About this task

After the setup completes, the Key HSM configuration is stored in `/usr/share/keytrustee-server-keyhsm/application.properties`.

You can view these settings using the `service keyhsm settings` command:

```
$ sudo service keyhsm settings

# keyHsm Server Configuration information:
keyhsm.management.address : 172.19.1.2
keyhsm.server.port : 9090
keyhsm.management.port : 9899
keyhsm.service.port : 19791
keyhsm.hardware : ncipher
# Module OCS Password
thales.ocs_password :
  GIqhXDuZsjlOetl37Lb+f+tgkYvKYDm/8StefpNqZWw1B+LfSY1B4eHd
  endtYJio8qLjjbT+e7j2th5xf8O9t8FwfVguuyFW+6wdD
  uNGvse1LY/itCwqF0ScM1B1Mnz4010xqC6y1PW71+0JjjkkqqM5gJJb18lsQFFaIGVM/pY=
```

These settings can be manually configured by modifying the `application.properties` file, with the exception of any passwords. These are encrypted by design, and can only be changed by re-running the setup utility.

Verifying Key HSM Connectivity to HSM

Procedure

1. To verify Hardware Security Module (HSM) operations using Key HSM, run the following command on the Key Trustee Server host (which should also be the Key HSM host as described in [Installing Cloudera Navigator Key HSM](#)):

```
curl -k https://keytrustee01.example.com:11371/test_hsm
```

If Key HSM operations to the HSM are successful, the command returns output similar to the following:

```
"Sample Key TEST_HELLO_DEPOSIT2016-06-03-072718 has been created"
```

You must run this command from the Key Trustee Server host. If you run it from a different host, the command returns an HTTP 403 error code.

2. If the command returns an HTTP 405 error code, restart Key Trustee Server and try again.



Note: If you are using the `test_hsm` script to verify that the Key Hardware Security Module (Key HSM) has successfully integrated with the Key Trustee Server, or to verify that the Key HSM is connected to HSM, and the Key Trustee Server private key file is password-protected, then the verification may fail. This can occur even if the integration is successful or connected.

If this occurs, create a key through Hadoop for the test.

Managing the Navigator Key HSM Service

Key HSM includes a command line tool for managing basic server operations. You can also manage logging, audits, and keys.

Using the keyhsm service

You can use the keyhsm service for basic server operations:

```
$ sudo service keyhsm
keyHsm service usage:
  setup <hsm name> - set up a new connection to an HSM
  trust <path>      - add a trusted client certificate
  validate          - validate that Key HSM is properly configured
  settings          - display the current server configuration
  start             - start the Key HSM proxy server
  status            - show the current Key HSM server status
  stop|shutdown     - force Key HSM server to shut down
  reload            - reload the server (without shutdown)
```

The reload command causes the application to restart internal services without ending the process itself. If you want to stop and start the process, use the restart command.

Logging and Audits

The Navigator Key HSM logs contain all log and audit information, which by default are stored in the `/var/log/keyhsm` directory.

You can configure the maximum log size (in bytes) and maximum number of log files to retain by adding or editing the following entries in the `/usr/share/keytrustee-server-keyhsm/conf/logback.xml` file.

```
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender"
">
  <file>/var/log/keyhsm/keyhsm.log</file>

  <encoder>
    <pattern>%date %level %logger: %msg%n</pattern>
  </encoder>

  <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRolli
ngPolicy">
    <fileNamePattern>/var/log/keyhsm/keyhsm.log.%i</fileNamePattern>
    <minIndex>1</minIndex>
    <maxIndex>10</maxIndex>
  </rollingPolicy>
  <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTrigg
eringPolicy">
    <maxFileSize>10MB</maxFileSize>
  </triggeringPolicy>
</appender>

<root level="info">
  <appender-ref ref="FILE" />
</root>
```

The default log level is info. The filename is `/var/log/keyhsm/keyhsm.log`, the max file size is 10MB, and the last 10 rolled log files will be retained.

**Warning:**

Modifying the settings for items such as the log file size, log level, and number of rolled logs should not cause any issues. However, if more extensive changes are made to the logback.xml file (for example, changing the policy classes or log message format) and these changes introduce malformed XML or incorrect logback settings, then the Key HSM service may return an error and fail to start. Check the validity of the logback.xml file by running the command `keyhsm` after any updates. If there are errors in the formatting of logback.xml, they will appear in the command line:

```
$ keyhsm
...
06:55:20,208 |-ERROR in ch.qos.logback.core.joran.spi.Interpreter@11:85
- no applicable action for [rollingPolicy], current ElementPath is
[[configuration][appender][rollingPolicy]]
06:55:20,208 |-ERROR in ch.qos.logback.core.joran.spi.Interpreter@12
:30 - no applicable action for [fileNamePattern], current ElementPath
is [[configuration][appender][rollingPolicy][fileNamePattern]]
```

Address any errors before restarting Key HSM to pick up the logging changes. A copy of the default logback.xml file is provided at `logback.xml.bkup`. If there is an error in the updates, you can use this file to recover the logging configuration.

