THALES

# Docker Container: Integration Guide

## THALES LUNA HSM AND LUNA CLOUD HSM

## Document Information

| | |
|---|---|
| **Document Part Number** | 007-000131-001 |
| **Revision** | C |
| **Release Date** | 7 November 2022 |

## Trademarks, Copyrights, and Third-Party Software

# CONTENTS

# Overview

Docker makes it easier to create, deploy, and run applications by using containers. Containers allow a developer to package an application with all the parts that it needs and then ship the application and its components as a single package.

This guide demonstrates how to integrate Luna HSM or Luna Cloud HSM with an application that has been containerized using Docker. This guide includes configurations for the container orchestrators Docker Swarm, Kubernetes, Openshift, and Apache Mesos. This guide provides a Java Code Signer use case as an example.

> **Note**: Java Code Signing is not the only use case. Various other applications can be deployed inside Docker Container and benefit from the integration with a Luna HSM or Luna Cloud HSM service.

The benefits of integrating an HSM with Docker Container include:

- Secure generation, storage, and protection of the signing private keys on FIPS 140-2 level 3 validated hardware.
- Full life cycle management of the keys.
- HSM audit trail.*
- Take advantage of cloud services with confidence.
- Significant performance improvements by off-loading cryptographic operations from application servers

*Luna Cloud HSM service does not have access to the secure audit trail

# Certified Platforms

This integration is certified on the following platforms:

Certified platforms on Luna HSM

Certified platforms on Luna Cloud HSM

### Certified platforms on Luna HSM

| HSM Type | Platforms Tested |
|---|---|
| Luna HSM | RHEL 7<br>RHEL 8 |

**Luna HSM:** Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. The Luna HSM on premise offerings include the Luna Network HSM, Luna PCIe HSM, and Luna USB HSMs. Luna HSMs are also available for access as an offering from cloud service providers such as IBM cloud HSM and AWS cloud HSM classic.

**Certified platforms on Luna Cloud HSM**

| HSM Type | Platforms Tested |
|---|---|
| Luna Cloud HSM | RHEL 7 |

**Luna Cloud HSM:** Luna Cloud HSM platform provides a wide range of cloud-based HSM and Key Management services through a simple graphical user interface. With Luna Cloud HSM, security is simple, cost effective and easy to manage because there is no hardware to buy, deploy and maintain. As an Application Owner, you click and deploy services, generate usage reports and maintain just the services you need.

# Prerequisites

Complete the following prerequisites before proceeding with this integration:

Set up Luna HSM

Set up Luna Cloud HSM Service

## Set up Luna HSM

If you are using Luna HSM:

1. Verify the HSM is set up, initialized, provisioned, and ready for deployment. Refer to the *Luna HSM Product Documentation* for more information.
2. Create a partition that will be used by Docker Container.
3. If using a Luna Network HSM, register a client for the system and assign the client to the partition to create an NTLS connection. Initialize the Crypto Officer and Crypto User roles for the registered partition.
4. Ensure that the partition is successfully registered and configured. The command to see the registered partitions is:

```
C:\Program Files\SafeNet\LunaClient>lunacm.exe

lunacm.exe (64-bit) v10.3.0-273. Copyright (c) 2020 SafeNet. All rights
reserved.

Available HSMs:

   Slot Id ->              0

   Label ->                INTG_Par01

   Serial Number ->        1238696044952

   Model ->                LunaSA 7.4.0

   Firmware Version ->     7.4.0

   Configuration ->        Luna User Partition With SO (PW) Key Export With
   Cloning Mode

   Slot Description ->     Net Token Slot

   FM HW Status ->         Non-FM
```

5. For PED-authenticated HSM, enable partition policies 22 and 23 to allow activation and auto-activation.

> **NOTE:** Refer to the Luna HSM documentation for detailed steps on creating NTLS connection, initializing the partitions, and assigning various user roles.

### Set up Luna HSM High-Availability

Refer to Luna HSM documentation to gain an understanding of the steps involved in setting up Luna HSM High Availability and configuring and setting up two or more HSM boxes on host systems. You must enable the HAOnly setting in HA for the failover to work so that if the primary goes down due to any reason, all the calls are automatically routed to the secondary until the primary recovers and starts up.

### Set up Luna HSM in FIPS Mode

> **NOTE:** This setting is not required for Universal Client. This setting is applicable only for Luna Client 7.x.

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for operation in a FIPS-compliant HSM. If you are using the Luna HSM in FIPS mode, you have to make the following change in the configuration file:

```
[Misc]
RSAKeyGenMechRemap=1
```

The above setting redirects the older calling mechanism to a new approved mechanism when Luna HSM is in FIPS mode.

## Set up Luna Cloud HSM Service

You can set up Luna Cloud HSM Service in the following ways:

- **Set up standalone Cloud HSM service using minimum client package**
- **Set up standalone Cloud HSM service using full Luna client package**
- **Set up Luna HSM and Luna Cloud HSM service in hybrid mode**
- **Set up Luna Cloud HSM Service in FIPS mode**

> **NOTE:** Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

### Set up standalone Cloud HSM service using minimum client package

To set up standalone Luna Cloud HSM service using minimum client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.

3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

   ```
   [Windows]
   cvclient-min.zip
   [Linux]
   cvclient-min.tar
   # tar -xvf cvclient-min.tar
   ```

4. Run the **setenv** script to create a new configuration file containing information required by the Luna Cloud HSM service.

   ```
   [Windows]
   Right-click setenv.cmd and select Run as Administrator.
   [Linux]
   Source the setenv script.
   # source ./setenv
   ```

   Run the **LunaCM** utility and verify the Cloud HSM service is listed.

### Set up standalone Cloud HSM service using full Luna client package

To set up standalone Luna Cloud HSM service using full Luna client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

   ```
   [Windows]
   cvclient-min.zip
   [Linux]
   cvclient-min.tar
   # tar -xvf cvclient-min.tar
   ```

4. Run the **setenv** script to create a new configuration file containing information required by the Luna Cloud HSM service.

   ```
   [Windows]
   Right-click setenv.cmd and select Run as Administrator.
   [Linux]
   Source the setenv script.
   # source ./setenv
   ```

5. Copy the server and partition certificates from the Cloud HSM service client directory to Luna client certificates directory:

   **Cloud HSM Certificates:**

   ```
   server-certificate.pem
   partition-ca-certificate.pem
   partition-certificate.pem
   ```

   **LunaClient Certificate Directory:**

   ```
   [Windows default location for Luna Client]
   C:\Program Files\Safenet\Lunaclient\cert\
   [Linux default location for Luna Client]
   /usr/safenet/lunaclient/cert/
   ```

   **NOTE:** Skip this step for Luna Client v10.2 or higher.

6. Open the configuration file from the Cloud HSM service client directory and copy the **XTC** and **REST** section.

   ```
   [Windows]
   crystoki.ini
   [Linux]
   Chrystoki.conf
   ```

7. Edit the Luna Client configuration file and add the **XTC** and **REST** sections copied from Cloud HSM service client configuration file.

8. Change server and partition certificates path from step 5 in **XTC** and **REST** sections. Do not change any other entries provided in these sections.

   [XTC]

   ```
   . . .
   PartitionCAPath=<LunaClient_cert_directory>\partition-ca-certificate.pem
   PartitionCertPath00=<LunaClient_cert_directory>\partition-certificate.pem
   . . .

   [REST]
   . . .
   SSLClientSideVerifyFile=<LunaClient_cert_directory>\server-certificate.pem
   . . .
   ```

   **NOTE:** Skip this step for Luna Client v10.2 or higher.

9. Edit the following entry from the **Misc** section and update the correct path for the **plugins** directory:

```
Misc]
PluginModuleDir=<LunaClient_plugins_directory>

[Windows Default]

C:\Program Files\Safenet\Lunaclient\plugins\

[Linux Default]

/usr/safenet/lunaclient/plugins/
```

10. Save the configuration file. If you wish, you can now safely delete the extracted Cloud HSM service client directory.
11. Reset the **ChrystokiConfigurationPath** environment variable and point back to the location of the Luna Client configuration file.

    **Windows**

    In the Control Panel, search for "environment" and select **Edit the system environment variables**. Click **Environment Variables**. In both list boxes for the current user and system variables, edit **ChrystokiConfigurationPath** and point to the **crystoki.ini** file in the Luna client install directory.

    **Linux**

    Either open a new shell session, or export the environment variable for the current session pointing to the location of the **Chrystoki.conf** file:

    ```
    # export ChrystokiConfigurationPath=/etc/
    ```

12. Run the **LunaCM** utility and verify that the Cloud HSM service is listed. In hybrid mode, both Luna and Cloud HSM service will be listed.

> **NOTE:** Follow the Luna Cloud HSM documentation for detailed steps for creating service, client, and initializing various user roles.

### Set up Luna HSM and Luna Cloud HSM service in hybrid mode

To set up Luna HSM and Luna Cloud HSM service in hybrid mode, follow the steps mentioned under the Set up standalone Cloud HSM service using full Luna client package section above.

> **NOTE:** Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

### Set up Luna Cloud HSM Service in FIPS mode

Luna Cloud HSM service operates in both FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, ensure you enable the **Allow non-FIPS approved algorithms** check box when configuring your Cloud HSM service. The FIPS mode is enabled by default. Refer to the Mechanism List in the SDK Reference Guide for more information about available FIPS and non-FIPS algorithms.

# Integrating Luna Cloud HSM service client with Docker Container

Create and run the Luna Docker image to use the Luna Cloud HSM service inside the Docker Container.

## Create Luna Docker image

To create Luna Docker image:

1. Create file Dockerfile in the current working directory and add the following entries:

```
FROM centos:centos7
RUN mkdir -p /usr/local/luna
COPY ForDocker_client.zip /usr/local/luna
ENTRYPOINT  /bin/bash
#End of the Dockerfile
```

2. Build a Docker Image.

```
# docker build . -t dpod-in-docker
```

3. Verify the Docker image is created.

```
# docker images
```

4. Start the docker container with the following command.

```
# docker run –it dpod-in-docker –name dpod-in-docker
```

Now you are inside the Docker Container.

## Configure Luna Cloud HSM service inside Docker Container

To configure Luna Cloud HSM service inside Docker Container:

1. Change the directory to /usr/local/luna.
2. Unzip the client package.

```
# unzip ForDocker_client.zip
```

The above Client zip package contains:

- Chrystoki.conf
- crystoki-template.ini
- cvclient-min.tar
- cvclient-min.zip
- EULA.zip
- partition-ca-certificate.pem
- partition-certificate.pem
- server-certificate.pem

3. Untar the cvclient-min.tar.

   ```
   # tar xfv cvclient-min.tar
   ```

4. Set the environment variable.

   ```
   # source ./setenv
   ```

5. Start LunaCM to verify the NTLS connection.

   ```
   # ./bin/64/lunacm
   ```

6. Set the active slot to the uninitialized application partition:

   ```
   lunacm:> slot set -slot <slotnum>
   ```

7. Initialize the application partition, to create the partition's Security Officer (SO), and set the initial password and cloning domain.

