Red Hat OpenShift

4.19.11

Integration Guide

u.trust GP HSM Se-Series

utimaco®

Imprint

Copyright 2025	Utimaco IS GmbH
	Krefelder Straße 220
	52070 Aachen
	Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226)
	EMEA +49 800-627-3081
	APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	1.0.0
Date	2025-11-07
Status	PUBLISHED
Document No.	IG-2025-0060

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco to facilitate secure cryptographic operations and key management in containerized environments. It outlines the integration of Red Hat OpenShift with Utimaco's Hardware Security Module (HSM), enabling secure key storage, hardware-based cryptographic processing, and centralized management of sensitive operations.

1.1 About This Guide

This guide describes how to integrate Utimaco Hardware Security Module (HSM) with Red Hat OpenShift to enable secure key management and cryptographic operations within containerized applications.

1.2 Target Audience

This guide is intended for Utimaco Hardware Security Module (HSM) and Red Hat OpenShift administrators.

1.3 Purpose of the Integration

The purpose of integrating Utimaco u.trust GP HSM Se-Series with Red Hat OpenShift is to ensure secure and centralized management of cryptographic keys used by containerized applications. This integration enables SSL/TLS operations to be performed directly inside the HSM using PKCS#11, ensuring that private keys remain protected in hardware at all times.

The primary objectives of this integration are:

- **Enhance security** by ensuring private keys never leave the HSM boundary.
- **Enable hardware-backed SSL/TLS** for applications running in OpenShift.
- **Improve operational security** by preventing key exposure in pods, images, or configuration files.

1.4 Abbreviations

Abbreviation	Meaning
HSM	Hardware Security Module
PKI	Public Key Infrastructure
PKCS	Public Key Cryptography Standards
PKCS#11	PKCS Part 11: The Cryptographic Token Interface Standard
SO	The PKCS#11 cryptographic slot Security Officer
JRE	Java Runtime Environment
MBK	Master Backup Key
P11CAT	the PKCS#11 graphical interface tool
CXI	Cryptographic eXtended Interface
FIPS	Federal Information Processing Standards
OC	OpenShift Client
HTTPS	Hypertext Transfer Protocol Secure
TLS	Transport Layer Security
IP	Internet Protocol

Abbreviation	Meaning
LAN	Local Area Network
PCIe	Peripheral Component Interconnect Express

Table 1: Abbreviations

1.5 Document Conventions

The following conventions are used in this guide:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press **OK**
`Monospaced`	File names, folder and directory names, commands, file outputs, programming code samples.	`chsm-create`
Italic	References and important terms	See *Sample Chapter* in the *SecurityServer - Sample Manual*

Table 2: Document Conventions

We use special icons to highlight the most important notes and information.

Here you will find important safety information that should be followed.

Here you will find additional notes or supplementary information.

This message marks the result expected after the successful execution of an instruction.

2 Product Overview

2.1 Overview of Red Hat OpenShift

Red Hat OpenShift is an enterprise Kubernetes platform that automates the deployment, scaling, and lifecycle management of containerized applications. It provides enhanced security features including container isolation, secure supply chain and cryptographic enforcement through its built-in trust policies.

For applications that require secure key storage and hardware-based cryptography, OpenShift can integrate with external Hardware Security Modules (HSMs) using PKCS#11. This enables workloads running in the cluster to securely access encryption keys stored inside the HSM for SSL/TLS operations and other cryptographic functions without exposing private keys in pods, configuration files, or images.

2.2 Overview of Utimaco u.trust GP HSM Se-Series

Utimaco u.trust GP HSM Se-Series is a hardware security module developed by Utimaco IS GmbH. It is a physically protected, specialized computer unit designed to perform sensitive cryptographic tasks and securely manage and store cryptographic keys and data. The u.trust GP HSM Se-Series can be used as a universal, independent security component for heterogeneous computer systems.

2.3 Joint Value Proposition

The integration of Red Hat OpenShift with the Utimaco u.trust GP HSM Se-Series delivers a unified and secure solution for cryptographic operations in cloud-native environments. OpenShift provides enterprise-grade application orchestration, while the HSM ensures hardware-protected cryptographic key storage and processing.

This combined solution offers significant benefits:

- **Hardware-backed Key Security:** Private keys remain inside the HSM at all times, preventing exposure even if application pods or node storage are compromised.
- **Trusted SSL/TLS for Applications:** NGINX and other services running in OpenShift can perform secure certificate-based authentication and encryption using HSM-managed keys via PKCS#11.
- **Centralized Cryptographic Control:** A single authoritative security appliance enforces policy-driven lifecycle management across all OpenShift workloads.

- **Scalable Cloud-Native Deployment:** Works seamlessly with multiple pods and replicas while maintaining consistent key usage across the cluster.

This integration enables organizations to protect sensitive application traffic, ensure cryptographic integrity, and adopt secure DevSecOps practices without compromising operational agility.

3 Integration Requirements and Prerequisites

Ensure that the system environment you will be using meets the following hardware and software requirements.

This guide assumes that the user has already installed and configured the required software.

3.1 Tested Versions

The following integration has been successfully tested with the Utimaco HSM and OpenShift.