Key Naming Convention

To ensure you can manage keys (for example, delete a key), you must understand the naming convention for keys. Keys adhere to the following naming convention: `handle name-uuid-date`, which means if you know the key name and date, you can make modifications to it.

The following example shows the key nomenclature in the context of a key list query on Luna HSM:

```
[root@user 64]# cmu list
Please enter password for token in slot 1 : *****
handle=220
label=key1-3T17-YYdn-2015-07-23
handle=806
label=key2-CMYZ-8Sym-2015-07-23
handle=108
label=key3-qo62-XQfx-2015-07-23
handle=908
label=key2-CMYZ-8Sym-2015-07-23--cert0
handle=614
label=key3-qo62-RWz0-2015-07-23--cert0
handle=825
label=key1-3T11-YYdz-2015-07-23--cert0
```

Integrating Key HSM with Key Trustee Server

Using a hardware security module with Navigator Key Trustee Server requires Key HSM. This service functions as a driver to support interactions between Navigator Key Trustee Server and the hardware security module, and it must be installed on the same host system as Key Trustee Server.

Before you begin

- Prepare Existing Keys for Migration

In this procedure, you are prompted to migrate any existing keys from the Key Trustee Server to the HSM. Successful migration depends on the existing keys conforming to the following constraints:



Warning: Migration fails if any existing keys do not adhere to these constraints.

- Key names can begin with alpha-numeric characters only
- Key names can include only these special characters:
 - Hyphen -
 - Period .
 - Underscore _

To prepare for migration, check your key names and do the following if any of them are non-conforming:

- Decrypt any data using the non-conforming key.
- Create a new key, named as described above.
- Re-encrypt the data using the new key.



Important: Keys are not available during migration, so you should perform these tasks during a maintenance window.

- Both Key HSM and Key Trustee Server must be set up and running. See [Installing Cloudera Navigator Key HSM](#) for details.

Procedure

1. Establish Trust from Key HSM to Key Trustee Server.

This step assumes that Key Trustee Server has a certificate for TLS (wire) encryption as detailed in [Managing Key Trustee Server Certificates](#). Key HSM service must explicitly trust the Key Trustee Server certificate (presented during TLS handshake). To establish this trust, run the following command:

```
sudo keyhsm trust /path/to/key_trustee_server/cert
```

The `/path/to/key_trustee_server/cert` in this command (and in the commands below) depends on whether the Key Trustee Server uses the default certificate (created by default during install), or uses a custom certificate (obtained from a commercial or internal CA). The two alternate paths are shown in the table below. The custom path is a common example but may differ from that shown.

Default	Custom
<code>/var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee.pem</code>	<code>/etc/pki/cloudera/certs/cert-file.crt</code>
<code>/var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee-pk.pem</code>	<code>/etc/pki/cloudera/private/private-key.key</code>



Note: The system requires TLS and Kerberos authentication throughout the system for security reasons. Connections attempted over SSL (1 through 3) and connections from untrusted clients are immediately terminated to prevent [POODLE](#) (Padding Oracle On Downgraded Legacy Encryption) exploits. See the [Cloudera Security Bulletin](#) for more information.

2. Integrate Key HSM and Key Trustee Server.

The following steps assume that both Key HSM and the Key Trustee Server are on the same host system, as detailed in [Installing Cloudera Navigator Key HSM](#). These steps invoke commands on the Key HSM service and the Key Trustee Server, and they must be run on the host—they cannot be run remotely from another host.

a. Ensure the Key HSM service is running:

```
sudo service keyhsm start
```

b. Establish trust from Key Trustee Server to Key HSM specifying the path to the private key and certificate (Key Trustee Server is a client to Key HSM). This example shows how to use the `--client-certfile` and `--client-keyfile` options to specify the path to non-default certificate and key:

```
$ sudo ktadmin keyhsm --server https://keyhsm01.example.com:9090 \
--client-certfile /etc/pki/cloudera/certs/mycert.crt \
--client-keyfile /etc/pki/cloudera/certs/mykey.key --trust
```

For a password-protected Key Trustee Server private key, add the `--passphrase` argument to the command and enter the password when prompted:

```
$ sudo ktadmin keyhsm --passphrase \
--server https://keyhsm01.example.com:9090 \
--client-certfile /etc/pki/cloudera/certs/mycert.crt \
--client-keyfile /etc/pki/cloudera/certs/mykey.key --trust
```



Note: The preceding commands also create a certificate file for the Key HSM that is used by the Key Trustee Server. This certificate file is stored in `/var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keyhsm.pem`.

c. Any keys that exist on the Key Trustee Server are automatically migrated when you run the `ktadmin keyhsm` command. To complete the migration, enter `y` or `yes` at the command prompt:

```
Some deposits were found that will need to be moved to the HSM.
    Note that although this operation can be interrupted, once complete,
    items stored in the HSM must remain there!

Do you want to perform this migration now? [y/N]: y
Migrating hsm deposits...

Migration Complete!
```

d. Restart the Key Trustee Server:

- Using Cloudera Manager: Restart the Key Trustee Server service (Key Trustee Server service Actions Restart).
- Using the Command Line: Restart the Key Trustee Server daemon:
 - RHEL 6-compatible: `$ sudo service keytrusteed restart`
 - RHEL 7-compatible: `$ sudo systemctl restart keytrusteed`

- e. Verify connectivity between the Key HSM service and the HSM:

```
curl -k https://keytrustee01.example.com:11371/test_hsm
```



Important: You must perform the connection verification between Key HSM and the HSM for all Key Trustee Server hosts.