   ```
   lunacm:> partition init -label <par_label>
   ```

8. Log in as Partition SO. You can also use the shortcut po.

   ```
   lunacm:> role login -name Partition SO
   ```

9. Initialize the Crypto Officer role and set the initial password. You can also use the shortcut co.

   ```
   lunacm:> role init -name Crypto Officer
   ```

10. The Partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. You must log out to allow the Crypto Officer to log in with the newly-set password.

    ```
    lunacm:> role logout
    ```

    **Note**: Once the Crypto Officer logs in and changes the initial credential set by the Partition SO, applications using the CO's challenge secret/password can perform cryptographic operations in the partition. The Crypto Officer can create, modify and delete crypto objects within the partition, and use existing crypto objects (sign/verify). You can also create a limited-capability role called Crypto User that can use the objects created by the Crypto Officer, but cannot modify them. The separation of roles is important in some security regimes and operational situations, and where you might be required to satisfy audit criteria for industry or government oversight.

11. Log in as the Crypto Officer.

    ```
    lunacm:> role login -name Crypto Officer
    ```

    **Note**: The password for the Crypto Officer role is valid for the initial login only. You must change the initial password using the command role changepw during the initial login session, or a subsequent login. Failing to change the password will result in a CKR_PIN_EXPIRED error when you perform role-dependent actions.

12. If you have not already done so, change the initial password set by the Partition SO.

    ```
    lunacm:> role changepw -name Crypto Officer
    ```

13. Create the Crypto User. You can also use the shortcut cu.

    ```
    lunacm:> role init -name Crypto User
    ```

    The Crypto User can now log in with the credentials provided by the Crypto Officer, and change the initial password. The Crypto User can now use applications to perform cryptographic operations using keys and objects created in the partition by the Crypto Officer.

Refer to Using Luna Cloud HSM service client inside Docker Container for an application demonstration inside a Docker using Luna Cloud HSM.

# Integrating Docker Container for Luna Network HSM

Luna Network HSMs provide strong physical protection of secure assets, including keys, and should be considered a best practice when working with Docker containers. Using the Luna HSM with Docker container requires the Luna minimal client. The minimal client installation contains the run-time libraries required for a cryptographic application to connect to the Luna Network HSM using PKCS#11 or Java APIs.

> **NOTE**: Please refer the APPENDIX B: Using Luna Client from Host to Docker Container if you want to use Luna Client configured on the host system via Docker volumes without installing any client inside the container.

Install the Luna minimal client inside of a Docker container and configure the Docker container to communicate with the Luna Network HSM. Complete the following to configure your Docker container to use a Luna Network HSM:

> Configure Luna Minimal Client for Docker Container

> Create NTLS connection to Luna Network HSM

> Create Luna Client Docker image

> Run Docker Container

## Configure Luna Minimal Client for Docker Container

Install the Luna minimal client. The minimal client contains the run-time libraries required for a cryptographic application to connect to the Luna Network HSM using PKCS#11 or Java APIs. To install the Luna minimal client in Docker container:

1. Install the full Luna HSM Client software (non-minimal) on the Docker host.
2. Create a directory. In this example:

   ```
   $HOME/luna-docker
   ```
3. Create the following subdirectories under the first directory:

   ```
   $HOME/luna-docker/config
   ```

   ```
   $HOME/luna-docker/config/certs
   ```

   Additionally, if you are configuring STC:

   ```
   $HOME/luna-docker/config/stc
   ```

   ```
   $HOME/luna-docker/config/stc/token/001
   ```

   Create the empty file:

   ```
   $HOME/luna-docker/config/stc/token/001/token.db
   ```

   > **NOTE**: The contents of the config directory are needed by the Docker Container.
4. Copy the Luna Minimal Client tarball to $HOME/luna-docker.

5. Untar the Luna Minimal Client tarball.

```
# tar -xf $HOME/luna-docker/LunaClient-Minimal-<release_version>.x86_64.tar -C $HOME/luna-docker
```

6. Copy the Chrystoki.conf file from the Minimal Client directory to $HOME/luna-docker/config.

```
# cp LunaClient-Minimal-<release_version>.x86_64/Chrystoki-template.conf $HOME/luna-  docker/config/Chrystoki.conf
```

7. Define the following environment variable:

```
# export ChrystokiConfigurationPath=$HOME/luna-docker/config
```

## Create NTLS connection to Luna Network HSM

Open an NTLS connection between the Docker container and the Luna Network HSM. This allows the HSM device to communicate securely with the Docker application. To create NTLS connection to Luna Network HSM:

1. Create a Luna HSM Client certificate for the Docker container.

```
# /usr/safenet/lunaclient/bin/vtl createCert -n <cert_name>
```

2. Copy the client certificate to the SafeNet Luna Network HSM appliance.

```
# scp ./certs/<cert_name>.pem admin@<Network_HSM_IP>:
```

3. Copy the appliance server certificate (server.pem) to $HOME/luna-docker/config/certs.

```
# scp admin@<Network_HSM_IP>:server.pem ./certs
```

4. Register the appliance server certificate with the client.

```
# /usr/safenet/lunaclient/bin/vtl addServer -c ./certs/server.pem -n <Network_HSM_IP>
```

5. Connect via SSH to the Luna Network HSM appliance and log in to LunaSH.

```
# ssh admin@<Network_HSM_IP>
```

6. Create a partition, if one does not already exist on the HSM.

```
# lunash:>partition create -partition <partition_name>
```

7. Register the full Luna HSM client with the appliance, and assign the partition to the client.

```
# lunash:>client register -client <client_name> {-ip <client_IP> | -hostname <client_hostname>}
```

```
# lunash:>client assignpartition -client <client_name> -partition <partition_name>
```

```
# lunash:>ntls ipcheck disable
```

```
# lunash:>exit
```

8. On the client workstation, run LunaCM, set the active slot to the registered partition and initialize it.

```
# lunacm:>slot set -slot <slotnum>
```

9. Initialize the application partition, to create the partition's Security Officer (SO), and set the initial password and cloning domain.

```
# lunacm:> partition init -label <par_label>
```

10. Log in as Partition SO. You can also use the shortcut `po`.

```
# role login -name Partition SO
```

11. Initialize the Crypto Officer role and set the initial password. You can also use the shortcut `co`.

```
# role init -name Crypto Officer
```

12. The Partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. You must log out to allow the Crypto Officer to log in with the newly-set password.

```
# role logout
```

> **NOTE**: Once the Crypto Officer logs in and changes the initial credential set by the Partition SO, applications using the CO's challenge secret/password can perform cryptographic operations in the partition. The Crypto Officer can create, modify and delete crypto objects within the partition, and use existing crypto objects (sign/verify). You can also create a limited-capability role called Crypto User that can use the objects created by the Crypto Officer, but cannot modify them. The separation of roles is important in some security regimes and operational situations, and where you might be required to satisfy audit criteria for industry or government oversight.

13. Log in as the Crypto Officer. You can also use the shortcut `co`.

```
lunacm:> role login -name Crypto Officer
```

> **NOTE**: The password for the Crypto Officer role is valid for the initial login only. You must change the initial password using the command role changepw during the initial login session, or a subsequent login. Failing to change the password will result in a CKR_PIN_EXPIRED error when you perform role-dependent actions.

14. Change the initial password set by the Partition SO, if you have not done so already.

```
lunacm:> role changepw -name Crypto Officer
```

15. Create the Crypto User. You can also use the shortcut `cu`.

```
lunacm:> role init -name Crypto User
```

The Crypto User can now log in with the credentials provided by the Crypto Officer, and change the initial password. The Crypto User can now use applications to perform cryptographic operations using keys and objects created in the partition by the Crypto Officer.

16. Change the path of the runtime libraries in config/Chrystoki.conf.

```
# sed -i -e 's#\./certs#/usr/local/luna/config/certs#g' -e
's#/usr/safenet/lunaclient/lib/libCryptoki2_64.so#/usr/local/luna/libs/64/libCr
yptoki2.so#g' -e
's#/usr/safenet/lunaclient/lib/libSoftToken.so#/usr/local/luna/libs/64/libSoftT
oken.so#g' config/Chrystoki.conf
```

## Create Luna Client Docker image

Create and run the Luna Docker Image to use the Luna Network HSM inside of the Docker Container. To create the Luna Docker Image you must create the Docker Container for use with the Luna Network HSM. To create the Luna Client Docker Image:

1. Create the file Dockerfile in the current working directory and add the following entries:

```
FROM centos:centos7
ARG MIN_CLIENT
COPY $MIN_CLIENT.tar /tmp
RUN mkdir -p /usr/local/luna
RUN tar xvf /tmp/$MIN_CLIENT.tar --strip 1 -C /usr/local/luna
ENV ChrystokiConfigurationPath=/usr/local/luna/config
COPY lunacm /usr/local/bin
COPY vtl /usr/local/bin
COPY multitoken /usr/local/bin
COPY ckdemo /usr/local/bin
ENTRYPOINT  /bin/bash
#End of the Dockerfile
```

> **NOTE**: The minimal client tarball does not include tools or files not necessary for basic operation. Copy any additional files you would like to include in the Docker image (i.e. lunacm, vtl, multitoken) to $HOME/luna-docker/ .

2. Build a Docker image.

```
# docker build . --build-arg MIN_CLIENT=LunaClient-Minimal-
<release_version>.x86_64 -t lunaclient-image
```

3. Verify the Docker image was created.

```
# docker images
```

## Run Docker Container

Once configured, you must start the Docker Container to access the associated Luna Network HSM. To run Docker Container:

1. Make the contents of the config directory available to the Containers when you create them, by mounting the config directory as a volume.

```
# docker run -it --name lunaclient -v $PWD/config:/usr/local/luna/config
lunaclient-image
```

2. From the Docker container, verify that the container has a connection to the Luna Network HSM partition.

```
# ./bin/64/lunacm
```

See Using the Luna HSM inside Docker Container for an application demonstration inside of a Docker using Luna Network HSM.

# Integrating Docker Swarm with Luna Cloud HSM

Docker can be configured in swarm mode. Swarm mode allows users to manage a cluster of Docker Engines or nodes as a single virtual system. This section demonstrates integrating a Docker Swarm configuration with a Luna Cloud HSM service. Luna Cloud HSM service provides strong physical protection of secure assets, including keys, and should be considered a best practice when working with Docker Swarm.

> **NOTE**: This integration assumes that the Docker Swarm Cluster is up and running, and that at least the Master and a single Node in the cluster exists.

This service provides your client machine with access to an HSM Application Partition for storing cryptographic objects used by your applications. Application partitions can be assigned to a single client, or multiple clients can be assigned to, and share, a single application partition. For detailed information, refer to the Configuring Luna Cloud HSM Service section.