Operating System	OpenSSL version	Nginx version	Red Hat OpenShift version	Utimaco SecurityServer version	Utimaco HSM
Linux Server (Rocky Linux 9.5)	3.2.2	1.20.1	4.19.11	6.2.0	u.trust GP HSM Se-Series

Table 3: List of tested versions

3.2 Hardware and Software Requirements

3.2.1 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	u.trust GP HSM Se-Series LAN with firmware SecurityServer 6.1.1 or higher
Utimaco PCI-e HSM	u.trust GP HSM Se-Series PCI-e with firmware SecurityServer 6.1.1 or higher

Table 4: List of hardware requirements

3.2.2 Software Requirements

Software	Software Requirements
Red Hat OpenShift	4.19.11
JRE 8	Java Runtime Environment 8
HSM Interface	SecurityServer PKCS#11 Provider
HSM Utility	SecurityServer PKCS#11 Tool (p11tool2)

Table 5: List of software requirements

3.3 Prerequisites

Before you begin, please ensure that you have:

- Installed and set up the operating system listed in *Tested Versions*.
- Installed and set up the HSM listed in *Tested Versions*.
- Replaced the HSM default admin with a new admin user.
- Created and stored the MBK on each HSM. Refer to the SecurityServer documentation to set up the MBK.
- Set up and configured SecurityServer. Refer to the SecurityServer documentation to set up the HSM.
- Set up and configured the PKCS#11 library according to your environment. Refer to the SecurityServer documentation for instructions on setting up and configuring the PKCS#11 library.
- Created the Security Officer (SO) user and crypto user/users.
- Installed and configured the OpenShift CLI (oc) in the operating system to connect to your target OpenShift cluster.
- Access to the container registry where the customized Utimaco sidecar and NGINX images are hosted.

4 Installation and Configuration

The following section outlines the procedures required to configure both Utimaco SecurityServer software and OpenShift cluster for seamless integration.

4.1 Setting Up the Utimaco SecurityServer Software

On Linux:

1. Copy the downloaded software to the appropriate location on the Oracle Database Server.
2. Create an “utimaco” folder under the `/opt` directory and create 2 directories `/opt/utimaco/bin and /opt/utimaco/lib`.

```
# mkdir -p /opt/utimaco/bin
# mkdir /opt/utimaco/lib
```

3. Copy the pkcs11 library file `libcs_pkcs11_R3.so` from the Utimaco SecurityServer software to the `/opt/utimaco/lib` directory and make the file executable.

```
cp ~/path_to_application_folder/lib/libcs_pkcs11_R3.so /opt/utimaco/lib
chmod +x /opt/utimaco/lib/libcs_pkcs11_R3.so
```

4. Copy the `csadm` and `p11tool2` files from the Utimaco SecurityServer software to the `/opt/utimaco/bin` directory and make both files executable.

```
# cd ~/path_to_application_folder
# cp csadm p11tool2 /opt/utimaco/bin
# chmod +x /opt/utimaco/bin/csadm /opt/utimaco/bin/p11tool2
```

5. Create the directory `/etc/utimaco`. Locate the Utimaco PKCS#11 configuration file in your SecurityServer directory, `Software\Linux\Crypto_APIs\PKCS11_R3\sample`. Copy the Utimaco PKCS#11 configuration file `cs_pkcs11_R3.cfg` to `/etc/utimaco` directory.

```
# mkdir /etc/utimaco
# cd ~/path_to_application_folder/Software/Linux/Crypto_APIs/PKCS11_R3/
sample
```

```
# cp cs_pkcs11_R3.cfg /etc/utimaco
```

On Windows:

On Windows, `cs_pkcs11_R3.cfg` will be automatically created and available in the `C:\ProgramData\UtimacoPKCS11_R3` folder as part of the SecurityServer software installation.

Edit the `cs_pkcs11_R3.cfg` file and make the appropriate changes to the file.

cs_pkcs11_R3.cfg (Sample file)

```
library = C:\oracle\extapi\64\hsm\utimaco\6.1.1.0\cs_pkcs11_R3.dll
slot = 0
pin = Oracle123

[Global]

# For Unix:

Logpath = /tmp

# For Windows:

# Logpath = C:/ProgramData/Utimaco/PKCS11_R3

# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)

Logging = 1

# Prevents expiring session after inactivity of 15 minutes

KeepAlive = true

# Set the Device to connect with

#[CryptoServer]

# Device specifier

Device = <HSM_IP>
```

For detailed guidance on commands and their parameters, please refer to the Utimaco u.trust GP HSM Se-Series documentation.

The HSM is available in either PCIe or LAN form factors. Depending on the type, the device configuration line will follow one of these formats:

- LAN-based HSM:
  `Device = 288@ipaddress`
- PCIe-based HSM:
  `Device = /dev/cs2.0`

Make sure to select the appropriate format based on your specific hardware setup.

`library` specifies the path where the cs_pkcs11_R3.dll file is located.

`Slot` indicates the slot number associated with the created USER.

`Pin` represents the password assigned to the USER.