Successful connection and test of operations returns output like the following:

```
"Sample Key TEST_HELLO_DEPOSIT2016-06-03-072718 has been created"
```



Note: If you are using the test_hsm script to verify that the Key Hardware Security Module (Key HSM) has successfully integrated with the Key Trustee Server, or to verify that the Key HSM is connected to HSM, and the Key Trustee Server private key file is password-protected, then the verification may fail. This can occur even if the integration is successful or connected.

If this occurs, then create a key through Hadoop for the test.

See Verifying Key HSM Connectivity to HSM for more information about the validation process.



Caution: There is no path for migrating keys from Key HSM back to the Key Trustee Server. Once Key Trustee Server is integrated with Key HSM, you cannot get the keys from HSM and remove Key HSM. This is because the underlying key metadata is changed on the HSM.

You can achieve this only by performing the following steps, which help you in removing all the keys and setting up the KMS from scratch:

1. Backup the data present in the encryption zone:
 - a. Create a new non-encrypted directory in HDFS.
 - b. Using `distcp`, copy the data from the encryption zone to the new HDFS directory.
2. Delete the keys created previously.
3. Disable encryption:
 - a. Stop Key HSM.
 - b. Stop the Key Trustee Server service.
 - c. Take backup of the Key Trustee Server.
 - d. Delete the Key Trustee Server service in Cloudera Manager (to completely erase its DB).
 1. Delete contents from `/var/lib/keytrustee/db` and `/var/lib/keytrustee/.keytrustee` directories.
 2. Keep the directory structures.
 - e. Delete KMS service in Cloudera Manager.
 1. Delete contents from `/var/lib/kms-keytrustee/keytrustee/.keytrustee` directory.
 2. Keep the directory structure.
4. Reinstall Key Trustee Server.
5. Reinstall KMS.
6. Create new keys and encryption zone.
7. Restore the content of encryption.

Integrating custom certificate with Key HSM

Learn how to integrate the custom certificate with Key HSM. This task mainly includes stopping the keyhsm service, backing up and deleting the existing certificate, add an application property, running the keyhsm setup, copying back the custom certificate, and start the keyhsm service

Procedure

1. Stop the keyhsm service.

```
service keyhsm stop
```

2. Backup and delete the existing certificate which is the keystore file.

```
cp keystore /root/backup
rm -rf keystore
```

3. Open the application.properties file and add the *keyhsm.keystore.password.set= yes* property.

```
vim application.properties
And property keyhsm.keystore.password.set= yes
```

4. Run the keyhsm setup again.

```
[root@dskmsktscu-4 keytrustee-server-keyhsm]# keyhsm setup luna
-- Configuring keyHsm General Setup --
Cloudera Recommends to use 127.0.0.1 as the listener port for Key HSM
Please enter Key HSM SSL listener IP address: [127.0.0.1]<fqdn>
e.g. Please enter Key HSM SSL listener IP address: [127.0.0.1]dskmsktscu-
4.dskmsktscu.root.hwx.site
Will attempt to setup listener on dskmsktscu-4.dskmsktscu.root.hwx.site

Please enter Key HSM SSL listener PORT number: 9090
validate Port:                               :[ Successful ]
Enter the KeyStore Password:
<Enter the custom password set for custom certificate>
This prompt will come because of step #3

-- Configuring SafeNet Luna HSM --
Please enter SafeNetHSM Slot Number: 0
Please enter SafeNet HSM password (input suppressed):
Configuration stored in: 'application.properties'. (Note: You can also
use keyhsm settings to quickly view your current configuration)
Configuration saved in 'application.properties' file
```

5. Copy the custom certificate at the base location of keyhsm which is `cd /usr/share/keytrustee-server-keyhsm`.
For example,

```
cp /root/keystore.jks keystore
```

File name of the custom certificate should be keystore. You can provide custom password.

6. Start the keyhsm service.

```
service keyhsm start
```

7. Configure KTS to trust the Key HSM server.

```
[root@dskmsktscu-4 keytrustee-server-keyhsm]# keyhsm trust /path/to/key_
trustee_server/cert
[root@dskmsktscu-4 keytrustee-server-keyhsm]# ktadmin keyhsm --server h
ttps://<fqdn>:9090 --trust
e.g. ktadmin keyhsm --server https://dskmsktscu-4.dskmsktscu.root.hwx.si
te:9090 --trust
```

8. Restart keyhsm.

```
service keyhsm restart
```

9. Restart the KTS from Cloudera Manager UI.

Working with an HSM

How to integrate Cloudera Data Encryption components to provide enterprise data encryption solutions.

Ranger KMS and Key Trustee Server (KTS)

Consists of Ranger KMS providing enterprise-grade key management and the Key Trustee Server key store that stores and manages cryptographic keys and other security artifacts.