To use a Luna Cloud HSM service inside a Docker swarm, complete the following procedures:

> **Create Luna Docker Image in Docker Registry**

> **Set up Docker Swarm Cluster**

> **Deploy application on Swarm Manager**

## Create Luna Docker Image in Docker Registry

Use Docker Registry to configure the Docker Image that you intend to integrate with the Luna Cloud HSM service. Customize the Docker image for integration with Thales software. To create the Luna Docker image in Docker Registry:

1. Download and unzip the service client package on the Master Node in a directory called /clientfiles. Copy the certificates and configuration files to a directory called /secrets. Verify the contents in each directory.

```
# ls clientfiles
bin  etc  EULA.zip  jsp  libs  setenv
# ls secrets
Chrystoki.conf partition-ca-certificate.pem partition-certificate.pem server-certificate.pem
```

2. Create a file named Dockerfile in the current working directory and add following information to this file:

```
FROM ubuntu:xenial
RUN mkdir -p /usr/local/luna
COPY clientfiles /usr/local/luna
WORKDIR /usr/local/luna/
ENV ChrystokiConfigurationPath=/usr/local/luna
#End of the Dockerfile
```

3. Build the Docker image using the new Dockerfile

```
# docker build . -t docker_swarm
```

4. Verify the image.

```
# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
docker_swarm        latest              f190c59cd551        About a minute ago
245MB
ubuntu              xenial              b9e15a5d1e1a        10 hours ago
115MB
```

5. Verify the Docker image is created.

```
# docker images
```

6. Log in to Docker Hub. Provide username and password when prompted.

```
# docker login
```

7. Tag the Docker image using the following command. Replace the <username> with your Docker Hub username.

```
# docker tag docker_swarm <username>/docker-swarm
```

8. Verify the newly tagged image is included in the Docker images list:

```
# docker images
REPOSITORY                  TAG                 IMAGE ID         CREATED
SIZE
<username>/docker-swarm     latest              f190c59cd551      2 minutes ago
245MB
docker_swarm                latest              f190c59cd551      2 minutes ago
245MB
ubuntu                      xenial              b9e15a5d1e1a      10 hours ago
115MB
```

9. Push the image to the Docker hub.

```
# docker push <username>/docker-swarm
```

Verify that the image is available now on Docker hub

> **NOTE**: You can make the Docker Hub repo private by accessing the following: Details > settings > Make private > Enter tag name > Confirm on Docker hub.

## Set up Docker Swarm Cluster

Set up the nodes in the Docker Swarm cluster for integration with the Luna Cloud HSM service. To set up a Docker Swarm Cluster:

1. Create the virtual machines for the Docker Swarm Cluster using the virtualbox driver:

```
# docker-machine create --driver virtualbox myvm1
# docker-machine create --driver virtualbox myvm2
# docker-machine create --driver virtualbox myvm3
```

2. List the virtual machines and get their ip addresses using following command:

```
# docker-machine ls
NAME    ACTIVE   DRIVER       STATE     URL                         SWARM   DOCKER        ERRORS
myvm1   -        virtualbox   Running   tcp://192.168.99.100:2376           v18.06.1-ce
myvm2   -        virtualbox   Running   tcp://192.168.99.101:2376           v18.06.1-ce
myvm3   -        virtualbox   Running   tcp://192.168.99.102:2376           v18.06.1-ce
```

3. Initialize the Swarm and add the node.

```
# docker-machine ssh myvm1 "docker swarm init --advertise-addr 192.168.99.100"
```

   The first machine, myvm1, acts as the manager, which executes management commands and authenticates workers to join the swarm, and the second machine functions as a worker.

4. Add the remaining machines to the configuration as workers.

```
# docker-machine ssh myvm2 "docker swarm join --token SWMTKN-1-
3vcz1rkswq78s7t5sor3hrlbmzda4z523g8rnwkb8m8nd7tnpt-9uk7csvuieqqdg4b85nkk5ty9
192.168.99.100:2377"
# docker-machine ssh myvm3 "docker swarm join --token SWMTKN-1-
3vcz1rkswq78s7t5sor3hrlbmzda4z523g8rnwkb8m8nd7tnpt-9uk7csvuieqqdg4b85nkk5ty9
192.168.99.100:2377"
```

5. Execute docker node ls on the manager, myvm1, to view the nodes in the swarm.

```
# docker-machine ssh myvm1 "docker node ls"
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER
STATUS      ENGINE VERSION
qsanp3vxs2mtccv9wk8fxdwur *   myvm1               Ready               Active              Leader
18.06.1-ce
dac4sgcob2i4djab0vp74pay5     myvm2               Ready               Active
18.06.1-ce
73gsgheap7x7c3n35de9nn0mb     myvm3               Ready               Active
18.06.1-ce
```

## Deploy application on Swarm Manager

Execute the following on the Manager Node to configure the Luna Cloud HSM service for your swarm configuration. To deploy application on swarm manager:

1. Copy all the secret files to the swarm manager.

```
# docker-machine scp -r -d secrets/ myvm1:/home/docker/
```

2. SSH to the manager myvm1.

```
# docker-machine ssh myvm1
```

3. Create a local copy of docker-compose.yml on the manager:

```
version: '3.1'
services:
test:
image: <username>/docker-swarm:latest
# command: 'cat /run/secrets/luna_secret '
stdin_open: true
tty: true
secrets:
- source: chrystoki-conf
target: /usr/local/luna/Chrystoki.conf
- source: partition-ca-certificate
target: /usr/local/luna/partition-ca-certificate.pem
- source: partition-certificate
target: /usr/local/luna/partition-certificate.pem
- source: server-certificate
target: /usr/local/luna/server-certificate.pem
deploy:
replicas: 5
resources:
limits:
cpus: "0.1"
memory: 50M

secrets:
chrystoki-conf:
file: ./Chrystoki.conf
partition-ca-certificate:
file: ./partition-ca-certificate.pem
```

```
partition-certificate:
file: ./partition-certificate.pem
server-certificate:
file: ./server-certificate.pem
```

4. Change the path in the Chrystoki.conf file on the Manager node, so that it points to the secrets:

```
$ sed -i  's#\./#/usr/local/luna/#g' Chrystoki.conf
```

5. Deploy the service.

```
$ docker stack deploy -c docker-compose.yml latest
```

6. Run Docker.

```
$ docker ps -a
CONTAINER ID        IMAGE                                COMMAND             CREATED
STATUS              PORTS               NAMES
66c57a9aa7da        deegupta1302/docker-swarm:latest   "/bin/bash"         About a minute ago   Up
About a minute                      latest_test.1.kot01ixg1oe8he3cixodk4hv7
```

7. Access the Docker container.

```
$ docker exec -it latest_test.1.kot01ixg1oe8he3cixodk4hv7 /bin/bash
```

Now you are inside container.

8. Access LunaCM from the Docker container.

```
# cd /usr/local/luna/
# ./bin/64/lunacm
LunaCM v1.0.0-638. Copyright (c) 2006-2017 SafeNet.


   Available HSMs:


   Slot Id ->              3
   Label ->                dockerswarm
   Serial Number ->        1285255181019
   Model ->                Luna K7
   Firmware Version ->     7.1.1
   Configuration ->        Luna User Partition With SO (PW) Signing With
Cloning Mode
   Slot Description ->     User Token Slot
   Current Slot Id: 3
```

9. SSH to the worker node myvm2.

```
# docker-machine ssh myvm2
```

10. Run the Docker image on worker node myvm2.

```
$ docker ps -a
CONTAINER ID        IMAGE                                COMMAND             CREATED             STATUS
PORTS               NAMES
4c2912fec394        deegupta1302/docker-swarm:latest   "/bin/bash"         3 minutes ago       Up 2
minutes                         latest_test.4.15m3zn8a8606r9zbfmnf3qypb
fd92e2aab65a        deegupta1302/docker-swarm:latest   "/bin/bash"         3 minutes ago       Up 2
minutes                         latest_test.2.u3ion33z3sjuwafg85hnawoej
```

11. Access the worker node myvm2.

```
$ docker exec -it latest_test.4.15m3zn8a8606r9zbfmnf3qypb /bin/bash
```

Now you are inside container

12. Access LunaCM from the worker node myvm2.

```
# cd /usr/local/luna/
root@4c2912fec394:/usr/local/luna# ./bin/64/lunacm
LunaCM v1.0.0-638. Copyright (c) 2006-2017 SafeNet.
   Available HSMs:

   Slot Id ->              3
   Label ->                dockerswarm
   Serial Number ->        1285255181019
   Model ->                Luna K7
   Firmware Version ->     7.1.1
   Configuration ->        Luna User Partition With SO (PW) Signing With
Cloning Mode
   Slot Description ->     User Token Slot
   Current Slot Id: 3
```

13. SSH to the worker node myvm3.

```
# docker-machine ssh myvm3
```

14. Run the Docker image on worker node myvm3.

```
# docker ps -a
CONTAINER ID        IMAGE                                COMMAND             CREATED             STATUS
PORTS               NAMES
eff7be65c9ec        deegupta1302/docker-swarm:latest   "/bin/bash"         4 minutes ago       Up 4
minutes                         latest_test.3.j9ylpdoiiza71ijdk7y50xfnz
13383a82145a        deegupta1302/docker-swarm:latest   "/bin/bash"         4 minutes ago       Up 4
minutes                         latest_test.5.o986y3yj840copcv58qhtjgrw
```

15. Access the worker node myvm3.

```
# docker exec -it latest_test.3.j9ylpdoiiza71ijdk7y50xfnz /bin/bash
```

Now you are inside container

16. Access LunaCM from the worker node myvm3.

```
# cd /usr/local/luna/
# ./bin/64/lunacm
LunaCM v1.0.0-638. Copyright (c) 2006-2017 SafeNet.
   Available HSMs:

   Slot Id ->              3
   Label ->                dockerswarm
   Serial Number ->        1285255181019
   Model ->                Luna K7
   Firmware Version ->     7.1.1
   Configuration ->        Luna User Partition With SO (PW) Signing With
Cloning Mode
   Slot Description ->     User Token Slot
   Current Slot Id: 3
```

See Using Luna Cloud HSM service client inside Docker Container for an application demonstration inside of a Docker using Luna Cloud HSM.

# Integrating Kubernetes with Luna Network HSM

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. This section demonstrates integrating Kubernetes with a Luna HSM. Luna Network HSM provides strong physical protection of secure assets, including keys, and should be considered a best practice when working with Kubernetes.

> **NOTE**: This integration assumes that a Kubernetes cluster is up and running, and that at least the Master and a single Node in the cluster exists. You can verify the state of your Kubernetes cluster by executing “kubectl get nodes”.

## Provision Luna Network HSM

To provision your Luna Network HSM:

HSM is setup, initialized, provisioned, and ready for deployment. Refer to the Luna HSM Product Documentation for further details.

1. Create a partition on the HSM for use by Kubernetes.
2. Register the client for the Kubernetes Master and assign the client to a partition to create an NTLS connection.

