

Docker Container

INTEGRATION GUIDE



Document Information

Document Part Number	007-000131-001
Release Date	February 2019

Revision History

Revision	Date	Reason
B	February 2019	Update

Trademarks, Copyrights, and Third-Party Software

© 2019 Gemalto. All rights reserved. Gemalto and the Gemalto logo are trademarks and service marks of Gemalto N.V. and/or its subsidiaries and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners.

Disclaimer

All information herein is either public information or is the property of and owned solely by Gemalto NV. and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemalto's information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

- > The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- > This document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

CONTENTS

PREFACE	6
Scope	6
Document Conventions	6
Command Syntax and Typeface Conventions	7
Support Contacts.....	8
Customer Support Portal	8
Telephone Support	8
Email Support.....	8
CHAPTER 1: Introduction	9
Supported Platforms	9
Prerequisites	10
Constraints on SafeNet Luna HSM and HSM on Demand Service.....	10
CHAPTER 2: Configuring Docker Container for SafeNet DPoD	11
Provision your HSM on Demand service	11
Configuring an HSMoD service client in Docker Container	12
CHAPTER 3: Configuring Docker Container for SafeNet Luna Network HSM	14
Configuring the SafeNet Luna Network HSM	14
Configuring the Docker Container for use with SafeNet Luna Network HSM.....	14
Creating the NTLS connection to SafeNet Luna Network HSM	15
Creating the Luna Client Docker image	17
Running the Docker Container.....	18
CHAPTER 4: Configuring Docker Swarm with SafeNet DPoD	19
Provision your HSM on Demand service	19
Configuring Docker Swarm with SafeNet DPoD	20
Creating the Luna Docker Image in Docker Registry	20
Setting up a Docker Swarm Cluster	21
Deploying the Application on the Swarm Manager.....	22
CHAPTER 5: Configuring Kubernetes with SafeNet Luna Network HSM	26
Configuring the SafeNet Luna Network HSM	26
Configuring and Installing Luna Minimal Client in Kubernetes	27
Creating the Luna Client Image in Docker Registry	27
Creating the Kubernetes Secrets.....	28
Deploying a Pod using the Luna Client Image and Kubernetes Secret	29
CHAPTER 6: Configuring OpenShift Origin with SafeNet DPoD	32
Provision your HSM on Demand service	32
Configuring DPoD in OpenShift Origin.....	33
Method I- Deploying pod using persistent volume.....	33

Method II- Deploying Pod using task file	38
CHAPTER 7: Configuring Apache Mesos with SafeNet DPoD	43
Provision your HSM on Demand service	43
Configuring DPoD in Apache Mesos.....	44
Creating the Luna Docker Image	44
Creating a sample Application in Marathon	45
Starting interactive session with the running Docker Container	46
APPENDIX A: Using SafeNet HSM inside of Docker Container	48
Using an HSMoD service client inside Docker Container	48
Using the SafeNet HSM inside Docker Container	52

PREFACE

This document is intended to guide security administrators through the steps for using a SafeNet Luna HSM or an HSM on Demand service with an example application inside of a Docker Container.

Scope

This guide demonstrates deploying a SafeNet Luna HSM or HSM on Demand service with an example application in Docker Container. This guide includes configurations for the Container orchestrators Docker Swarm, Kubernetes, Openshift and Apache Mesos and demonstrates using the HSM with Java Code Signer.

Document Conventions

This section provides information on the conventions used in this template.

Notes

Notes are used to alert you to important or helpful information. These elements use the following format:

NOTE: Take note. Notes contain important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. These elements use the following format:

CAUTION! Exercise caution. Caution alerts contain important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. These elements use the following format:

****WARNING**** Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury

Command Syntax and Typeface Conventions

Convention	Description
bold	<p>The bold attribute is used to indicate the following:</p> <ul style="list-style-type: none"> > Command-line commands and options (Type dir /p.) > Button names (Click Save As.) > Check box and radio button names (Select the Print Duplex check box.) > Window titles (On the Protect Document window, click Yes.) > Field names (User Name: Enter the name of the user.) > Menu names (On the File menu, click Save.) (Click Menu > Go To > Folders.) > User input (In the Date box, type April 1.)
<i>italic</i>	The italic attribute is used for emphasis or to indicate a related document. (See the <i>Installation Guide</i> for more information.)
Double quote marks	Double quote marks enclose references to other sections within the document. For example: Refer to “ Error! Reference source not found. ” on page Error! Bookmark not defined.
<variable>	In command descriptions, angle brackets represent variables. You must substitute a value for command line arguments that are enclosed in angle brackets.
[optional] [<optional>]	Square brackets enclose optional keywords or <variables> in a command line description. Optionally enter the keyword or <variable> that is enclosed in square brackets, if it is necessary or desirable to complete the task.
[a b c] [<a> <c>]	Square brackets enclose optional alternate keywords or variables in a command line description. Choose one command line argument enclosed within the braces, if desired. Choices are separated by vertical (OR) bars.
{ a b c } { <a> <c> }	Braces enclose required alternate keywords or <variables> in a command line description. You must choose one command line argument enclosed within the braces. Choices are separated by vertical (OR) bars.