1. Install Ranger KMS backed by KTS using CM Administration Security HDFS Encryption Wizard .

Ranger KMS, KTS, and Key HSM

Consists of Ranger KMS, KTS and Key HSM which provides seamless integration of all Cloudera encryption components with a HSM added.

1. Install Ranger KMS backed by KTS using CM Administration Security HDFS Encryption Wizard .
2. Obtain and Integrate one of the following hardware security modules (HSM) supplied by a vendor.
 - Luna 7
 - CipherTrust
 - GCP Cloud HSM
 - Azure Key Vault

Setting up Luna 7 HSM for KTS and Key HSM

How to integrate Ranger KMS, KTS, and KeyHSM with with the Luna 7 HSM appliance supplied by SafeNet.

About this task

The task described in this procedural section guides you through setting up the Luna 7 hardware security module (HSM) supplied by SafeNet, for use with Ranger components supplied by Cloudera. The process includes setting up Luna 7 HSM on a client (host), installing KeyHSM and using Luna 7 HSM to validate keys.

Before you begin

You must:

- Acquire the Luna 7 HSM from SafeNet.
- Have both Ranger Key Management System and Key Trustee Server installed in your environment.
- Get KeyHSM software.

See related topics for more information about installing Ranger KMS and KTS to store keys.

Procedure

Set Up the Luna 7 Client



Note: Perform the following steps on both active and passive KTS nodes.

1. SSH to (active or passive) KTS node.

```
alternatives --install /usr/bin/java java /usr/java/jdk1.8.0_232-cloudera/bin/java 1
```

2. Untar the Luna 7 client.

```
tar -xvf safenet-linux-64bit-client-7.3.2.tar
```

the LunaClient_7.3.0-x_Linux/ folder gets created.

3. Navigate to the Luna client folder.

```
cd LunaClient_7.3.0-x_Linux/64/
```

4. In the Luna client folder, install Luna products and components.



Note: Before the install begins, you must select network HSM and JSP as shown in the example.

```
<LunaHome>/64/install.sh
```

Example:

- a) At the (y/n) prompt, choose y.

If you select no or n, this product will not be installed.

- b) At the Products prompt, choose Luna products to be installed:

- [1]: Luna Network HSM
- [2]: Luna PCIe HSM
- [3]: Luna USB HSM
- [4]: Luna Backup HSM
- [N|n]: Next
- [Q|q]: Quit

Enter selection: 1, then enter selection n.

- c) At the Components prompt, choose Luna Components to be installed

- [1]: Luna SDK
- [2]: Luna JSP (Java)
- [3]: Luna JCPProv (Java)
- [B|b]: Back to Products selection
- [I|i]: Install
- [Q|q]: Quit

Enter selection: i, then enter selection Q.

Enter selection: 1,2,and 3 then type i.

5. Register the HSM on this client.

- a) Retrieve the HSM's public key:

```
$ scp admin@luna-2.atx.cloudera.com:server.pem .
```

- b) Register the HSM on the client machine.

```
$ /usr/safenet/lunaclient/bin/vtl addServer -n luna-2.atx.cloudera.com -  
c server.pem
```

- c) Confirm the HSM has been added.

```
$ /usr/safenet/lunaclient/bin/vtl list
```

you should see the following:

ls

new server <luna.server.name> successfully added to server list

6. Create client certificate.

```
$ /usr/safenet/lunaclient/bin/vtl createCert -n $(hostname -f)  
where $(hostname -f) is the ip address if running on a virtual machine.
```



Note: The Luna appliance must be able to connect to the hostname across your network.

7. Send the client's public key created in the previous step to the HSM.

```
$ scp /usr/safenet/lunaclient/cert/client/$(hostname -f).pem
```

```
$ scp /usr/safenet/lunaclient/cert/client/$(hostname -f).pem admin@luna-  
2.atx.cloudera.com.
```


8. Register the client on the HSM.**a) SSH to the HSM.**

```
$ ssh admin@luna-2.atx.cloudera.com
```

b) Register the client with a friendly name on the HSM.

```
lunaclient> client register -client <friendly.name> -h <hostname.from.step 5.a>
```

```
[luna-2] lunash:> client register -client
dsranktkmslunahsm-4.dsranktkmslunahsm.root.hwx.site -h
dsranktkmslunahsm-4.dsranktkmslunahsm.root.hwx.site
```

c) Assign a partition to the client.

```
lunaclient> client assignpartition -client <friendly name> -partition par1
```



Note: This example assumes only access to partitions 1 and 2.

```
[luna-2] lunash:> client assignpartition -client dsranktkmslunahsm-4.dsranktkmslunahsm.root.hwx.site -partition par1
```

```
[luna-2] lunash:> client register -client dsranktkmslunahsm-4.dsranktkmslunahsm.root.hwx.site -h dsranktkmslunahsm-4.dsranktkmslunahsm.root.hwx.site
'client register successful.
Command result : 0 (Success)
[luna-2] lunash:> client assignpartition -client dsranktkmslunahsm-4.dsranktkmslunahsm.root.hwx.site -partition par1
'client assignPartition successful.
Command result : 0 (Success)
```

9. Verify registration on the client.

```
$ /usr/safenet/lunaclient/bin/vtl verify
```

```
root@dsranktkmslunahsm-4 /usr/safenet/lunaclient/bin/vtl verify
```

The following Luna SA Slots/Partitions were found:

Slot	Serial #	Label
0	462309014	par1

Install and Configure HSM

Note: Perform the following steps on both active and passive KTS node.