3. Disable the IP checking for NTLS by executing the following on the Luna console:

```
lunash:> ntls ipcheck disable
NTLS client source IP validation disabled
Command Result: 0 (Success)
```

4. Initialize the Crypto Officer and Crypto User roles for the partition.
5. Verify that the partition is successfully registered and configured on Kubernetes Master.

```
# cd /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v7.2.0-220. Copyright (c) 2018 SafeNet. All rights reserved.
    Available HSMs:

    Slot Id ->              0
    Label ->                Kubernetes_CLS
    Serial Number ->        1238712343066
    Model ->                LunaSA 7.2.0
    Firmware Version ->     7.2.0
    Configuration ->        Luna User Partition With SO (PED) Key Export With
Cloning Mode
    Slot Description ->     Net Token Slot
    Current Slot Id: 0
```

## Configure and install Luna Minimal Client in Kubernetes

Configure and install the Luna minimal client in Kubernetes to create a Pod communicating with the Luna HSM partition over NTLS.

> **NOTE**: Please refer the APPENDIX C: Using Luna Client from Host to Kubernetes Pods if you want to use Luna Client configured on the host system via Kubernetes volumes without installing any client inside the pod.

Complete the following procedures on the Kubernetes Master. Any configuration updates to the Kubernetes Master will automatically deploy on any Nodes connected to the Master.

> Create the Luna Client Image in Docker Registry

> Create the Kubernetes Secrets

> Deploy a Pod using the Luna Client Image and Kubernetes Secret

### Create the Luna Client Image in Docker Registry

Create a Docker image containing the minimal required packages and utilities for communicating with the Luna HSM. To create the Luna client image in Docker registry:

1. Copy the Luna minimal client package on the Kubernetes Master.

2. Create the file Dockerfile in the directory where Luna Minimal Client package is copied with the following contents.

```
# cat Dockerfile
FROM centos:centos7
COPY LunaClient-Minimal-7.x.x.x86_64.tar /tmp
RUN mkdir -p /usr/safenet/lunaclient
RUN mkdir -p /usr/safenet/lunaclient/bin
RUN mkdir -p /usr/safenet/lunaclient/certs
RUN mkdir -p /usr/safenet/lunaclient/certs/client
RUN mkdir -p /usr/safenet/lunaclient/certs/server
RUN tar -xvf /tmp/LunaClient-Minimal-7.x.x.x86_64.tar --strip 1 -C
/usr/safenet/lunaclient
ENV ChrystokiConfigurationPath=/etc
COPY lunacm /usr/safenet/lunaclient/bin
COPY vtl /usr/safenet/lunaclient/bin
COPY openssl.cnf /usr/safenet/lunaclient/bin
ENTRYPOINT /bin/bash
```

3. Create a Docker build using the new Dockerfile.

```
# docker build . -t lunaclient
```

4. Verify that the image is created.

```
# docker images
```

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|---|---|---|---|---|
| lunaclient | latest | 13b904fbddc2 | 4 days ago | 240MB |
| centos | centos7 | 75835a67d134 | 6 days ago | 200MB |
| k8s.gcr.io/kube-apiserver | v1.12.0 | ab60b017e34f | 2 weeks ago | 194MB |
| k8s.gcr.io/kube-controller-manager | v1.12.0 | 07e068033cf2 | 2 weeks ago | 164MB |
| k8s.gcr.io/kube-scheduler | v1.12.0 | 5a1527e735da | 2 weeks ago | 8.3MB |
| k8s.gcr.io/kube-proxy | v1.12.0 | 9c3a9d3f09a0 | 2 weeks ago | 6.6MB |
| k8s.gcr.io/etcd | 3.2.24 | 3cab8e1b9802 | 3 weeks ago | 220MB |
| k8s.gcr.io/coredns | 1.2.2 | 367cdc8433a4 | 6 weeks ago | 9.2MB |
| quay.io/coreos/flannel | v0.10.0-amd64 | f0fad859c909 | 8 months ago | 44MB |
| k8s.gcr.io/pause | 3.1 | da86e6ba6ca1 | 9 months ago | 742kB |

5. Log in to Docker Hub.

```
# docker login
```

6. Tag the lunaclient build using the command below. Replace the <username> with your Docker hub username.

```
# docker tag lunaclient <username>/lunaclient
```

7. Push the lunaclient image to Docker hub.

```
# docker push <username>/lunaclient
```

### Create the Kubernetes Secrets

Kubernetes secrets are used to pass sensitive information at run time without exposing them publicly. In this step, we will create Kubernetes secret for client/server certificates and configuration file containing server and client information. To create Kubernetes secrets:

1. Create a server certificate secret and a CA certificate secret. In the following command replace the <server IP> with the actual HSM IP.

```
# kubectl create secret generic server-auth --from-
file=/usr/safenet/lunaclient/cert/server/<server IP>Cert.pem --from-
file=/usr/safenet/lunaclient/cert/server/CAFile.pem
```

2. Create a client certificate secret and a client private key secret. In the following command replace the <hostname> with the actual client hostname where Kubernetes Master is running.

```
# kubectl create secret generic client-auth --from-
file=/usr/safenet/lunaclient/cert/client/<hostname>.pem --from-
file=/usr/safenet/lunaclient/cert/client/<hostname>Key.pem
```

3. Edit the following sections of the /etc/Chrystoki.conf file to use the Luna Minimal Client.

> **NOTE**: Do not change any other section of the Chrystoki.conf file.

```
Chrystoki2 = {
   LibUNIX = /usr/safenet/lunaclient/libs/64/libCryptoki2.so;
   LibUNIX64 = /usr/safenet/lunaclient/libs/64/libCryptoki2.so;
}
Secure Trusted Channel = {
   ClientTokenLib = /usr/safenet/lunaclient/libs/64/libSoftToken.so;
}
```

4. Create a configuration file secret.

```
# kubectl create secret generic chrystoki-conf --from-file=/etc/Chrystoki.conf
```

5. Verify that the secrets exist. You should have a server certificate secret, a CA certificate secret, a client certificate secret, a client private key secret, and a configuration file secret.

```
# kubectl get secrets
```

| NAME | TYPE | DATA | AGE |
|---|---|---|---|
| chrystoki-conf | Opaque | 1 | 4d15h |
| client-auth | Opaque | 2 | 4d15h |
| default-token-8wtbg | kubernetes.io/service-account-token | 3 | 10d |
| server-auth | Opaque | 2 | 4d15h |

### Deploy a Pod using the Luna Client Image and Kubernetes Secret

Deploy a Pod on Kubernetes using the Luna Client Image that was pushed to the Docker Registry and a Kubernetes secret. At the end of deployment, the Pod will start running on all Nodes with an NTLS connection to the HSM partition. To deploy a pod using the Luna Client Image and Kubernetes secret:

1. Create a yaml file for Pod deployment e.g. secret-volume.yaml. Add the following entries to the secret-volume.yaml. Replace <username> with your Docker Hub username:

```
# cat secret-volume.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-lunaclient
spec:
  containers:
  - name: lunaclient
    image: <username>/lunaclient
    # Just spin & wait forever
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
    volumeMounts:
    - name: myconf
      mountPath: /etc
      mountPath: /etc/Chrystoki.conf
      subPath: Chrystoki.conf
      readOnly: true
    - name: myserver
      mountPath: /usr/safenet/lunaclient/cert/server
      readOnly: true
    - name: myclient
      mountPath: /usr/safenet/lunaclient/cert/client
      readOnly: true
  volumes:
  - name: myconf
    secret:
      secretName: chrystoki-conf
  - name: myserver
    secret:
      secretName: server-auth
```

```
  - name: myclient
    secret:
      secretName: client-auth
```

2. Create a pod deployment using the kubectl command and yaml file created above.

   ```
   # kubectl create -f secret-volume.yaml
   ```

   This may take a few minutes.

3. Verify the deployment status.

   ```
   # kubectl get pods
   NAME                 READY   STATUS    RESTARTS   AGE
   pod-with-lunaclient  1/1     Running   0          4d2h
   ```

4. When STATUS is RUNNING, you can connect the pod to verify the NTLS connection. Execute the following command on the Master or any Node connected to the Master:

   ```
   # kubectl exec -it pod-with-lunaclient -- /bin/bash
   [root@pod-with-lunaclient /]#
   ```

5. Verify that the pod can access the HSM partition.

   ```
   [root@pod-with-lunaclient /]# /usr/safenet/lunaclient/bin/lunacm
   lunacm (64-bit) v7.2.0-220. Copyright (c) 2018 SafeNet. All rights reserved.


       Available HSMs:


       Slot Id ->              0
       Label ->                Kubernetes_CLS
       Serial Number ->        1238712343066
       Model ->                LunaSA 7.1.0
       Firmware Version ->     7.1.0
       Configuration ->        Luna User Partition With SO (PED) Key Export With
   Cloning Mode
       Slot Description ->     Net Token Slot


       Current Slot Id: 0
   ```

   This completes the integration of Kubernetes with a Luna HSM. To verify the integration with the Luna HSM, run any application in the Pod that uses the HSM services. See Using the Luna HSM inside Docker Container for Java Code Signing demonstration inside of a Docker.

# Integrating OpenShift Origin with Luna Cloud HSM

OpenShift is a container application platform for Docker and Kubernetes. Luna Cloud HSM service provides strong physical protection of secure assets, including keys, and should be considered a best practice when working with OpenShift Origin.

> **NOTE**: This integration assumes that an OpenShift Origin Cluster with a configured registry, router, image streams, and default templates is deployed and operating on the host system.

Following are the steps involved in using Luna Cloud HSM service with OpenShift Origin:

> Provision Luna Cloud HSM service

> **Deploy OpenShift Origin pod**

## Provision Luna Cloud HSM service

This service enables your client machine to access an HSM application partition for storing cryptographic objects. Application partitions can be assigned to a single client or multiple clients can share a single application partition. To provision Luna Cloud HSM service:

1. Log in to Luna Cloud HSM as an Application Owner user.
2. Under the **Services** tab, select the **Add New Service** heading.
3. Click **Deploy** on the HSM on Demand tile. The service wizard will appear on your screen.
4. Review the terms of services, accept these terms by checking the checkbox, and then click **Next**.
5. On the **Add HSM on Demand** service page, provide a **Service Name** (e.g. `fordocker`)
6. Click the service name. The **Create Service Client** window will appear on your screen.
7. In the Create Service Client window, enter a Service Client Name (e.g. `fordocker_client`) and select **Create Service Client**. A new HSM service client package (in this case, `ForDocker_client.zip`) gets generated and is ready to be downloaded and installed on your client machine.
8. Transfer the client package to your host machine. You can use SCP, PSCP, WinSCP, FTPS, or some other secure transfer tool to transfer the client package.

> **NOTE:** Refer to the section HSM On Demand Services in the Luna Cloud HSM Application Owner Guide for detailed information.