To simplify your testing process, it's recommended that you enable the PKCS#11 log file by adjusting the logging settings. Specifically:

- Set the `LogPath` to a writable directory (not a specific file).
- Set the `Logging Loglevel` to 1 for basic logging. Increase it to 4 for more detailed output during testing.

This will generate a log file named `cs_pkcs11_R3.log` within the specified `LogPath` directory. Reviewing this log can help with troubleshooting if you encounter issues.

Once testing is complete, it's advisable to reduce `Logging Loglevel` to `1` or `2` to limit output to only critical or important messages.

4.2 Setting Up Red Hat OpenShift

Follow the instructions in the link to set up the OpenShift cluster: https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html-single/installing_on_a_single_node/index

5 Integration Steps

5.1 Configuration on Utimaco HSM

Ensure that the Security Officer (SO) user and Crypto User(s) are created and that the Utimaco SecurityServer HSM is properly initialized so it can be accessed by the containerized applications running on the OpenShift platform.

5.2 Configuration on OpenShift

5.2.1 Create Directories

1. Create a working directory named `hsm-build` under the path `/home/admin/`. This directory will be used to store all build-related files for the integration.
2. Place the OpenShift CLI (`oc`) binary inside the `/home/admin/hsm-build` directory. The `oc` tool will be included in the Utimaco sidecar Docker image to enable the container to interact with the OpenShift cluster — for example, to create and manage Secrets directly from within the key-generation job during deployment.

```
[root@master-node admin]# pwd
/home/admin
[root@master-node admin]# mkdir hsm-build
[root@master-node admin]# cd hsm-build
[root@master-node hsm-build]# ls
oc
```

Figure 1 : Create hsm-build Directory

3. Create a subdirectory named `hsm` under `/home/admin/hsm-build` and place all necessary Utimaco client library files (such as `libcs_pkcs11_R3.so`, `pkcs11_R3.cfg`, `p11tool2`, `csadm,` and `cxitool`) inside it. These files will be included later in the sidecar container image used for HSM communication.

```
[root@master-node hsm-build]# mkdir hsm
[root@master-node hsm-build]# cd hsm
[root@master-node hsm]# ls
csadm  cs_pkcs11_R3.cfg  cxitool  libcs_pkcs11_R3.so  p11tool2
```

Figure 2 : Create hsm Directory

If you change the directory paths, ensure it is updated in all the places where it is used, including configuration files, deployment specifications, and related settings, to maintain consistency and prevent errors.

5.2.2 Build Docker Images

This section outlines the process for building the required container images—one for the Utimaco sidecar and one for the NGINX application—and pushing them to a container registry for use in the OpenShift deployment.

5.2.2.1 Build Utimaco Sidecar Image

Ensure that all required directories and files are created (Refer to section *Create Directories*) before proceeding with the instructions below.

1. Create a Dockerfile named `Dockerfile.sidecar` inside the `/home/admin/hsm-build` directory.

2. Add the following contents to the `Dockerfile.sidecar` file and save it.

```
FROM redhat/ubi9:latest

# Install dependencies
RUN dnf install -y openssl-pkcs11 && dnf clean all

# Create directories
RUN mkdir -p /opt/utimaco/bin /opt/utimaco/lib /usr/local/bin

# Copy ONLY the Utimaco client files
COPY hsm/libcs_pkcs11_R3.so /opt/utimaco/lib/
COPY hsm/p11tool2 /opt/utimaco/bin/
COPY hsm/csadm /opt/utimaco/bin/
COPY hsm/cxitool /opt/utimaco/bin/

# Copy the oc client into the image
COPY oc /usr/local/bin/oc
# Make it executable
RUN chmod +x /usr/local/bin/oc

RUN chmod -R +x /opt/utimaco/bin
```

```
CMD ["sleep", "infinity"]
```