10. SSH to active/passive KTS node.**11. Obtain Key HSM software.****12. Install Key HSM software.**

```
# rpm -ivh keytrustee-keyhsm-*.rpm
```

13. Move the Key Trustee Server and Key HSM installation directory.

```
cd /usr/share/keytrustee-server-keyhsm/
```



Note: For Luna 7, you must add the keyhsm user to the hsmgroup group to provide that user access to certificates used by the Luna client software.

```
usermod -G keytrustee,hsmusers keyhsm
```

14. Configure Key HSM to use SafeNet Luna client.

- a) Run # keyhsm setup luna

```
# keyhsm setup luna
```

- b) Use the hostname and any port above 1024)

The recommended port is 9090.

- c) Provide data about the HSM slot.

```
# service keyhsm setup luna
-- Configuring keyHsm General Setup --
Please enter keyHsm SSL listener IP address: oks-hsm.vpc.cloudera.com
Please enter keyHsm SSL listener PORT number: 9090
validate Port:                               :[ Successful ]

-- Configuring SafeNet Luna HSM --
Please enter SafeNetHSM Slot Number: 0
Please enter SafeNet HSM password (input suppressed):
Configuration stored in: 'application.properties'. (Note: You can also
use service keyHsm settings to quickly view your current configuration)
Configuration saved in 'application.properties' file
```

15. Validate the Key HSM service.

```
$ service keyhsm validate
```

```
Check Key HSM is stopped           :[Successful]
Configuration Available            :[Successful]
Port 127.0.0.1:9090 available      :[Successful]
Unlimited-Strength JCE              :[Successful]
Validate cipher list               :[Successful]
HSM availability                   :[Successful]
All services available:            :[Successful]
```

16. Start the Key HSM service.

```
$ service keyhsm start
```

17. Configure Key HSM to trust KTS.

Note: Must be the full path to the file.

```
$ keyhsm trust /var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee.
pem
```

18. Configure KTS to trust the Key HSM server.

```
$ ktadmin keyhsm --server http://$(hostname -f):<port configured in step 14.b> --trust
```

```
$ktadmin keyhsm --server http://127.0.0.1:9090 --trust
```

19. Restart Key HSM.

```
$ service keyhsm restart
```

20. Restart the KTS from Cloudera Manager UI.**21. Test the HSM.**

```
curl -k https://$(hostname -f):11371/test_hsm
```

22. Login Ranger UI using keyadmin user role for creating an encryption zone key.

Validating Keys in Luna HSM

23. Login to Luna HSM machine.

```
ssh admin@luna-2.atx.cloudera.com
```

24. [luna-2] lunash:>partition showContents -par par1**25. Enter the password for the partition and the Keys will be visible as partition objects.****Results**

Ranger KMS is successfully started.

What to do next

You can now create Encryption zone keys using hadoop command or from Ranger UI using credentials of keyadmin user. Optionally, you can change the default encryption algorithm for KeyHSM and Luna 7.

Configuring encryption algorithms for Luna 7

How to change the default encryption algorithm for KeyHSM and Luna 7.7.

About this task

KeyHSM supports configuring the specific encryption algorithm used by the Luna 7.7 HSM. This section describes how to configure which specific encryption algorithm Luna 7 uses, by replacing the KeyHSM default algorithm with one of the optional supported algorithms.



Note: Do not change the algorithm value after creating encryption zone keys and encryption keys. Changing the algorithm value will impact the key retrieval and encryption zone operations.

Procedure

1. Stop the KeyHSM service.
2. Navigate to the KeyHSM root directory.
3. In the KeyHSM root dir, open the application.properties file.
4. Find the hsm.luna.encryption.algorithm property, with default value=RSA/ECB/PKCS1Padding.

5. Edit the application properties file to replace the default value with one of the following ones:

Encryption algorithms supported by KeyHSM/Luna 7.7 HSM:

- RSA/ECB/PKCS1Padding (default)
- RSA
- RSA_ECB_OAEP_SHA_224AndMGF1Padding
- AES_RSA_NONE_OAEP_SHA_512AndMGF1Padding
- AES_CBC_PKCS5Padding
- RSA_NONE_OAEP_WITH_SHA224AndMGF1Padding
- RSA_NONE_OAEP_WITH_SHA256AndMGF1Padding
- RSA_NONE_OAEP_WITH_SHA384AndMGF1Padding
- RSA_NONE_OAEP_WITH_SHA1AndMGF1Padding

Upgrade Scenario:



Note: When upgrading KeyHSM from any version < 7.1.7.1 to 7.1.7.1, zone keys and hdfs encryption zone(s) will exist. Do not change the encryption algorithms without first creating new encryption keys, using the following steps:

- a. Unlock the encryption zones with existing keys.
- b. Backup the zone data.
- c. Stop the KeyHSM.
- d. Change the algorithm value as described previously.
- e. Start the KeyHSM.
- f. Create new keys using the new algorithm value.
- g. Lock the encryption zone with new keys.