## Deploy OpenShift Origin pod

Configure Luna Cloud HSM to function in an OpenShift Origin pod. A pod is one or more containers deployed together on one host, and the smallest compute unit that can be defined, deployed, and managed. Each pod is allocated its own internal IP address, therefore owning its entire port space, and containers within pods can share their local storage and networking. You can deploy Luna Cloud HSM service inside an OpenShift Origin pod using one of the following methods:

> Deploy OpenShift Origin pod using persistent volume (Method I)

> Deploy OpenShift Origin pod using task file (Method II)

### Deploy OpenShift Origin pod using persistent volume (Method I)

Containers in Openshift don't persist data. Every time you start an application, it is started in a new container with an immutable Docker image. Any persisted data in the file systems is lost when the container stops. As a result, if a container is rebuilt or restarted, you cannot view previous data. We recommend using Persistent Volume. You can share this Persistent Volume with multiple pods at a time. Following are the steps involved in deploying OpenShift Origin using persistent volume:

> Create Luna Docker image
> Configure Luna Cloud HSM service inside OpenShift Origin
> Configure OpenShift Origin pod to run with root privileges
> Add persistent volume to the pod
> Create persistent volume using the Web interface
> Copy Secrets to Persistent Volume
> Configure Luna Cloud HSM inside a pod

## Create Luna Docker Image

To use a Luna Cloud HSM service with OpenShift Origin, you must create and run the Luna Docker image. Create the Docker file and extract the Luna Cloud HSM service inside the Docker container. To create the Luna Docker image:

1. Unzip the downloaded client package and store the files in a directory named clientfiles excluding certificates and configuration file which have server and client information.

```
# ls clientfiles/
bin  etc  EULA.zip  jsp  libs  setenv
```

2. Store the certificates and configuration file in separate directory named secrets.

```
# ls secrets
Chrystoki.conf partition-ca-certificate.pem partition-certificate.pem server-certificate.pem
```

3. Create a file named Dockerfile in the current working directory and add the following information to this file:

```
FROM centos:centos7
RUN mkdir -p /usr/local/luna
COPY clientfiles /usr/local/luna
ENV ChrystokiConfigurationPath=/usr/local/luna/secrets
CMD ["sh", "-c", "tail -f /dev/null"]
#End of the Dockerfile
```

4. Build the Docker image using the Dockerfile.

```
# docker build . -t dpod-image
```

5. Verify that the Docker image was created.

```
# docker images
```

6. Log in to Docker Registry. Provide username and password for Docker Registry when prompted.

   ```
   # docker login
   ```

7. Tag the lunaclient build using the following command. Replace the <username> with your Docker registry username.

   ```
   # docker tag dpod-image <username>/dpod
   ```

8. Push the image to the docker hub.

   ```
   # docker push <username>/dpod
   ```

## Configure Luna Cloud HSM service inside OpenShift Origin

Configure Luna Cloud HSM service inside of OpenShift Origin for use with OpenShift Origin. To configure Luna Cloud HSM service inside OpenShift Origin:

1. Create a project in OpenShift.

   ```
   # oc new-project mylunaproject
   ```

2. Create an app within the project.

   ```
   # oc new-app --docker-image=<username>/dpod --name=mylunaapp
   ```

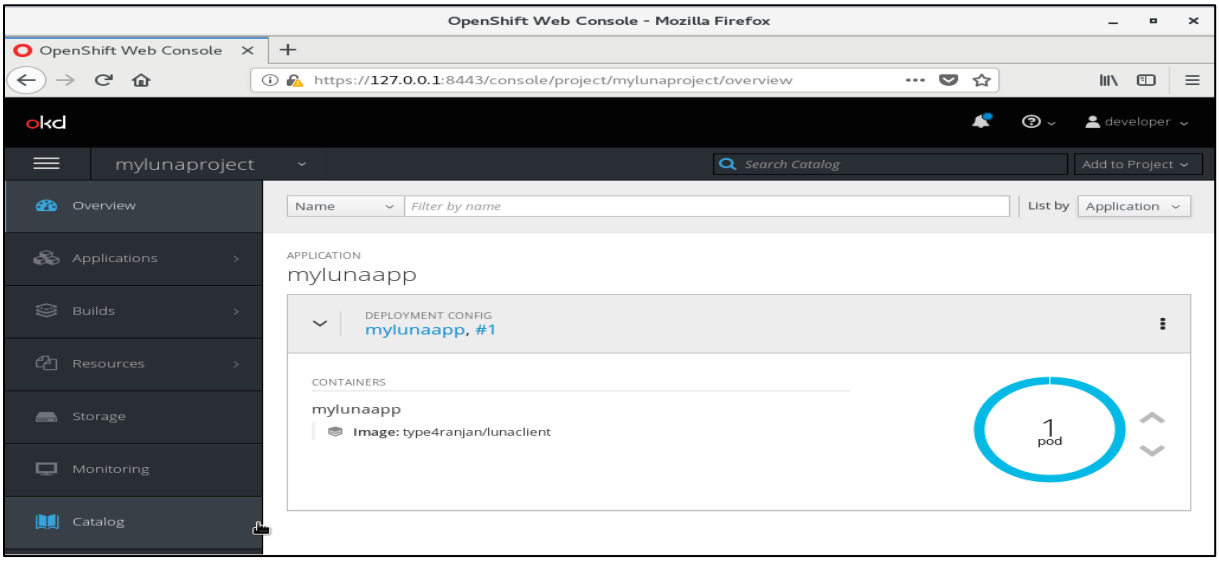
   The application will automatically deploy on a pod.

3. List all the pods and their status.

   ```
   # oc get pods
   ```

   You will see output similar to the following:


4. Verify the pod on the OpenShift web console:



## Configure OpenShift Origin pod to run with root privileges

On the initial login to the pod console, the default user in non-root. To configure OpenShift Origin pod to run with root privileges:

1. Create a service account and associate it with the Luna Cloud HSM project.

```
# oc login -u system:admin

# oc create serviceaccount useroot

# oc adm policy add-scc-to-user anyuid -z useroot -n mylunaproject
```

2. Apply the patch to the application:

```
# oc patch dc/mylunaapp --patch
'{"spec":{"template":{"spec":{"serviceAccountName": "useroot"}}}}'
```

   This applies the patch to all Pods. You can now run the Pods with root privileges.

## Add persistent volume to the pod

Persistent volume is used to share the certificates and configuration files from local to all the pods. To create persistent volume over the command line interface (CLI), run the following command to create a persistent storage and mount it to /usr/local/luna/secrets:

```
# oc set volume dc/mylunaapp --add --name=tmp-mount --claim-name=mylunastorage
--claim-mode="ReadWriteMany" --type pvc --claim-size=1G --mount-path
/usr/local/luna/secrets
```

## Create persistent volume using the Web interface

1. Log in to the OpenShift Origin web portal.
2. Go to the **storage** section of mylunaproject.
3. Click **Create Storage**.
4. Provide the following field values:

```
Name=mylunastorage

Access Mode=Shared Access(RWX)

Size=1GiB
```

5. Click **Create**. Persistent storage generates.
6. Navigate to the application mylunaapp and click **Add storage** to **mylunaapp**.
7. Select **Storage** as mylunastorage.
8. Provide following fields values:

```
Mount Path=/usr/local/luna/secrets
```

   Leave volume and Subpath name blank.

   For this deployment config, do not select "read only" and "pause rollout" options.
9. Click on **Add**. All the pods will automatically restart.

## Copy Secrets to Persistent Volume

You need to copy the secrets directory to the persistent volume added to the Pods so that it has access to the certificates and configuration file need to run the Luna Cloud HSM service. To copy secrets to persistent volume:

1. Make changes to chrystoki.conf file before copying it to the storage:

   ```
   # sed -i -e 's#\./#/usr/local/luna/#g' Chrystoki.conf
   ```

   ```
   # sed -i -e  's#partition-ca-certificate.pem#secrets/partition-ca-certificate.pem#g' -e 's#partition-certificate.pem#secrets/partition-certificate.pem#g' -e  's#server-certificate.pem#secrets/server-certificate.pem#g' Chrystoki.conf
   ```

2. Get the running pod name with following command.

   ```
   # oc get pods
   ```

3. Select any latest running pod name for example mylunaapp-1-qlfc4. Copy the secrets with following command:

   ```
   # oc rsync /root/secrets mylunaapp-1-v6jh9:/usr/local/luna/
   ```

   As the persistent storage was already mounted on /usr/local/luna/secrets, so the secrets will copied to the persistent storage and will be available to all the pods.

## Configure Luna Cloud HSM inside a pod

To configure Luna Cloud HSM service inside a pod:

1. Open the terminal.

   ```
   # oc rsh mylunaapp-1-v6jh9
   ```

2. Run `lunacm` and verify the connection to the partition:

   ```
   # cd /usr/local/luna/bin/64/
   ```

   ```
   # ./lunacm
   ```

3. Initialize the application partition, to create the partition's Security Officer (SO), and set the initial password and cloning domain.

   ```
   lunacm:> partition init -label <par_label>
   ```

4. Log in as Partition SO. You can also use the shortcut `po`.

   ```
   lunacm:> role login -name Partition SO
   ```

5. Initialize the Crypto Officer role and set the initial password. You can also use the shortcut `co`.

   ```
   lunacm:> role init -name Crypto Officer
   ```

6. The Partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. You must log out to allow the Crypto Officer to log in with the newly-set password.

   ```
   lunacm:> role logout
   ```

> **NOTE**: Once the Crypto Officer logs in and changes the initial credential set by the Partition SO, applications using the Crypto Officer challenge secret/password can perform cryptographic operations in the partition. The Crypto Officer can create, modify and delete crypto objects within the partition, and use existing crypto objects (sign/verify). You can also create a limited-capability role called Crypto User that can use the objects created by the Crypto Officer, but cannot modify them. The separation of roles is important in some security regimes and operational situations, and where you might be required to satisfy audit criteria for industry or government oversight.

7. Log in as the Crypto Officer. You can also use the shortcut `co`.

```
lunacm:> role login -name Crypto Officer
```

> **NOTE**: The password for the Crypto Officer role is valid for the initial login only. You must change the initial password using the command role changepw during the initial login session, or a subsequent login. Failing to change the password will result in a CKR_PIN_EXPIRED error when you perform role-dependent actions.

8. Change the initial password set by the Partition SO if you have not done so already.

```
lunacm:> role changepw -name Crypto Officer
```

9. Create the Crypto User. You can also use the shortcut `cu`.

```
lunacm:> role init -name Crypto User
```

The Crypto User can now log in with the credentials provided by the Crypto Officer and change the initial password. The Crypto User can now use applications to perform cryptographic operations using keys and objects created in the partition by the Crypto Officer.