3. Build the Utimaco sidecar image using the Dockerfile.

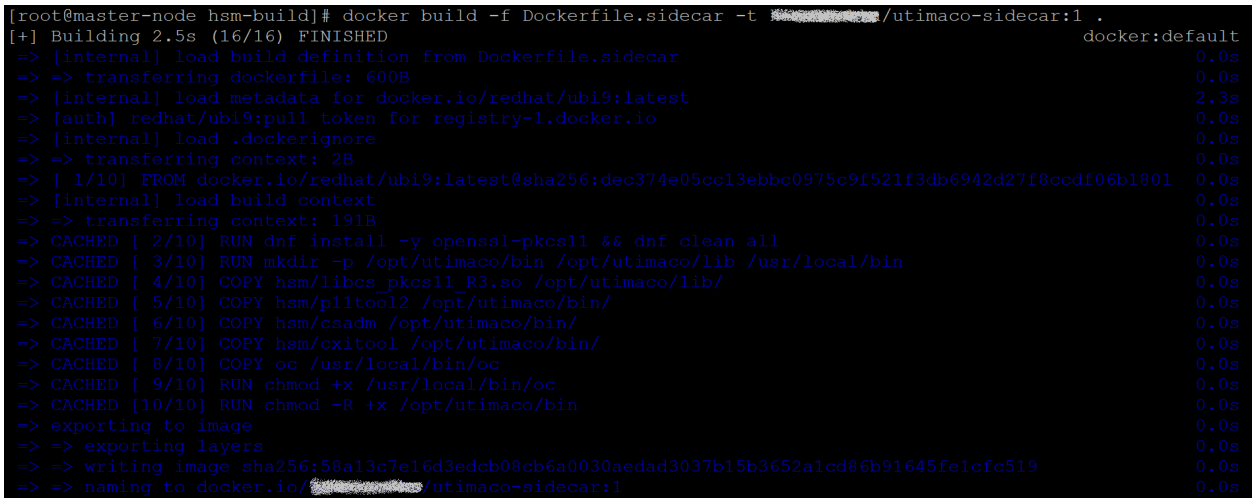

Figure 3 : Build Utimaco sidecar Image

In the example command, replace `<username>` with your actual Docker Hub username when building the image (e.g., `docker build -f Dockerfile.sidecar -t <your-username>/utimaco-sidecar:1 .`).

The screenshots in this guide show the build process with the username blurred for security reasons. Use your own Docker Hub username while executing the commands in your environment.

4. Verify that the image has been built successfully and push it to your Docker repository.

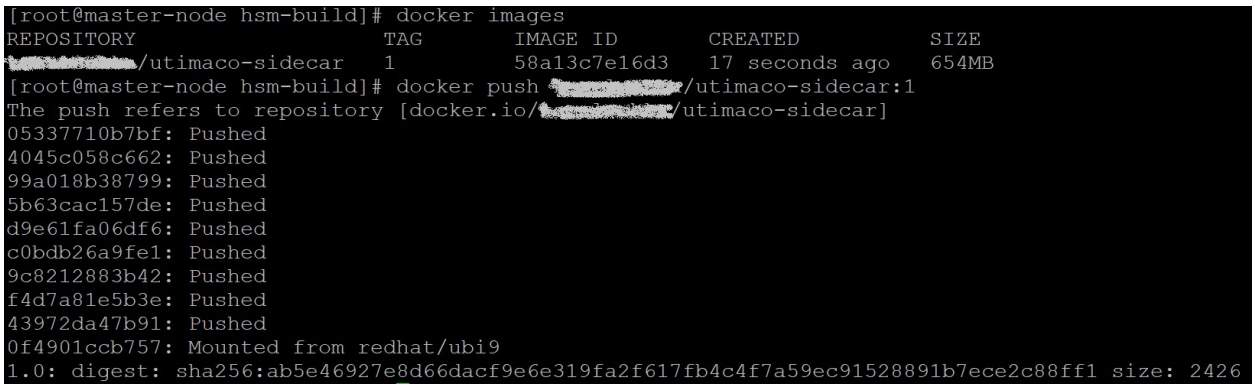

Figure 4 : Verify and Push sidecar Image

5.2.2.2 Build NGINX Image

1. Create a Dockerfile named `Dockerfile.nginx` inside the `/home/admin/hsm-build` directory.

2. Add the following contents to the `Dockerfile.nginx` file and save it.

```
FROM redhat/ubi9:latest

# Install NGINX + OpenSSL + PKCS#11 engine
RUN dnf install -y nginx openssl openssl-pkcs11 && dnf clean all

# Expose port and start NGINX
EXPOSE 443
CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
```

3. Build the nginx image using the Dockerfile.

Figure 5 : Build nginx Image

In the example command, replace `<username>` with your actual Docker Hub username when building the image (e.g., `docker build -f Dockerfile.nginx -t <your-username>/nginx:1 .`).

The screenshots in this guide show the build process with the username blurred for security reasons. Use your own Docker Hub username while executing the commands in your environment.

4. Verify that the image has been built successfully and push it to your Docker repository.

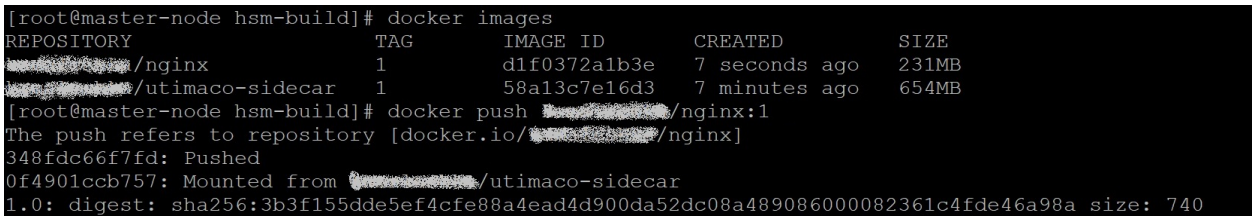

Figure 6 : Verify and Push nginx Image

5.2.3 Configure Job Permissions

Before deploying the key-generation job, you need to create specific roles and bindings that allow the job to create a Kubernetes Secret (which will store the generated SSL certificate).

This step ensures the key-generation job running in OpenShift has the necessary permissions to securely create and manage the SSL certificate secret used by your application pods.