Support Contacts

If you encounter a problem while installing, registering, or operating this product, refer to the documentation. If you cannot resolve the issue, contact your supplier or [Gemalto Customer Support](#).

Gemalto 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 Gemalto 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.gemalto.com>, is a where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable database 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 Gemalto Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support@gemalto.com.

CHAPTER 1: Introduction

Docker makes it easier to create, deploy and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship the application and its components out as a single package.

This guide demonstrates how to configure applications in Docker Container to use SafeNet Luna HSM or an HSM on Demand (HSMoD) service to increase security. 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 application.

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 SafeNet Luna HSM or an HSMoD 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.

*FIPS validation in progress for HSMoD services.

**HSMoD services do not have access to the secure audit trail.

Supported Platforms

List of the platforms which are tested with the following HSMs:

SafeNet Luna HSM: It is a standalone network-attached appliance that physically and logically secure cryptographic keys and cryptographic processing. The purpose of an HSM is to protect sensitive data from being stolen by providing a highly secure operation structure. HSMs are fully contained and complete solutions for cryptographic processing, key generation, and key storage.

This integration is supported/verified with SafeNet Luna HSM on the following operating systems:

- > RHEL 7

SafeNet Data Protection on Demand (DPoD): It is a cloud-based platform that provides on-demand HSM and key management services through a simple graphical user interface. With DPoD, 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 only the services that you need.

This integration is supported/verified with SafeNet DPoD on the following operating systems:

- > RHEL 7

Prerequisites

Before beginning the integration, ensure you have access to the following:

- > A host system with Docker installed (See the Docker Documentation at docs.docker.com for more information about installation and configuration procedures.)
- > Access to Docker Hub.
- > HSM device or service:
 - If using a SafeNet Luna HSM, install a copy of the Luna HSM client software and Luna Minimal Client tarball package.
 - If using a DPoD HSM on Demand service, you require access to an Application Owner account.

Constraints on SafeNet Luna HSM and HSM on Demand Service

Please consider the following when using the SafeNet Luna HSM or HSMoD service:

Using SafeNet Luna HSM or HSM on Demand Services in FIPS Mode

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 SafeNet HSM in FIPS mode, you have to make the following change in configuration file:

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

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

HSM on Demand Service in FIPS mode

HSMoD services operate in 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 HSM on Demand 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.

Verify HSM on Demand <slot> value

LunaCM commands work on the current slot. If there is only one slot, then it is always the current slot. If you are completing an integration using HSMoD services, you need to verify the slot on the HSMoD service to send the commands. If there is more than one slot, then use the **slot set** command to send a command to a specified slot. You can use **slot list** to determine which slot numbers are in use by which HSMoD service.

CHAPTER 2: Configuring Docker Container for SafeNet DPoD

SafeNet Data Protection on Demand (DPoD) HSM on Demand (HSMoD) services provide strong physical protection of secure assets, including keys, and should be considered a best practice when working on Docker Container.

This section demonstrates provisioning an HSMoD service inside of a Docker Container for use with applications available in the Docker Container.

Provision your HSM on Demand service

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.

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

To provision HSM on Demand Service

1. Log in to DPoD 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 Services DPOD,” check the box accepting these Terms of Service 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 displays.
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`) generates and is provided for downloading and installing on your client machine.
8. Transfer the client package to your host machine. You can use SCP, PSCP, WinSCP, FTPS or other secure transfer tool to transfer the client package.

Configuring an HSMoD service client in Docker Container

Create and run the Luna Docker Image to use the HSMoD service inside of the Docker Container. To create the Luna Docker Image you must create the Docker Container and install the HSMoD service client in the Docker Container.

To create the Luna Docker Image

1. Create the 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.

To configure an HSMoD service inside the 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. 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.

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.

See [Using an HSMoD service client inside Docker Container](#) for an application demonstration inside of a Docker using SafeNet DPoD.