10. You can scale up or down for the number of pods you want. To scale up or down, use the following command:

```
# oc scale dc mylunaapp --replicas=3
```

```
# oc get pods
```

```
NAME                READY     STATUS    RESTARTS   AGE
mylunaapp-1-v6jh9   1/1       Running   0          5m
mylunaapp-1-qnt5r   1/1       Running   0          18s
mylunaapp-1-rtn4f   1/1       Running   0          18s
```

This completes the integration of OpenShift Origin with Luna Cloud HSM. To verify the integration with the Luna Cloud HSM service, run any application in the Pod that uses the HSM services.

See Using Luna Cloud HSM service client inside Docker Container for Java Code Signing demonstration inside of OpenShift Pod using Luna Cloud HSM.

### Deploy OpenShift Origin pod using task file (Method II)

Containers in OpenShift Origin can be deployed and configured using a task file. Compile the task file and deploy the OpenShift Origin pods.

## Create the Luna Docker Image

To use a Luna Cloud HSM service with OpenShift Origin you must create and run the Luna Docker image. Create the Docker file and extract the Luna Cloud HSM service inside of the Docker container. To create Luna Docker image:

1. Unzip the downloaded client package and store the files in a directory named clientfiles excluding certificates and configuration file which have server and client information.

   ```
   # ls clientfiles/
   bin  etc  EULA.zip  jsp  libs  setenv
   ```

2. Store the certificates and configuration file in separate directory named secrets.

   ```
   # ls secrets
   Chrystoki.conf partition-ca-certificate.pem partition-certificate.pem server-certificate.pem
   ```

3. Create a file named Dockerfile in the current working directory and add the following information to this file:

   ```
   FROM centos:centos7
   RUN mkdir -p /usr/local/luna
   COPY clientfiles /usr/local/luna
   ENV ChrystokiConfigurationPath=/usr/local/luna/secrets
   ENTRYPOINT /bin/bash
   #End of the Dockerfile
   ```

4. Build the Docker image using the new Dockerfile.

   ```
   # docker build . -t dpod-image
   ```

5. Verify the Docker image was created.

   ```
   # docker images
   ```

6. Log in to Docker registry. Provide username and password for Docker registry when prompted.

   ```
   # docker login
   ```

7. Tag the lunaclient build using the following command. Replace the <username> with your Docker registry username.

   ```
   # docker tag dpod-image <username>/dpod
   ```

8. Push the image to Docker hub.

   ```
   # docker push <username>/dpod
   ```

## Configure Luna Cloud HSM service inside OpenShift Origin

Configure Luna Cloud HSM service inside of OpenShift Origin for use with OpenShift Origin. To configure the Luna Cloud HSM service inside OpenShift Origin:

1. Create a project in OpenShift:

```
# oc new-project mylunaproject
```

2. Update the Chrystoki.conf file in secrets directory using the following command:

```
# sed -i -e 's#\./#/usr/local/luna/#g' Chrystoki.conf
```

```
# sed -i -e  's#partition-ca-certificate.pem#secrets/partition-ca-certificate.pem#g' -e 's#partition-certificate.pem#secrets/partition-certificate.pem#g' -e  's#server-certificate.pem#secrets/server-certificate.pem#g' Chrystoki.conf
```

3. Create the a generic secret with the following command:

```
# oc create secret generic mysecrets --from-file=/root/secrets/Chrystoki.conf --from-file=/root/secrets/partition-ca-certificate.pem --from-file=/root/secrets/partition-certificate.pem --from-file=/root/secrets/server-certificate.pem
```

4. Verify the secrets:

```
# oc get secrets
```

| NAME | TYPE | DATA | AGE |
|---|---|---|---|
| builder-dockercfg-htkjj | kubernetes.io/dockercfg | 1 | 4m |
| builder-token-2llws | kubernetes.io/service-account-token | 4 | 4m |
| builder-token-ntjmd | kubernetes.io/service-account-token | 4 | 4m |
| default-dockercfg-zxs9c | kubernetes.io/dockercfg | 1 | 4m |
| default-token-g9bpf | kubernetes.io/service-account-token | 4 | 4m |
| default-token-hk45v | kubernetes.io/service-account-token | 4 | 4m |
| deployer-dockercfg-2pgbz | kubernetes.io/dockercfg | 1 | 4m |
| deployer-token-46gf8 | kubernetes.io/service-account-token | 4 | 4m |
| deployer-token-pfkzw | kubernetes.io/service-account-token | 4 | 4m |
| mysecrets | Opaque | 4 | 3m |

5. Create a configuration file deploypod.yaml and add the following information to it:

```
apiVersion: v1
kind: Pod
metadata:
  name: mylunaapp-pod
spec:
  containers:
 - image: 'namespace/dpod'
    # Just spin & wait forever
    name: mylunaapp
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
    volumeMounts:
```

```
    - name: lunasecret
      mountPath: /usr/local/luna/secrets
      readOnly: true
  volumes:
  - name: lunasecret
    secret:
      secretName: mysecrets
```

6. Deploy the application using the new deployment file.

```
# oc create -f deploypod.yaml
```

7. List all pods and their status.

```
# oc get pods
NAME           READY     STATUS    RESTARTS   AGE
mylunaapp-pod  1/1       Running   0          45s
```

## Configure pod to run with root privileges

On the initial login to the pod console, the default user in non-root. Complete the following procedure to enable root permissions, allowing the user to execute luna client utilities. To configure pod to run with root privileges:

1. Create a service account and associate it with the project.

```
# oc login -u system:admin
# oc create serviceaccount useroot
# oc adm policy add-scc-to-user anyuid -z useroot -n mylunaproject
```

2. Apply the patch to the application.

```
# oc patch dc/mylunaapp --patch
'{"spec":{"template":{"spec":{"serviceAccountName": "useroot"}}}}'
```

   This applies the patch to all Pods. You can now run the Pods with root privileges.

## Configure Luna Cloud HSM service inside Pods

Execute the following on the terminal of a Pod where you want to use the Luna Cloud HSM service. To configure the Luna Cloud HSM service inside pods:

1. Open the terminal:

```
# oc rsh mylunaapp-pod
```

2. Run lunacm and verify the connection to the partition.

```
# bin/64/lunacm
# ./lunacm
```

3. Initialize the application partition to create the partition's Security Officer (SO) and set the initial password and cloning domain.

```
lunacm:> partition init -label <par_label>
```

4. Log in as Partition SO. You can also use the shortcut `po`.

   ```
   lunacm:> role login -name Partition SO
   ```

5. Initialize the Crypto Officer role and set the initial password. You can also use the shortcut `co`.

   ```
   lunacm:> role init -name Crypto Officer
   ```

6. The Partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. You must log out to allow the Crypto Officer to log in with the newly set password.

   ```
   lunacm:> role logout
   ```

7. Log in as the Crypto Officer. You can also use the shortcut `co`.

   ```
   lunacm:> role login -name Crypto Officer
   ```

   **NOTE**: The password for the Crypto Officer role is valid for the initial login only. You must change the initial password using the command role changepw during the initial login session, or a subsequent login. Failing to change the password will result in a CKR_PIN_EXPIRED error when you perform role-dependent actions.

8. Change the initial password set by the Partition SO if you have not done so already.

   ```
   lunacm:> role changepw -name Crypto Officer
   ```

9. Create the Crypto User. You can also use the shortcut `cu`.

   ```
   lunacm:> role init -name Crypto User
   ```

   The Crypto User can now log in with the credentials provided by the Crypto Officer, and change the initial password. The Crypto User can now use applications to perform cryptographic operations using keys and objects created in the partition by the Crypto Officer.

10. You can scale up or down for the number of pods you want. To scale up or down, use the following command:

    ```
    # oc scale dc mylunaapp --replicas=3
    # oc get pods
    NAME                READY     STATUS    RESTARTS   AGE
    mylunaapp-1-v6jh9   1/1       Running   0          5m
    mylunaapp-1-qnt5r   1/1       Running   0          18s
    mylunaapp-1-rtn4f   1/1       Running   0          18s
    ```

    This completes the integration of OpenShift Origin with Luna Cloud HSM. To verify the integration with the Luna Cloud HSM service, run any application in the Pod that uses the HSM services.

    See Using Luna Cloud HSM service client inside Docker Container for an application demonstration inside of OpenShift Pod using Luna Cloud HSM.

# Integrating Apache Mesos with Luna Cloud HSM

Apache Mesos makes it easier to develop and manage fault-tolerant and scalable distributed applications. Mesos is a cluster manager aiming for improved resource utilization by dynamically sharing resources among multiple frameworks. Luna Cloud HSM service provides strong physical protection of secure assets, including keys, and should be considered a best practice when working with Apache Mesos. The steps involved in integrating Apache Mesos with Luna HSM are:

> Provision Luna Cloud HSM service for Apache Mesos
> Configure Luna Cloud HSM in Apache Mesos
> Create Luna Docker image
> Create a sample application in Marathon
> Start interactive session with Docker container

**NOTE**: This integration assumes that an Apache Mesos has been set up with an active Master node (elected using ZooKeeper) and at least one slave node.

## Provision Luna Cloud HSM service for Apache Mesos

This service enables your client machine to access an HSM application partition for storing cryptographic objects. Application partitions can be assigned to a single client, or multiple clients can share a single application partition. To provision Luna Cloud HSM service for Apache Mesos:

1. Log in to Luna Cloud HSM as an Application Owner user.
2. Under the **Services** tab, select the **Add New Service** heading.
3. Click **Deploy** on the HSM on Demand tile. The service wizard displays.
4. Review the terms of service, accept these terms by checking the checkbox, and then click **Next**.
5. On the **Add HSM on Demand** service page, provide a **Service Name** (for example, `fordocker`)
6. Click the service name. The **Create Service Client** window will appear on your screen.
7. In the Create Service Client window, enter a Service Client Name (for example, `fordocker_client`) and select **Create Service Client**. A new HSM service client package (in this case, `ForDocker_client.zip`) gets generated and is ready to be downloaded and installed on your client machine.
8. Transfer the client package to your host machine. You can use SCP, PSCP, WinSCP, FTPS, or some other secure transfer tool to transfer the client package.

**NOTE:** Refer to the section *HSM On Demand Services* in the *Luna Cloud HSM Application Owner Guide* for detailed information on configuring an HSM on Demand service.

## Configure Luna Cloud HSM in Apache Mesos

Create the Luna Docker image and upload the Luna Docker Image as a sample application operating within the Docker Container, and then open an interactive session with the Docker Container.