1. Create a file named `job-permissions.yaml` in the path `/home/admin/hsm-build/`.

```
[root@master-node hsm-build]# vim job-permissions.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: secret-creator
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["create", "get", "delete" , "patch"]
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: key-generator-sa
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: secret-creator-binding
subjects:
- kind: ServiceAccount
  name: key-generator-sa
roleRef:
  kind: Role
  name: secret-creator
  apiGroup: rbac.authorization.k8s.io
~
~
```

Figure 7 : Create job-permissions.yaml

2. Paste the contents provided below and save the file.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: secret-creator
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["create", "get", "delete" , "patch"]
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: key-generator-sa
---
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: RoleBinding
metadata:
  name: secret-creator-binding
subjects:
- kind: ServiceAccount
  name: key-generator-sa
roleRef:
  kind: Role
  name: secret-creator
  apiGroup: rbac.authorization.k8s.io
```

3. Apply the configuration.

```
[root@master-node hsm-build]# oc apply -f job-permissions.yaml
role.rbac.authorization.k8s.io/secret-creator created
serviceaccount/key-generator-sa created
rolebinding.rbac.authorization.k8s.io/secret-creator-binding created
```

Figure 8 : Apply job-permissions.yaml

4. Verify that the Role, ServiceAccount, and RoleBinding are created successfully.

```
[root@master-node hsm-build]# oc get role,sa,rolebinding
NAME                                                  CREATED AT
role.rbac.authorization.k8s.io/prometheus-k8s         2025-09-23T10:36:31Z
role.rbac.authorization.k8s.io/secret-creator         2025-10-24T13:42:22Z

NAME                                SECRETS   AGE
serviceaccount/builder              0         31d
serviceaccount/default              0         31d
serviceaccount/deployer             0         31d
serviceaccount/key-generator-sa     0         2m

NAME                                                                                ROLE                                                AGE
rolebinding.rbac.authorization.k8s.io/machine-config-controller-events              ClusterRole/machine-config-controller-events        31d
rolebinding.rbac.authorization.k8s.io/machine-config-daemon-events                  ClusterRole/machine-config-daemon-events            31d
rolebinding.rbac.authorization.k8s.io/machine-os-builder-events                     ClusterRole/machine-os-builder-events               31d
rolebinding.rbac.authorization.k8s.io/prometheus-k8s                                Role/prometheus-k8s                                 31d
rolebinding.rbac.authorization.k8s.io/secret-creator-binding                        Role/secret-creator                                 2m
rolebinding.rbac.authorization.k8s.io/system:deployers                              ClusterRole/system:deployer                         31d
rolebinding.rbac.authorization.k8s.io/system:image-builders                         ClusterRole/system:image-builder                    31d
rolebinding.rbac.authorization.k8s.io/system:image-pullers                          ClusterRole/system:image-puller                     31d
```

Figure 9 : Verify Permissions

5.2.4 Create Configmaps

5.2.4.1 Create Utimaco ConfigMap

The Utimaco ConfigMap defines all configuration parameters required for the Utimaco PKCS#11 client and OpenSSL to communicate with the Utimaco Hardware Security Module (HSM). It includes the `cs_pkcs11_R3.cfg` file, which specifies the HSM connection details (such as IP, port, timeout, and session settings) and the `openssl.cnf` file, which instructs OpenSSL to load

the Utimaco PKCS#11 engine so that cryptographic operations are securely performed inside the HSM.

1. Create a file named `utimaco-cm.yaml` in the path `/home/admin/hsm-build/` .
2. Paste the contents provided below and save the file.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: utimaco-config
data:
  cs_pkcs11_R3.cfg: |
    [Global]
    Logging = 1
    Logpath = /var/log/utimaco
    Logsize = 10mb
    SlotMultiSession = true
    SlotLoginRestriction = true
    SlotCount = 10
    KeepLeadZeros = false
    FallbackInterval = 0
    KeepAlive = true
    ConnectionTimeout = 5000
    CommandTimeout = 60000
    KeysExternal = false

    [HSMCluster]
    # Point this to your SecurityServer simulator IP & port
    Devices = 3001@172.31.1.62

  openssl.cnf: |
    openssl_conf = openssl_init

    [openssl_init]
    engines = engine_section

    [engine_section]
    pkcs11 = pkcs11_section

    [pkcs11_section]
    engine_id = pkcs11
    dynamic_path = /usr/lib64/engines-3/pkcs11.so
    MODULE_PATH = /opt/utimaco/lib/libcs_pkcs11_R3.so
    PIN = 12345678
    init = 0
```

Before applying the ConfigMap, update the configuration values to match your environment:

- `Devices = 3001@172.31.1.62` → Replace with your actual HSM or simulator IP address and port.
- `PIN = 12345678` → Replace with your Crypto User PIN.
- `init = 0` → Represents the slot number (default is 0). Modify if your environment uses a different slot.

3. Apply the ConfigMap.

```
[root@master-node hsm-build]# oc apply -f utimaco-cm.yaml
configmap/utimaco-config created
```

Figure 10 : Apply Utimaco ConfigMap

5.2.4.2 Create NGINX ConfigMap

The NGINX ConfigMap defines the SSL/TLS configuration for NGINX to enable secure HTTPS communication using keys managed by the Utimaco HSM. It contains the `nginx.conf` file, which configures NGINX to use the Utimaco PKCS#11 engine for SSL operations, load the certificate from the mounted secret, and access the private key directly from the HSM for enhanced security.