CHAPTER 3: Configuring Docker Container for SafeNet Luna Network HSM

SafeNet 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 SafeNet 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 SafeNet Luna Network HSM using PKCS#11 or Java APIs.

Configuring the SafeNet Luna Network HSM

Before you get started with SafeNet Luna HSM ensure the following:

- > SafeNet Luna Network HSM appliance has a secure admin password.
- > SafeNet Luna Network HSM has a hostname suitable for your network.
- > SafeNet Luna Network HSM network parameters are set to work with your network.
- > SafeNet Luna Network HSM appliance initialized.
- > SafeNet Luna Network HSM has a partition for use inside Docker Container.

Configuring the Docker Container for use with SafeNet Luna Network HSM

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

- > **Error! Reference source not found.**
- > Creating the NTLS connection to SafeNet Luna Network HSM
- > Creating the Luna Client Docker image
- > Running the Docker Container

Install the Luna minimal client. The minimal client contains the run-time libraries required for a cryptographic application to connect to the SafeNet 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
```

Creating the NTLS connection to SafeNet Luna Network HSM

Open an NTLS connection between the Docker Container and the SafeNet Luna Network HSM. This allows the HSM device to communicate securely with the Docker application.

To create the NTLS connection to SafeNet 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 SafeNet 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. If you have not already done so, change the initial password set by the Partition SO.

```
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
```

Creating the Luna Client Docker image

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

Running the Docker Container

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

To run the 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 SafeNet Luna Network HSM partition.

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

See [Using the SafeNet HSM inside Docker Container](#) for an application demonstration inside of a Docker using SafeNet Luna Network HSM.

CHAPTER 4: Configuring Docker Swarm with SafeNet DPoD

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 an HSMoD service.

SafeNet DPOD – HSM on Demand (HSMoD) 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.

Provision your HSM on Demand service

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.

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

To provision HSM on Demand Service

1. Log in to DPoD 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 Services DPOD,” check the box accepting these Terms of Service 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 displays.
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`) generates and is provided for downloading and installing on your client machine.
8. Transfer the client package to your host machine. You can use SCP, PSCP, WinSCP, FTPS or other secure transfer tool to transfer the client package.

Configuring Docker Swarm with SafeNet DPoD

To use a SafeNet DPoD – HSMoD service inside a Docker swarm, complete the following procedures:

- > [Creating the Luna Docker Image in Docker Registry](#)
- > [Setting up a Docker Swarm Cluster](#)
- > [Deploying the Application on the Swarm Manager](#)

Creating the Luna Docker Image in Docker Registry

Use Docker Registry to configure the Docker Image that you intend to integrate with the HSMoD service. Customize the Docker image for integration with SafeNet 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 the file Dockerfile in the current working directory and add the following:

```
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
docker_swarm	latest	f190c59cd551	About a minute ago
ubuntu	xenial	b9e15a5d1e1a	10 hours ago

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

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.](#)

Setting up a Docker Swarm Cluster

Set up the nodes in the Docker Swarm cluster for integration with the HSMoD 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-
3vcz1rkswq78s7t5sor3hr1bmzda4z523g8rnwkb8m8nd7tnpt-9uk7csvuieqqdg4b85nkk5ty9
192.168.99.100:2377"

# docker-machine ssh myvm3 "docker swarm join --token SWMTKN-1-
3vcz1rkswq78s7t5sor3hr1bmzda4z523g8rnwkb8m8nd7tnpt-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                STATUS      ENGINE VERSION   HOSTNAME      STATUS      AVAILABILITY   MANAGER
qsanp3vxs2mtccv9wk8fxdwur * 18.06.1-ce  Ready           myvm1        Active      Leader
dac4sgcob2i4djab0vp74pay5 18.06.1-ce  Ready           myvm2        Active
73gsgheap7x7c3n35de9nn0mb 18.06.1-ce  Ready           myvm3        Active
```

Deploying the Application on the Swarm Manager

Execute the following on the Manager Node to configure the HSMoD service for your swarm configuration.

To deploy the application on the swarm manager

1. Copy all of 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
About a minute		latest_test.1.kot01ixg1oe8he3cixodk4hv7	Up

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 PORTS	IMAGE NAMES	COMMAND	CREATED	STATUS
4c2912fec394 minutes	deegupta1302/docker-swarm:latest	"/bin/bash"	3 minutes ago	Up 2
fd92e2aab65a minutes	deegupta1302/docker-swarm:latest	"/bin/bash"	3 minutes ago	Up 2

11. Access the worker node myvm2.