## Create Luna Docker image

To use Luna Cloud HSM service with Apache Mesos, you must create and run Luna Docker image. To create the Luna Docker image:

1. Unzip the downloaded client package and store the files in a directory named clientfiles, excluding certificates and configuration file which have server and client information.

```
# ls clientfiles/
bin  etc  EULA.zip  jsp  libs  setenv
```

2. Store the certificates and configuration file in separate directory named secrets.

```
# ls secrets
Chrystoki.conf partition-ca-certificate.pem partition-certificate.pem server-certificate.pem
```

3. Create the file Dockerfile in the current working directory and add the following information:

```
FROM centos:centos7
RUN mkdir -p /usr/local/luna
COPY clientfiles /usr/local/luna
ENV ChrystokiConfigurationPath=/usr/local/luna
CMD ["sh", "-c", "tail -f /dev/null"]
#End of the Dockerfile
```

4. Create a zip file named secrets.zip that contains all the files of directory secrets.
5. Build the Docker Image using the new Dockerfile.

```
# docker build . -t dpod-image
```

6. Verify that the Docker Image was created

```
# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
dpod-image          latest              8099de65ceb5        4 seconds ago       217MB
centos              centos7             75835a67d134        6 days ago          200MB
```

7. Log in to Docker Registry. Provide username and password for Docker Registry when prompted.

```
# docker login
```

8. Tag the lunaclient build using the command below. Replace the <username> with your docker hub username.

```
# docker tag dpod-image <username>/dpod
```

9. Push the image to the Docker Hub Repository.

```
# docker push <username>/dpod
```

## Create a sample application in Marathon

By default, Marathon runs on port 8080. Open the browser to public IP address and port 8080 to access its GUI. To create a sample Application in Marathon:

1. Click **Create Application** on the console.
2. Toggle to enable **JSON Mode** on the New Application Window.
3. Create sample application app.json to deploy. Add the following to the sample application:

```
{
  "id": "testapp",
  "cmd": null,
  "cpus": 1,
  "mem": 128,
  "disk": 1000,
  "instances": 1,
  "acceptedResourceRoles": [
    "*"
  ],
  "container": {
    "type": "DOCKER",
    "docker": {
      "forcePullImage": false,
      "image": "<username>/dpod",
      "parameters": [],
      "privileged": false
    }
  },
  "portDefinitions": [
    {
      "port": 10000,
      "name": "default",
      "protocol": "tcp"
    }
  ],
  "fetch": [
    {
      "uri": "file:///secrets/secrets.zip",
      "extract": true,
      "executable": false,
      "cache": false
    }
```

```
  ]
}
```

4. Click on Create Application. The Application is created under **Apps** in the console.
5. Wait for the application to go from **Deploying** to **Running** state.

> **NOTE**: You can also deploy application on mesos slave by creating app.json on master and use the HTTP API to deploy the app on Marathon ip-address by following command :
>
> ```
> curl -X POST http://<ip address>:8080/v2/apps -d @app.json -H
> "Content-type: application/json"
> ```

Switch to the Mesos console to see an Active task running.

## Start interactive session with Docker Container

Luna Cloud HSM service is now deployed in Docker Container. You can open the terminal inside the container to use the Luna Cloud HSM service. To start an interactive session with the running Docker Container:

1. Obtain the running container id.

   ```
   # docker ps –a
   ```

2. Start the interactive session of a running container using the container id.

   ```
   # docker attach <container id>
   ```

3. Copy the configuration file and certificates from /mnt/mesos/sandbox to directory /usr/local/luna.
4. Run lunacm and verify the connection to the partition.

   ```
   # bin/64/lunacm
   ```

   ```
   # ./lunacm
   ```

5. Initialize the application partition to create the partition's Security Officer (SO) and set the initial password and cloning domain.

   ```
   lunacm:> partition init -label <par_label>
   ```

6. Log in as Partition SO. You can also use the shortcut po.

   ```
   lunacm:> role login -name Partition SO
   ```

7. Initialize the Crypto Officer role and set the initial password. You can also use the shortcut co.

   ```
   lunacm:> role init -name Crypto Officer
   ```

8. The Partition SO can create the Crypto Officer, but only the Crypto Officer can create the Crypto User. You must log out to allow the Crypto Officer to log in with the newly-set password.

   ```
   lunacm:> role logout
   ```

> **NOTE**: Once the Crypto Officer logs in and changes the initial credential set by the Partition SO, applications using the CO's challenge secret/password can perform cryptographic operations in the partition. The Crypto Officer can create, modify, and delete crypto objects within the partition, and use existing crypto objects (sign/verify). You can also create a limited-capability role called Crypto User that can use the objects created by the Crypto Officer, but cannot modify them. The separation of roles is important in some security

regimes and operational situations, and where you might be required to satisfy audit criteria for industry or government oversight.

9. Log in as the Crypto Officer. You can also use the shortcut `co`.

```
lunacm:> role login -name Crypto Officer
```

**NOTE**: The password for the Crypto Officer role is valid only for the initial login. You must change the initial password using the command role changepw during the initial login session or a subsequent login. Failing to change the password will result in a CKR_PIN_EXPIRED error when you perform role-dependent actions.

10. Change the initial password set by the Partition SO, if you have not done so already.

```
lunacm:> role changepw -name Crypto Officer
```

11. Create the Crypto User. You can also use the shortcut `cu`.

```
lunacm:> role init -name Crypto User
```

The Crypto User can now log in with the credentials provided by the Crypto Officer, and change the initial password. The Crypto User can now use applications to perform cryptographic operations using keys and objects created in the partition by the Crypto Officer.

12. Scale up or down for the number of pods you want using the following command:

```
# oc scale dc mylunaapp --replicas=3
```

```
# oc get pods
```

```
NAME                READY     STATUS    RESTARTS   AGE
mylunaapp-1-v6jh9   1/1       Running   0          5m
mylunaapp-1-qnt5r   1/1       Running   0          18s
mylunaapp-1-rtn4f   1/1       Running   0          18s
```

This completes the integration of Apache Mesos with Luna Cloud HSM. To verify the integration with the Luna Cloud HSM service, run any application in the Pod that uses the HSM services.

See Using a Luna Cloud HSM service client inside Docker Container for Java Code Signing demonstration inside of Apache Mesos Pod using Luna Cloud HSM.

# APPENDIX A: Using Luna HSM inside Docker Container

This section demonstrates using Luna HSM and Luna Cloud HSM service inside Docker container. It demonstrates the method for using the Java keytool utility to generate signing keys and certificates on Luna HSM .It then demonstrates using the Java Code Signer to sign a JAR file inside of the Docker container. Install Java Development Kit (JDK) on the Docker container or Pod.

## Configure Java Keytool Utility to use Luna Cloud HSM service inside a Docker Container

To configure Java Keytool Utility to use Luna Cloud HSM service inside a Docker Container:

1. Edit the Java Security Configuration file java.security located in the security directory under <JDK Installation directory>/jre/lib/.

   ```
   Add the Luna Provider in java.security file as shown below:
     security.provider.1=sun.security.provider.Sun
     security.provider.2=sun.security.rsa.SunRsaSign
     security.provider.3=com.sun.net.ssl.internal.ssl.Provider
     security.provider.4=com.sun.crypto.provider.SunJCE
     security.provider.5=sun.security.jgss.SunProvider
     security.provider.6=com.sun.security.sasl.Provider
     security.provider.7=org.jcp.xml.dsig.internal.dom.XMLDSigRI
     security.provider.8=sun.security.smartcardio.SunPCSC
     security.provider.9=com.safenetinc.luna.provider.LunaProvider
     Save the changes in the java.security file.
   ```

2. Copy the LunaProvider.jar and libLunaAPI.so (UNIX) /LunaAPI.dll (Windows) from the <Luna Installation Directory>/jsp/lib folder to JAVA extension folder under <JDK Installation directory>/jre/lib/ext.

3. Set the environment variables for JAVA_HOME and PATH.

   ```
   # export JAVA_HOME=<JDK Installation directory>
   # export PATH=$JAVA_HOME/bin:$PATH
   ```

4. Create a blank file named lunastore and add the following entry where <Partition Name> would be your Luna HSM partition label:

   ```
   tokenlabel:<Partition Name>
   ```

   Save the file in current working directory.

## Generate a key pair and sign a JAR file inside a Docker Container

To generate a key pair and sign a JAR file inside a Docker Container:

1. Generate a key pair using Java keytool utility in the keystore which will generate the key pair in Luna HSM.

   ```
   # keytool -genkeypair -alias lunakey -keyalg RSA -sigalg SHA256withRSA -keypass
   userpin1 -keysize 2048 -keystore lunastore -storepass userpin1 -storetype luna

   What is your first and last name?
   ```

```
[Unknown]:  HSM
What is the name of your organizational unit?
[Unknown]:  HSM
What is the name of your organization?
[Unknown]:  Gemalto
What is the name of your City or Locality?
[Unknown]:  MyCity
What is the name of your State or Province?
[Unknown]:  MyState
What is the two-letter country code for this unit?
[Unknown]:  IN
Is CN=HSM, OU=HSM, O=Gemalto, L=MyCity, ST=MyState, C=IN correct?
[no]:  yes
```

A new key pair will be generated on registered Luna HSM partition.

> **NOTE**: The command above used “userpin1” as storepass which is the partition Crypto Officer Pin you set when initialized the CO role for the partition.

2. Verify that the private key is in the Luna HSM partition.

```
# keytool -list -v -storetype luna -keystore lunastore
```

The system prompt to enter the keystore password and after providing the password it display the contents.

```
Enter keystore password:

 Keystore type: LUNA
 Keystore provider: LunaProvider

 Your keystore contains 1 entry

Alias name: lunakey
Creation date: Apr 16, 2018
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=HSM, OU=HSM, O=Gemalto, L=MyCity, ST=MyState, C=IN
Issuer: CN=HSM, OU=HSM, O=Gemalto, L=MyCity, ST=MyState, C=IN
Serial number: 1353bc67
Valid from: Mon Apr 16 12:01:45 PDT 2018 until: Sun Jul 15 12:01:45 PDT 2018
```

```
Certificate fingerprints:
        MD5:  90:D9:4A:25:DD:C4:9E:7F:55:60:3D:ED:D0:84:18:C1
         SHA1: 01:FF:94:6B:24:3C:FB:5F:05:F9:7F:AC:3A:3B:4D:AB:0D:9A:69:36
         SHA256:
FD:09:09:3A:71:1C:69:A1:24:5E:78:AB:BB:7C:0C:D9:81:02:64:D2:AE:7C:A1:00:91:21:E
A:41:9E:3D:FA:0D
Signature algorithm name: SHA256withRSA
Version: 3


*******************************************
```

3. Generate a certificate request from a key in the keystore. When prompted for password, provide the keystore password.

   ```
   # keytool -certreq -alias lunakey -sigalg SHA256withRSA -file certreq_file -
   storetype luna -keystore lunastore
   ```

   Enter keystore password:

   File certreq_file will be generated in the current directory.

   > **NOTE**: After creating your CSR, make sure that you keep track of your keystore file because it contains your private key. In addition, you need the keystore file to install your Code Signing Certificate.

4. Copy the certificate request file generated to host machine to submit it to your Certification Authority (CA) by executing following command on host machine.

   ```
   # docker cp <container-id>:<certreq_file> .
   ```

   For Example:

   ```
   # docker cp d34fd7f4bc51:/usr/local/certreq_file .
   ```

5. The CA authenticates the request and returns a signed certificate or a certificate chain. Save the reply and the root certificate of the CA. Copy both certificates to Docker container.