1. Create a file named `nginx-cm.yaml` in the path `/home/admin/hsm-build/` .
2. Paste the contents provided below and save the file.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-config
data:
  nginx.conf: |
    user root;
    worker_processes  1;
    ssl_engine pkcs11;
    error_log /var/log/nginx/error.log warn;
    pid       /var/run/nginx.pid;
    events {
        worker_connections  1024;
```

```
    }
    http {
        server {
            listen               443 ssl;
            server_name          localhost;
            ssl_certificate      /etc/ssl/certs/tls.crt;
            ssl_certificate_key
"engine:pkcs11:pkcs11:object=RSAKey;type=private";
            location / {
                root   /usr/share/nginx/html;
                index  index.html index.htm;
            }
        }
    }
```

Before applying the ConfigMap, ensure that the file paths and configuration values match your environment:

- `ssl_certificate` → Must point to the correct mounted path of your SSL certificate (from the Kubernetes Secret, usually `/etc/ssl/certs/tls.crt`).
- `ssl_certificate_key` → Must use the same PKCS#11 object label (for example, `RSAKey`) that was generated and stored in the HSM during key generation.
- Confirm that the paths and engine configuration are aligned with your Deployment YAML (for example, the PKCS#11 library path and certificate mount path).

3. Apply the ConfigMap.

```
[root@master-node hsm-build]# oc apply -f nginx-cm.yaml
configmap/nginx-config created
```

Figure 11 : Apply NGINX ConfigMap

5.2.4.3 Verify ConfigMaps

Verify the applied Utimaco and NGINX ConfigMaps.

```
[root@master-node hsm-build]# oc get cm
NAME                        DATA   AGE
kube-root-ca.crt            1      34d
nginx-config                1      4m21s
openshift-service-ca.crt    1      34d
utimaco-config              2      11m
```

Figure 12 : Verify ConfigMaps

5.2.5 Create Key Generation Job

This step defines a Job that automatically generates an RSA key pair inside the Utimaco HSM during deployment. The key generation process is securely performed using the Utimaco PKCS#11 library, ensuring that the private key never leaves the HSM boundary. The same Job also generates a self-signed SSL certificate using the HSM key and stores it in an OpenShift Secret, which will later be mounted into the NGINX pods for SSL/TLS operations.

- The Label (RSAKey) used in this Job must match the key label referenced in your NGINX ConfigMap (`ssl_certificate_key "engine:pkcs11:pkcs11:object=RSAKey;type=private";`).
- Ensure that the Utimaco ConfigMap (`cs_pkcs11_R3.cfg` and `openssl.cnf`) has been created and applied before running this Job.
- Update the LoginUser (PIN) and HSM connection details in your configuration according to your environment setup.

1. Create a file named `key-generation.yaml` under the directory `/home/admin/hsm-build/` .
2. Paste the following content into the file and save it.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: key-generator-job
spec:
  template:
    spec:
      serviceAccountName: key-generator-sa
      volumes:
      - name: utimaco-cfg-volume
```

```
        configMap:
          name: utimaco-config
      - name: cert-volume
        emptyDir: {}
      containers:
      - name: key-generator
        image: <docker_username>/utimaco-sidecar:1
        command: ["/bin/sh", "-c"]
        args:
        - >
          /opt/utimaco/bin/p11tool2 slot=0 LoginUser=12345678 Label="RSAKey"
 DeleteObject || true;
          /opt/utimaco/bin/p11tool2 slot=0 LoginUser=12345678
 PubKeyAttr=CKA_LABEL="RSAKey" PrvKeyAttr=CKA_LABEL="RSAKey"
 GenerateKeyPair=RSA;
          openssl req -engine pkcs11 -new -x509 -days 365 -key
"pkcs11:object=RSAKey;type=private" -keyform engine -out /certs/tls.crt
-subj "/CN=test.utimaco.com";
          oc create secret generic ssl-cert --from-file=tls.crt=/certs/
tls.crt;
          oc create secret generic ssl-cert --from-file=tls.crt=/certs/
tls.crt --dry-run=client -o yaml | oc apply -f -;
        volumeMounts:
        - name: utimaco-cfg-volume
          mountPath: /etc/pki/tls/openssl.cnf
          subPath: openssl.cnf
        - name: utimaco-cfg-volume
          mountPath: /etc/utimaco/cs_pkcs11_R3.cfg
          subPath: cs_pkcs11_R3.cfg
        - name: cert-volume
          mountPath: /certs
      restartPolicy: Never
  backoffLimit: 4
```

Replace `<docker_username>` with your actual Docker Hub username for the images you built and pushed to your repository.