Setting up GCP Cloud HSM for KTS and Key HSM

How to integrate Ranger KMS and KTS with the Google Cloud Platform (GCP) HSM service.

About this task

This task describes how to set up the Google Cloud Platform (GCP) hardware security module (HSM) service provided by Google. The process includes setting up the GCP HSM service on a client (host), setting up KeyHSM and using the GCP HSM to validate keys.

Before you begin

You must:

- Log in to the Google cloud console using your account. (Requires Google account access).
- Have Ranger Key Management System, Key Trustee Server and Key HSM installed in your environment.
- Have Java (jdk1.8.0.232) installed.

See related topics for more information about installing Ranger KMS, KTS and KeyHSM.

Procedure

Set Up Google Cloud HSM

1. Login to Google Cloud console using Cloudera account.
2. Create a generic service account by selecting or creating the Project.
3. Create the key for that generic service account you created in the previous step.

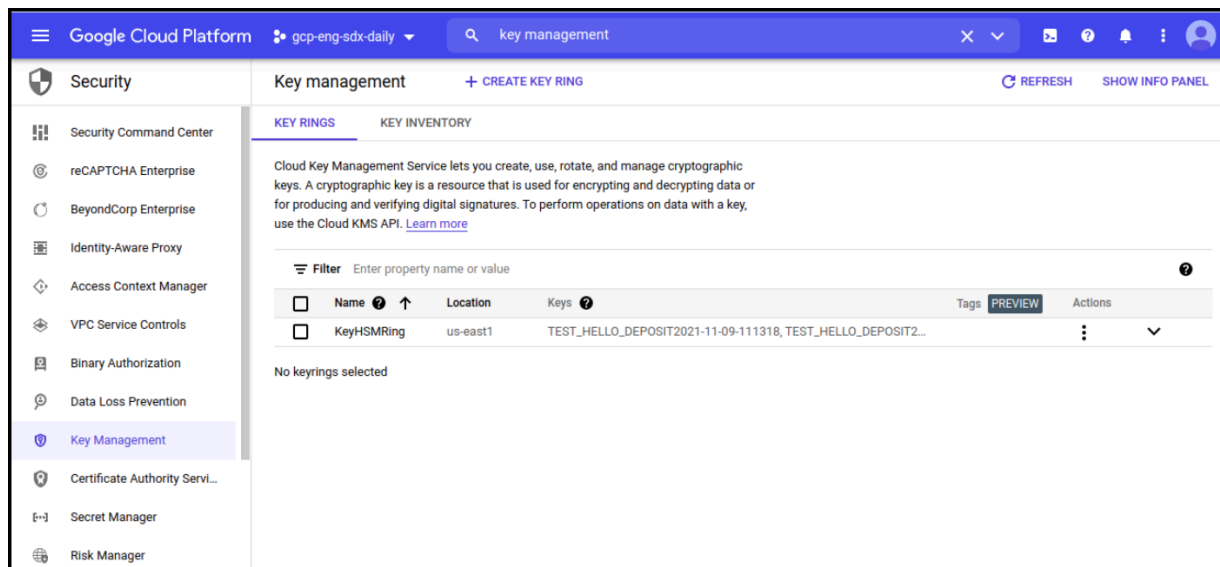
- Download and save your generic service account key in JSON format.



Note: You will use the generic service account key later in this process. Record the project ID, Location ID and JSON file.

- In GCP Console Key Management create the key ring.

Figure 1: Creating a key ring in Google Cloud Platform



This example shows a project gcp-eng-sdx-daily, service account keyhsm, and key ring KeyHSMRing.

Integrate GCP with KeyHSM



Note: Perform the following steps on both active and passive KTS nodes.

- In your Key HSM root directory, copy the authentication key (json file) you created in the setup process, and provided the appropriate access.

```
# rpm -ivh keytrustee-keyhsm-*.rpm
```

```
cd /usr/share/keytrustee-server-keyhsm/
chown keyhsm:keytrustee <key.filename>.json
```

- Set up the GCP HSM.

```
keyhsm setup googlecloudhsm
```

```
keyhsm.management.address=127.0.0.1
keyhsm.server.port=9090
google.cloud.hsm.key.ring=KeyHSMRing

# Google App Credential File
google.cloud.hsm.app.cred.file=<authentication file>.json

# Google HSM Project Id
google.cloud.hsm.project.id=<project ID>

# Google HSM Location Id
```

```
google.cloud.hsm.location.id=<location ID>
```



Note: Use the key ring, authentication file, project ID, and location ID strings you created during setup.