```
$ docker exec -it latest_test.4.15m3zn8a8606r9zbfmfnf3qypb /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 PORTS	IMAGE NAMES	COMMAND	CREATED	STATUS
eff7be65c9ec minutes	deegupta1302/docker-swarm:latest	"/bin/bash"	4 minutes ago	Up 4
13383a82145a minutes	deegupta1302/docker-swarm:latest	"/bin/bash"	4 minutes ago	Up 4

15. Access the worker node myvm3.

```
# docker exec -it latest_test.3.j9yldapoiiza71ijdk7y50xfnz /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 an HSMoD service client inside Docker Container](#) for an application demonstration inside of a Docker using SafeNet DPoD.

CHAPTER 5: Configuring Kubernetes with SafeNet 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 SafeNet Luna HSM.

SafeNet 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`.

Configuring the SafeNet Luna Network HSM

Ensure the following in your SafeNet Luna HSM setup:

HSM is setup, initialized, provisioned, and ready for deployment. Refer to the SafeNet 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:> ntlsh 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

```

Configuring and Installing 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.

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.

- > [Creating the Luna Client Image in Docker Registry](#)
- > [Creating the Kubernetes Secrets](#)
- > [Deploying a Pod using the Luna Client Image and Kubernetes Secret](#)

Creating the Luna Client Image in Docker Registry

Create a Docker image containing the minimal required packages and utilities for communicating with the SafeNet 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 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. Login 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
```

Creating 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 the 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 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

Deploying 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 on the Master or any Node connected to the Master.

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

```
[root@pod-with-lunaclient /]#
```

5. Verify 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 SafeNet Luna HSM. To verify the integration with the SafeNet Luna HSM, run any application in the Pod that uses the HSM services. See [Using the SafeNet HSM inside Docker Container](#) for an application demonstration inside of a Docker.

CHAPTER 6: Configuring OpenShift Origin with SafeNet DPoD

OpenShift is a container application platform for Docker and Kubernetes. OpenShift Origin integrates with SafeNet DPoD – HSM on Demand services, which provide strong physical protection of secure assets, including keys, and should be considered a best practice when working with OpenShift Origin.

This section demonstrates provisioning an HSM on Demand Service inside of 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.

Provision your HSM on Demand service

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.

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

To provision HSM on Demand Service

1. Log in to DPoD 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 Services DPoD,” check the box accepting these Terms of Service 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 displays.
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`) generates and is provided for downloading and installing on your client machine.
8. Transfer the client package to your host machine. You can use SCP, PSCP, WinSCP, FTPS or other secure transfer tool to transfer the client package.

Configuring DPoD in OpenShift Origin

Configure DPoD 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 configure DPoD service inside an OpenShift Origin pod.

You can deploy the OpenShift Origin pod using the following two methods:

- > Method I- Deploying pod using persistent volume
- > Method II- Deploying Pod using task file

Method I- Deploying pod using persistent volume

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 than you can't view previous data. We recommend using Persistent Volume. You can share this Persistent Volume with multiple pods at a time.

Creating the Luna Docker Image

To use an HSM on Demand service with OpenShift Origin you must create and run the Luna Docker image. Create the Docker file and extract the HSM on Demand service inside of 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 the file Dockerfile in the current working directory and add the following:

```
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 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
```

Configuring the HSMoD service inside OpenShift Origin

Configure the HSMoD service inside of OpenShift Origin for use with OpenShift Origin.

To configure the HSMoD 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.

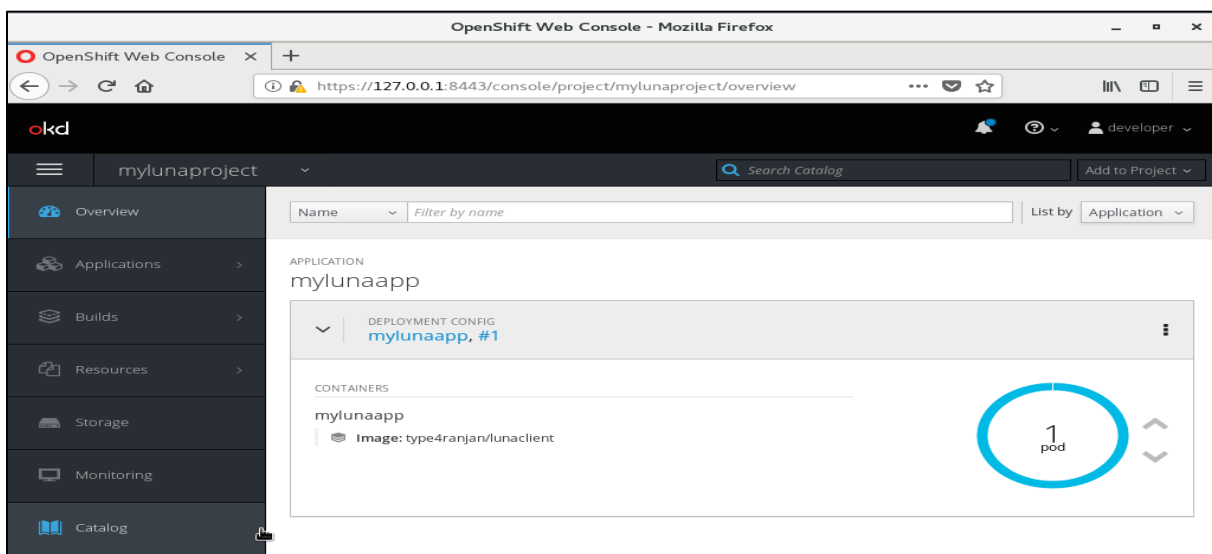
3. List all pods and their status.

```
# oc get pods
```

You will see output similar to the following:

```
[root@localhost ~]# oc get pods
NAME                READY   STATUS    RESTARTS   AGE
mylunaapp-1-v6jh9   1/1     Running   0           2m
[root@localhost ~]#
```

Verify the pod on the OpenShift Web Console:



Configuring Pod to run with root privileges

On the initial login to the pod console, the default user is 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 DPoD project.

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

2. Apply the patch to the application:

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

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

Adding 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)

1. 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
```

To 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.

- Click on **Add**. All the pods will automatically restart.

Copying 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 DPoD service.

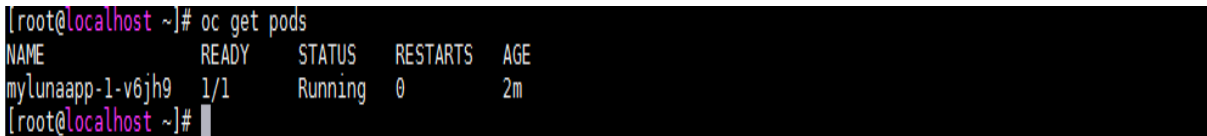
To copy secrets to persistent volume

- 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
```

- Get the running pod name with following command.

```
# oc get pods
```



```
[root@localhost ~]# oc get pods
NAME          READY   STATUS    RESTARTS   AGE
mylunaapp-1-v6jh9  1/1     Running   0           2m
[root@localhost ~]#
```

- 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.

Configuring the HSMoD service inside Pods

Deploy the HSMoD service client inside of a Pod. Execute the following on the terminal of a Pod where you want to use the HSMoD service.

To configure the HSMoD service inside Pods

- Open the terminal.

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

- Run `lunacm` and verify the connection to the partition:

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

```
# ./lunacm
```

- 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>
```

- 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 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.

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

```
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 SafeNet DPoD. To verify the integration with the HSMoD service, run any application in the Pod that uses the HSM services.

See [Using an HSMoD service client inside Docker Container](#) for an application demonstration inside of OpenShift Pod using SafeNet DPoD.

Method II- Deploying Pod using task file

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

Creating the Luna Docker Image

To use an HSM on Demand service with OpenShift Origin you must create and run the Luna Docker image. Create the Docker file and extract the HSM on Demand service inside of 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 the file Dockerfile in the current working directory and add the following:

```
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
```

Configuring the HSMoD service inside OpenShift Origin

Configure the HSMoD service inside of OpenShift Origin for use with OpenShift Origin.

To configure the HSMoD service inside OpenShift Origin

1. Create a project in OpenShift.

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

2. Make changes to 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 deployod.yaml and add the following:

```
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 statuses.

```
# oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mylunaapp-pod	1/1	Running	0	45s

Configuring Pod to run with root privileges

On the initial login to the pod console, the default user is 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 userroot
# oc adm policy add-scc-to-user anyuid -z userroot -n mylunaproject

```

2. Apply the patch to the application.

```

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

```

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

Configuring the HSMoD service inside Pods

Execute the following on the terminal of a Pod where you want to use the HSMoD service.

To configure the HSMoD 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
```

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.

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

```
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 SafeNet DPoD. To verify the integration with the HSMoD service, run any application in the Pod that uses the HSM services.