   ```
   # docker cp <root-ca-cert> <container-id>:<directory to place cert>
   # docker cp <signed-cert-chain> <container-id>:<directory to place cert>
   ```

   For Example:

   ```
   # docker cp root.cer d34fd7f4bc51:/usr/local/
   # docker cp signing.p7b d34fd7f4bc51:/usr/local/
   ```

   **root.cer** and **signing.p7b** are the CA Root Certificate and Signed Certificate Chain respectively.

6. Import the CA Root certificate and signed certificate or certificate chain in to the keystore.

   To import the CA root certificate execute the following:

   ```
   # keytool -trustcacerts -importcert -alias rootca -file root.cer -keystore
   lunastore -storetype luna
   ```

   To import the signed certificate reply or certificate chain execute the following:

```
# keytool -trustcacerts -importcert -alias lunakey -file signing.p7b -keystore
lunastore -storetype luna
```

The keystore contents can also be verified by executing lunacm command: partition contents.

7. Copy the JAR file from host machine to docker container's current working directory. Execute the following command on host machine:

```
# docker cp <jar-to-be-signed> <container-id>:<directory to place jar file>
```

For Example:

```
# docker cp sample.jar d34fd7f4bc51:/usr/local/
```

8. Use jarsigner toll provided with Java to sign the jar file and provide the keystore password when it prompt. A message “jar signed” will display when the file is signed successfully.

```
# jarsigner -keystore lunastore -storetype luna -signedjar <name-of-signedjar-
to-be-generated>    <jar-to-be-signed> <alias-of-private-key> -tsa <time-
stamping-authority-url>
```

For Example:

```
# jarsigner -keystore lunastore -storetype luna -signedjar signedsample.jar
sample.jar lunakey -tsa http://timestamp.globalsign.com/scripts/timestamp.dll
```

Enter Passphrase for keystore:

```
jar signed.
```

9. Execute the following command to verify the signed jar. You will see a confirmation message at the end if the jar is verified.

```
# jarsigner -verify signedsample.jar -verbose –certs
s      565 Tue Apr 17 09:42:36 PDT 2018 META-INF/MANIFEST.MF
    [entry was signed on 4/16/18 9:12 PM]
    X.509, CN=Administrator, CN=Users, DC=CA, DC=com
    [certificate is valid from 4/16/18 12:02 PM to 4/16/19 12:02 PM]
    X.509, CN=my-CA, DC=CA, DC=com
    [certificate is valid from 4/5/18 10:25 AM to 4/5/23 10:35 AM]
    647 Tue Apr 17 09:42:36 PDT 2018 META-INF/LUNAKEY.SF
    5869 Tue Apr 17 09:42:36 PDT 2018 META-INF/LUNAKEY.RSA
    0 Thu Dec 02 10:41:40 PST 2010 META-INF/
m     506 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer$1.class
m     533 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer$2.class
m     1567 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer$3.class
m     3905 Mon May 21 00:09:04 PDT 2007 JSmoothPropertiesDisplayer.class
s = signature was verified
m = entry is listed in manifest
k = at least one certificate was found in keystore
i = at least one certificate was found in identity scope
```

```
    -Signed by "CN=Administrator, CN=Users, DC=CA, DC=com"
Digest algorithm: SHA256
Signature algorithm: SHA256withRSA, 2048-bit key
Timestamped by "CN=GlobalSign TSA for Standard - G2, O=GMO GlobalSign Pte Ltd,
C=SG" on Tue
Apr 17  04:12:52 UTC 2018
   Timestamp digest algorithm: SHA-256
   Timestamp signature algorithm: SHA1withRSA, 2048-bit key
jar verified.
```

This completes the demonstration of using the Java Keytool utility with a Luna HSM inside of a Docker Container. The JAR is signed and verified in the Docker Container while the private key and certificate are securely stored in the Luna HSM.

# APPENDIX B: Using Luna Client from Host to Docker Container

We can run the Luna Client installed on the host to Docker container using volumes without packaging and installing the Luna Client inside the Docker container. If you want to run Luna Client in multiple container without packaging the Luna Client with each container, volume mount can be an efficient way to utilize Luna HSM in container without installing the Luna Client inside the container.

1. Ensure that Luna client is installed and configured on the host system on default location.

```
[root@localhost home]# /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v10.4.0-417. Copyright (c) 2021 SafeNet. All rights reserved.


        Available HSMs:

        Slot Id ->              0
        Label ->                INTG_Par01
        Serial Number ->        1312109861420
        Model ->                LunaSA 7.7.1
        Firmware Version ->     7.7.1
        Bootloader Version ->   1.1.2
        Configuration ->        Luna User Partition With SO (PW) Key Export With Cloning Mode
        Slot Description ->     Net Token Slot
        FM HW Status ->         Non-FM


        Current Slot Id: 0


lunacm:>
```

2. Create the yaml file docker-compose.yaml and ensure that the file has the following contents:

> **Note:** This is just an example to run the NTL service in a container without installing the Luna Client in container. You can use this as a reference and use the Luna Client mounted in container with the actual application requires Luna HSM.

```
version: '3'
services:
  LunaInDocker:
    container_name: luna_container
    image: centos:latest
    volumes:
      - /etc/Chrystoki.conf:/etc/Chrystoki.conf
      - /usr/safenet/lunaclient:/usr/safenet/lunaclient
    environment:
      - ChrystokiConfigurationPath=/etc
      - LC_ALL=C
    tty: true
```

3. Create and start the container using docker-compose up command.

```
# docker-compose up &
```

```
[root@localhost home]# docker-compose up &
[1] 362778
Creating luna_container ... done
Attaching to luna_container

[root@localhost home]#
```

4. Connect to the container and run the lunacm to run the NTL connection in the container.

```
# docker exec -it luna_container bash
```

```
[root@localhost home]# docker exec -it luna_container bash
[root@3a196f715983 /]#
[root@3a196f715983 /]# /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v10.4.0-417. Copyright (c) 2021 SafeNet. All rights reserved.


       Available HSMs:

       Slot Id ->              0
       Label ->                INTG_Par01
       Serial Number ->        1312109861420
       Model ->                LunaSA 7.7.1
       Firmware Version ->     7.7.1
       Bootloader Version ->   1.1.2
       Configuration ->        Luna User Partition With SO (PW) Key Export With Cloning Mode
       Slot Description ->     Net Token Slot
       FM HW Status ->         Non-FM


       Current Slot Id: 0


lunacm:>
```

Refer to Using the Luna HSM inside Docker Container for Java Code Signing demonstration inside of a Docker using Luna Network HSM.

# APPENDIX C: Using Luna Client from Host to Kubernetes Pods

We can run the Luna Client installed on the Host to pods using volumes without packaging and installing the Luna Client inside the pods. If you want to run Luna Client in multiple pods without packaging the Luna Client with each pod, volume mount can be an efficient way to utilize Luna HSM in pods without installing the Luna Client inside the pods.

1. Ensure that Luna Client is installed and configured on the Host System on default location.

```
[root@k8s-node1 ~]# /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v10.3.0-275. Copyright (c) 2020 SafeNet. All rights reserved.


       Available HSMs:

       Slot Id ->              0
       Label ->                INTG_Par01
       Serial Number ->        1312109861420
       Model ->                LunaSA 7.7.1
       Firmware Version ->     7.7.1
       Bootloader Version ->   1.1.2
       Configuration ->        Luna User Partition With SO (PW) Key Export With Cloning Mode
       Slot Description ->     Net Token Slot
       FM HW Status ->         Non-FM

       Slot Id ->              7
       HSM Label ->            DKE-HA
       HSM Serial Number ->    11312109861420
       HSM Model ->            LunaVirtual
       HSM Firmware Version -> 7.7.1
       HSM Configuration ->    Luna Virtual HSM (PW) Key Export With Cloning Mode
       HSM Status ->           N/A - HA Group



       Current Slot Id: 0
```

2. Create the yaml file **lunaclient.yaml** and ensure that the file has the following information:

> **Note:** This is just an example to run the NTL service in a Kubernetes pod without installing the Luna Client inside the pod. You can use this as a reference and use the Luna Client mounted in pod with the actual application requires Luna HSM.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-lunaclient
spec:
  containers:
  - name: pod-with-lunaclient
    image: centos:latest
```

```
    # Just spin & wait forever
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
    volumeMounts:
    - name: lunaclient-config
      mountPath: /etc/Chrystoki.conf
      readOnly: true
    - name: lunaclient
      mountPath: /usr/safenet/lunaclient
    env:
    - name: LC_ALL
      value: C
  volumes:
  - name: lunaclient-config
    hostPath:
      path: /etc/Chrystoki.conf
      type: File
  - name: lunaclient
    hostPath:
      path: /usr/safenet/lunaclient
      type: Directory
```

3. Create a Pod deployment using the kubectl command and yaml file created above.

   ```
   # kubectl create -f lunaclient.yaml
   ```

   This may take a few minutes.

```
[root@k8s-master ~]# kubectl create -f lunaclient.yaml
pod/pod-with-lunaclient created
```

4. Verify the deployment status.

   ```
   # kubectl get pods
   ```

```
[root@k8s-master ~]# kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
pod-with-lunaclient   1/1     Running   0          25s
```

5. When STATUS is RUNNING, you can connect the Pod to verify the NTLS connection. Execute the following command on the Master or any Node connected to the Master.

```
# kubectl exec -it pod-with-lunaclient -- /bin/bash
```

```
[root@k8s-master ~]# kubectl exec -it pod-with-lunaclient -- /bin/bash
[root@pod-with-lunaclient /]#
```

6. Verify that the Pod can access the HSM partition.

```
[root@pod-with-lunaclient /]#
[root@pod-with-lunaclient /]# /usr/safenet/lunaclient/bin/lunacm
lunacm (64-bit) v10.3.0-275. Copyright (c) 2020 SafeNet. All rights reserved.


        Available HSMs:

        Slot Id ->              0
        Label ->                INTG_Par01
        Serial Number ->        1312109861420
        Model ->                LunaSA 7.7.1
        Firmware Version ->     7.7.1
        Bootloader Version ->   1.1.2
        Configuration ->        Luna User Partition With SO (PW) Key Export With Cloning Mode
        Slot Description ->     Net Token Slot
        FM HW Status ->         Non-FM

        Slot Id ->              7
        HSM Label ->            DKE-HA
        HSM Serial Number ->    11312109861420
        HSM Model ->            LunaVirtual
        HSM Firmware Version -> 7.7.1
        HSM Configuration ->    Luna Virtual HSM (PW) Key Export With Cloning Mode
        HSM Status ->           N/A - HA Group



        Current Slot Id: 0
```

See Using the Luna HSM inside Docker Container for Java Code Signing demonstration inside of a container using Luna Network HSM.

# Contacting Customer Support

If you encounter a problem while installing, registering, or operating this product, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

## Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

## Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.