8. Validate the Key HSM service.

```
$ service keyhsm validate
```

```
Check Key HSM is stopped           :[Successful]
Configuration Available            :[Successful]
Port 127.0.0.1:9090 available      :[Successful]
Unlimited-Strength JCE              :[Successful]
Validate cipher list               :[Successful]
HSM availability                   :[Successful]
All services available:            :[Successful]
```

9. Start the Key HSM service.

```
$ service keyhsm start
```

10. Configure KTS to trust the Key HSM server.

```
$ ktadmin keyhsm --server http://$(hostname -f):<port configured in setup>
--trust
```

```
$ktadmin keyhsm --server http://127.0.0.1:9090 --trust
```

11. Restart Key HSM.

```
$ service keyhsm restart
```

12. Restart the KTS from Cloudera Manager UI.

13. Test the HSM.

```
curl -k https://$(hostname -f):11371/test_hsm
```

14. Login to the Ranger UI using keyadmin user role for creating an encryption zone key.

Results

Keys will be created in the Key ring on GCP.

What to do next

Further keys for zone operation can be created using Ranger UI with keyadmin role credentials and also using hadoop commands.

Setting up CipherTrust HSM for KTS and Key HSM

Learn how to integrate Ranger KMS, KTS, and Key HSM with the CipherTrust HSM appliance.

About this task

This task describes how to set up the CipherTrust hardware security module (HSM) appliance provided by Thales. The process describes configuring the NAE port using CipherTrust Manager, setting up and configuring Key HSM in your cluster, and validating keys using CipherTrust Manager.

Before you begin

You must have installed the following in your environment:

- Thales CipherTrust Manager
- Ranger Key Management System, Key Trustee Server and Key HSM
- Java (jdk1.8.0.232)

For more information about installing Ranger KMS, KTS and Key HSM, see *Configuring NAE port in Thales CipherTrust Manager*

Procedure

1. Log in to Thales CipherTrust Manager.
2. In CipherTrust Manager Admin Settings , select Add Interface.
3. In Type, Select NAE (default).
4. In Network Interface, selectAll.
5. In Port, type a value for the port number.
9000
6. In Mode, select one of the following options to match your environment:
 - No TLS, user must supply password.
 - TLS, Ignore client cert. user must supply password.
7. Click Add.
8. Create a user.
 - a) In Access Management Users , click Create New User .
 - b) In Create a New User, provide a username, password, and other required information.
 - c) Click Create.

Setting up a cluster and configuring Key HSM



Note: Perform the following steps on both active and passive KTS nodes.

9. In your Key HSM root directory, make sure that appropriate versions of Key HSM files are available with proper permissions.

```
cd /usr/share/keytrustee-server-keyhsm/
```



Note:

Cipher Trust HSM supports two Key HSM versions.

- a. If using Ingrian v6.x, then copy all JAR files into this folder and make sure you have provided the required permissions.
- b. If using Ingrian v8.12.x, then copy all the JAR files except gson-2.1.jar into this folder and ensure that you have provided the required permissions.

10. If SSL is enabled on CipherTrust Manager run the following command:

```
echo "thales_machine_ip nae.keysecure.local" >> /etc/hosts
```

11. Setup Key HSM service.

```
keyhsm setup keysecure
```

```
-- Configuring keyHsm General Setup --
Cloudera Recommends to use 127.0.0.1 as the listener port for Key HSM
Please enter Key HSM SSL listener IP address: [127.0.0.1] Hit Enter
Will attempt to setup listener on 127.0.0.1
```

```

Please enter Key HSM SSL listener PORT number: 9090

validate Port:                                :[ Successful ]

-- Ingrian HSM Credential Configuration --
Please enter HSM login USERNAME: username
Please enter HSM login PASSWORD: password

Please enter HSM IP Address or Hostname: 18.218.251.172
Please enter HSM Port number: 9000
Valid address:                                :[ Successful ]

Use SSL? [Y/n] Y (If TSL is enabled on NAE port then press Y else type n
and hit enter and act accordingly)
org.bouncycastle.cert.X509CertificateHolder@f20f09ff
org.bouncycastle.cert.X509CertificateHolder@ebb30faf
Trust this server? [y/N] y

Trusted server:                                :[ Successful ]

```

12. Validate the Key HSM service.

```
$ service keyhsm validate
```

13. Start the Key HSM service.

```
$ service keyhsm start
```

14. Configure Key HSM to trust KTS.



Note: You must provide the full path to the file.

```
$ keyhsm trust /var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee.
pem
```