3. Apply the Job.

```
[root@master-node hsm-build]# oc apply -f key-generation.yaml
job.batch/key-generator-job created
```

Figure 13 : Apply Key Generation Job

4. Verify the Job completion.

```
[root@master-node hsm-build]# oc get jobs
NAME                 STATUS     COMPLETIONS   DURATION   AGE
key-generator-job    Complete   1/1           25s        16m
[root@master-node hsm-build]# oc get pods
NAME                       READY   STATUS      RESTARTS   AGE
key-generator-job-zwqpq    0/1     Completed   0          16m
```

Figure 14 : Verify Key Generation Job

5.2.6 Create Deployment

This step defines the OpenShift Deployment for deploying NGINX pods integrated with the Utimaco SecurityServer HSM.

The deployment uses a two-container structure:

- An Init Container (`utimaco-init`) to copy the required Utimaco PKCS#11 libraries and tools into a shared volume.
- The Main Application Container (`nginx-app`) which runs NGINX configured to use the HSM for SSL/TLS operations via the PKCS#11 interface.

The deployment also mounts the previously created ConfigMaps (Utimaco and NGINX) and the Secret (generated by the Key Generation Job), enabling full integration between OpenShift workloads and the HSM.

Ensure that the `ssl-cert` secret has already been created by the Key Generation Job before applying this deployment (refer to *Verify Secret Creation*).

1. Create a file named `deploy.yaml` in the path `/home/admin/hsm-build/` .
2. Paste the contents provided below and save the file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: securityserver
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-app
```

```
template:
  metadata:
    labels:
      app: my-app
  spec:
    volumes:
      # Shared library between initContainers and main app
      - name: pkcs11-shared-libs
        emptyDir: {}

      # Utimaco + OpenSSL configuration
      - name: utimaco-cfg-volume
        configMap:
          name: utimaco-config

      # NGINX configuration
      - name: nginx-cfg-volume
        configMap:
          name: nginx-config

      # Shared Secret created by Job (contains SSL.cert)
      - name: cert-volume
        secret:
          secretName: ssl-cert

      - name: utimaco-logs
        emptyDir: {}

    # --- INIT CONTAINERS ---
    initContainers:
      - name: utimaco-init
        image: <docker_username>/utimaco-sidecar:1
        command: ["/bin/sh", "-c"]
        args:
          - |
            echo "Copying Utimaco libraries and tools..."
            mkdir -p /shared/lib /shared/bin
            cp /opt/utimaco/lib/libcs_pkcs11_R3.so /shared/lib/
            cp /opt/utimaco/bin/* /shared/bin/
            echo "Libraries and tools are ready."
        volumeMounts:
          - name: pkcs11-shared-libs
            mountPath: /shared

    # --- MAIN CONTAINER ---
    containers:
      - name: nginx-app
        image: <docker_username>/nginx:1
        ports:
          - containerPort: 443
        env:
          - name: PKCS11_CONFIG
            value: /etc/utimaco/cs_pkcs11_R3.cfg
          - name: PKCS11_MODULE
```

```
              value: /opt/utimaco/lib/libcs_pkcs11_R3.so
            - name: PATH
              value: "/opt/utimaco/bin:/usr/local/sbin:/usr/local/bin:/usr/
sbin:/usr/bin:/sbin:/bin"
          volumeMounts:
            # Shared Utimaco library from init container
            - name: pkcs11-shared-libs
              mountPath: /opt/utimaco/lib
              subPath: lib
              readOnly: true
            - name: pkcs11-shared-libs
              mountPath: /opt/utimaco/bin
              subPath: bin
              readOnly: true

            # Mount OpenSSL configuration from ConfigMap
            - name: utimaco-cfg-volume
              mountPath: /etc/pki/tls/openssl.cnf
              subPath: openssl.cnf

            # Mount Utimaco PKCS#11 config from ConfigMap
            - name: utimaco-cfg-volume
              mountPath: /etc/utimaco/cs_pkcs11_R3.cfg
              subPath: cs_pkcs11_R3.cfg

            # Mount NGINX configuration
            - name: nginx-cfg-volume
              mountPath: /etc/nginx/nginx.conf
              subPath: nginx.conf

            # Mount shared certificate (Secret from Job)
            - name: cert-volume
              mountPath: /etc/ssl/certs
              readOnly: true

            - name: utimaco-logs
              mountPath: /var/log/utimaco

          lifecycle:
            preStart:
              exec:
                command:
                  - /bin/sh
                  - -c
                  - |
                    echo "? Waiting for SSL certificate..."
                    while [ ! -f /etc/ssl/certs/SSL.cert ] && [ ! -f /etc/
ssl/certs/tls.crt ]; do
                      sleep 2
                    done
                    echo "Certificate ready. Starting NGINX."
```

Replace `<docker_username>` with your actual Docker Hub username for the images you built and pushed to your repository.

3. Apply the deployment.

```
[root@master-node hsm-build]# oc apply -f deploy.yaml
deployment.apps/securityserver created
```

Figure 15 : Apply Deployment

4. Verify the deployment.