See [Using an HSMoD service client inside Docker Container](#) for an application demonstration inside of OpenShift Pod using SafeNet DPoD.

CHAPTER 7: Configuring Apache Mesos with SafeNet DPoD

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.

SafeNet DPoD – HSM on Demand service provides strong physical protection of secure assets, including keys, and should be considered a best practice when working with Apache Mesos.

This section demonstrates provisioning an HSMoD service for use with Apache Mesos.

NOTE: This integration assumes that an Apache Mesos configuration with an active Master (elected using ZooKeeper) and at least one Slave.

Provision your HSM on Demand service

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.

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

To provision HSM on Demand Service

1. Log in to DPoD 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 Services DPoD,” check the box accepting these Terms of Service 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 displays.
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`) generates and is provided for downloading and installing on your client machine.
8. Transfer the client package to your host machine. You can use SCP, PSCP, WinSCP, FTPS or other secure transfer tool to transfer the client package.

Configuring DPoD 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.

Creating the Luna Docker Image

To use an HSM on Demand service with OpenShift Origin you must create and run the Luna Docker image. Create the Docker file and extract the HSM on Demand service inside of 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 the file Dockerfile in the current working directory and add the following:

```
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 which contains all the files of directory secrets.
5. Build the Docker Image using the new Dockerfile.

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

6. Verify 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
```

Creating 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.

Starting interactive session with the running Docker Container

Deploy the HSMoD service to the Container application. Execute the following in the terminal of a Container where you want to use the HSMoD 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
```

- 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.

- 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.

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

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

- 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.

- 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 Apache Mesos with SafeNet DPoD. To verify the integration with the HSMoD service, run any application in the Pod that uses the HSM services.

See [Using an HSMoD service client inside Docker Container](#) for an application demonstration inside of Apache Mesos Pod using SafeNet DPoD.

APPENDIX A: Using SafeNet HSM inside of Docker Container

This section demonstrates using SafeNet Luna HSM and HSMoD service inside Docker container.

- > Using an HSMoD service client inside Docker Container
- > Using the SafeNet HSM inside Docker Container

Using an HSMoD service client inside Docker Container

This section demonstrates the method for using the Java keytool utility to generate signing keys and certificates on an HSMoD service. 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.

To configure the Java Keytool Utility to use the HSMoD 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 `<service_client_installation_directory>/jsp/lib` folder to the 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 HSMoD partition label, i.e. the name given to the HSMoD service during HSM initialization.

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

Save the file in current working directory.

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

1. Now generate a key pair using the Java `keytool` utility in the keystore. This will generate a key pair on the HSMoD service.

```
# 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 generates on the registered HSMoD service.

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

2. Verify that the private key is in the HSMoD service.

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

The system prompts for the keystore password. Enter the keystore password, and the system will output contents similar to the following:

```
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 the password, provide the keystore password.

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

File `certreq_file` generates 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 require the keystore file to install your Code Signing Certificate.

4. Copy the certificate request file to the host machine to submit it to your Certification Authority (CA)

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

For Example:

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

5. Provide the CSR to a CA for approval.
6. 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 is the CA Root certificate and signing.p7b is the Signed Certificate Chain respectively.

7. Import the CA's Root certificate in to the keystore.

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

Import the Signed Certificate Chain in to the keystore.

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

You can verify the keystore contents by executing partition contents in lunacm.

8. Copy the JAR file from the host machine to the Docker Container's current working directory. Execute the following command on the host machine:

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

For Example:

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

9. To sign the JAR, execute the following jarsigner command. The system will prompt you for the keystore password.

```
# 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
```

When complete, the system outputs:

```
jar signed
```

10. To verify the signed jar, execute the following. The system will output jar verified if the operation is successful.

```
# 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 an HSMoD service inside of a Docker Container. The JAR is signed and verified in Docker Container while the private key and certificate are securely stored on the HSMoD service.

Using the SafeNet HSM inside Docker Container

This section demonstrates the method for using the Java keytool utility to generate signing keys and certificates on SafeNet 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.

To configure the Java Keytool Utility to use the HSMoD 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.

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

1. Now generate a key pair using Java keytool utility in the keystore which will generate the key pair in SafeNet 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 SafeNet 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 SafeNet 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 prompt 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's 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. To sign the JAR, use jarsigner tool as follows, provide the keystore password when prompted and it will display the message after signing the jar.

```
# 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. To verify the signed jar, execute the following command and it will display the message at the end if 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 SafeNet 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 SafeNet Luna HSM.