15. Configure KTS to trust the Key HSM server.

```
$ ktadmin keyhsm --server http://127.0.0.1:9090 --trust
```

16. Restart the Key HSM service.

```
$ service keyhsm restart
```

17. Restart the KTS from Cloudera Manager UI.

18. Test the HSM.

```
curl -k https://$(hostname -f):11371/test_hsm
```

19. Login to the Ranger UI using keyadmin user role for creating an encryption zone key and do further validation.

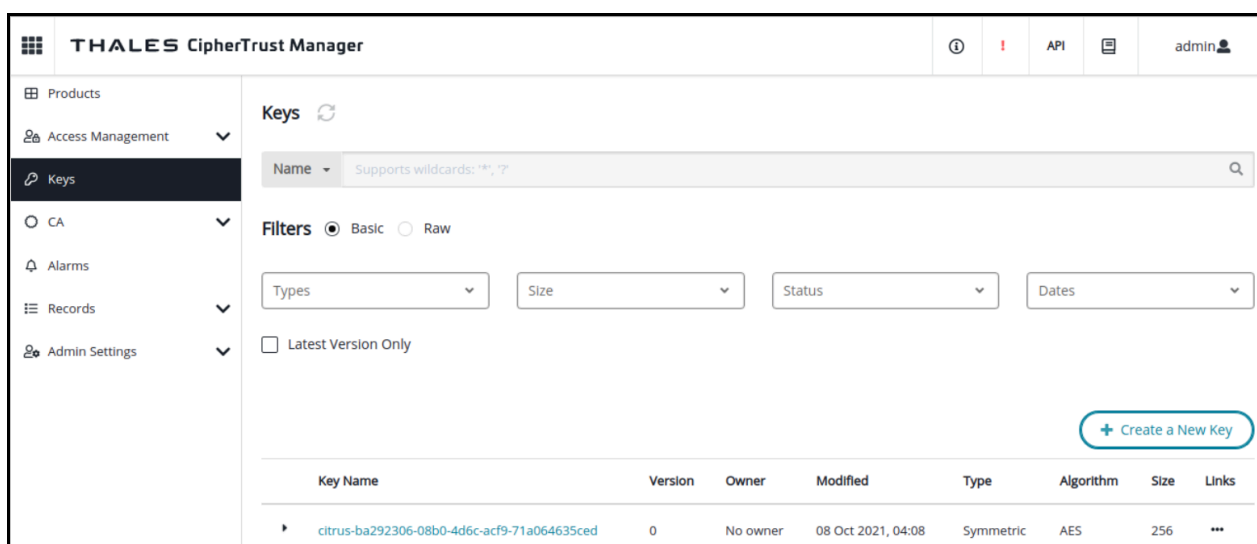
Validating Keys in Cipher Trust HSM

20. In Thales Cipher Trust Manager Left Navigation Panel , click Keys.

Results

Keys created in the above steps should be present, as shown in the following image:

Figure 2: Validating Keys in CipherTrust Manager



What to do next

Further keys for zone operation can be created using Ranger UI with keyadmin role credentials and also using hadoop commands.

Connecting KeySecure HSM to CipherTrust Manager after migration from Key Secure HSM

How to configure the KeySecure HSM to connect to CipherTrust.

About this task

After the Thales team successfully migrates the keys from Key Secure HSM to CipherTrust Manager, you must configure the Key HSM to connect to CipherTrust. You must perform the following steps on both the Active and Passive KTS nodes.

Before you begin

The Thales team must have successfully migrated the keys from Key Secure HSM to CipherTrust Manager.

Procedure

1. Stop the Key HSM service.

```
$ service keyhsm stop
```

2. Back up the existing application.properties file.
3. If SSL is enabled on CipherTrust Manager, run the following command:

```
$ echo "thales_machine_ip nae.keysecure.local" >> /etc/hosts
```

4. Set up the Key HSM service.

```
$ keyhsm setup keysecure
```

```
-- Configuring keyHsm General Setup --
Cloudera Recommends to use 127.0.0.1 as the listener port for Key HSM
Please enter Key HSM SSL listener IP address: [127.0.0.1]
Will attempt to setup listener on 127.0.0.1
```

```

Please enter Key HSM SSL listener PORT number: 9090

validate Port:                                :[ Successful ]

-- Ingrian HSM Credential Configuration --
Please enter HSM login USERNAME: testuser (user created on CipherTrust
Manager)
Please enter HSM login PASSWORD:

Please enter HSM IP Address or Hostname: ec2-3-144-233-194.us-east-2.com
pute.amazonaws.com
Please enter HSM Port number: 9000
Valid address:                                :[ Successful ]

Use SSL? [Y/n] (As per the configuration done on CipherTrust Manager)

Configuration saved in 'application.properties' file
Configuration stored in: 'application.properties'. (Note: You can also use
keyhsm settings to quickly view your current configuration)

```

5. Validate the Key HSM.

```
$ service keyhsm validate
```

```

Check Key HSM is stopped                :[ Successful ]
Configuration Available                  :[ Successful ]
Port 127.0.0.1:9090 available           :[ Successful ]
Unlimited-Strength JCE                   :[ Successful ]
Validate cipher list                     :[ Successful ]
HSM availability                         :[ Successful ]
All services available:                  :[ Successful ]

```

6. Start the Key HSM service.

```
$ service keyhsm start
Starting KeyHSM, please wait...
```

7. Configure Key HSM to trust KTS by providing the full path to the file.

```
$ keyhsm trust /var/lib/keytrustee/.keytrustee/.ssl/ssl-cert-keytrustee.
pem
```

8. Configure KTS to trust the Key HSM server.

```
$ ktadmin keyhsm --server http://127.0.0.1:9090 --trust
```



Note: Perform the next two steps after the above steps are successfully completed on both Active and Passive KTS nodes.

9. Restart KTS from Cloudera Manager.

10. Restart Ranger KMS KTS service from Cloudera Manager.