```
[root@master-node hsm-build]# oc get deployment
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
securityserver   2/2     2            2           66s
[root@master-node hsm-build]# oc get pods -o wide
NAME                              READY   STATUS      RESTARTS   AGE   IP            NODE                NOMINATED NODE   READINESS GATES
key-generator-job-9mgjd           0/1     Completed   0          19m   10.128.1.20   00-50-56-86-32-b6   <none>           <none>
securityserver-6bb57d6dbd-9mzsm   1/1     Running     0          72s   10.128.1.30   00-50-56-86-32-b6   <none>           <none>
securityserver-6bb57d6dbd-b8fx2   1/1     Running     0          71s   10.128.1.31   00-50-56-86-32-b6   <none>           <none>
```

Figure 16 : Verify Deployment

6 Verification and Testing

6.1 Functional Testing

6.1.1 Verify Secret Creation

After running the Key Generation Job, verify that the SSL certificate secret (`ssl-cert`) has been successfully created in the OpenShift environment.

```
[root@master-node ~]# oc get secrets
NAME               TYPE                                  DATA   AGE
my-docker-secret   kubernetes.io/dockerconfigjson   1      28d
ssl-cert           Opaque                                1      22s
```

Figure 17 : Verify Secret

This secret contains the SSL certificate generated using the RSA key stored securely inside the Utimaco HSM. During deployment, this secret will be mounted to the NGINX container to enable SSL/TLS communication using hardware-protected keys.

6.1.2 Verify HSM Key Presence and Connectivity

After deploying the application and running the key generation process, verify that the RSA key has been successfully created and that the OpenShift pods can communicate with the Utimaco HSM using the configured PKCS#11 libraries.

1. Login to the running securityserver pod.
2. Inside the pod, execute the `p11tool2` command to list the key objects stored in the HSM.

```
[root@master-node hsm-build]# oc get pods -o wide
NAME                              READY   STATUS      RESTARTS   AGE   IP            NODE                NOMINATED NODE   READINESS GATES
key-generator-job-9mgjd           0/1     Completed   0          19m   10.128.1.20   00-50-56-86-32-b6   <none>           <none>
securityserver-6bb57d6dbd-9mzsm   1/1     Running     0          72s   10.128.1.30   00-50-56-86-32-b6   <none>           <none>
securityserver-6bb57d6dbd-b8fx2   1/1     Running     0          71s   10.128.1.31   00-50-56-86-32-b6   <none>           <none>
[root@master-node hsm-build]# oc rsh securityserver-6bb57d6dbd-9mzsm
Defaulted container "nginx-app" out of: nginx-app, utimaco-init (init)
sh-5.1# p11tool2 LoginUser=12345678 ListObjects


CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE                  = CKK_RSA
  CKA_UNIQUE_ID                 = 284CEDEE-D469-46DA-AA20-8B5CBCA01374
  CKA_LABEL                     = RSAKey
  CKA_ID                        =



CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE                  = CKK_RSA
  CKA_UNIQUE_ID                 = 7852007E-4AEF-4E11-8F62-E4CDC9F40770
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = RSAKey
  CKA_ID                        =
```

Figure 18 : Verify Keys

3. Verify that an object with the label **"RSAKey"** is listed in the output.

This confirms that the RSA key pair was successfully generated during key generation job and securely stored inside the Utimaco HSM. Successful execution of this command also confirms that the deployed pod has successfully established secure connectivity with the HSM using the configured Utimaco PKCS#11 libraries.

6.1.3 Verify SSL/TLS Communication Between Pods

1. Get the list of running pods.

```
[root@master-node hsm-build]# oc get pods -o wide
NAME                              READY   STATUS      RESTARTS   AGE   IP            NODE                NOMINATED NODE   READINESS GATES
key-generator-job-9mgjd           0/1     Completed   0          19m   10.128.1.20   00-50-56-86-32-b6   <none>           <none>
securityserver-6bb57d6dbd-9mzsm   1/1     Running     0          72s   10.128.1.30   00-50-56-86-32-b6   <none>           <none>
securityserver-6bb57d6dbd-b8fx2   1/1     Running     0          71s   10.128.1.31   00-50-56-86-32-b6   <none>           <none>
```

Figure 19 : List Running Pods

2. Access one of the pods and use the openssl s_client command to initiate a secure connection to the other pod.

```
[root@master-node hsm-build]# oc rsh securityserver-6bb57d6dbd-b8fx2
Defaulted container "nginx-app" out of: nginx-app, utimaco-init (init)
sh-5.1# openssl s_client -connect 10.128.1.30:443
Connecting to 10.128.1.30
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN=test.utimaco.com
verify error:num=18:self-signed certificate
verify return:1
```

```
depth=0 CN=test.utimaco.com
verify return:1
---
Certificate chain
 0 s:CN=test.utimaco.com
   i:CN=test.utimaco.com
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Oct 27 14:11:19 2025 GMT; NotAfter: Oct 27 14:11:19 2026 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIC5TCCAc2gAwIBAgIUG5Ql5EJPaMiGD6iwEUQF0KQMIG8wDQYJKoZIhvcNAQEL
BQAwGzEZMBcGA1UEAwwQdGVzdC51dGltYWNvLmNvbTAeFw0yNTEwMjcxNDExMTla
Fw0yNjEwMjcxNDExMTlaMBsxGTAXBgNVBAMMEHRlc3QudXRpbWFjby5jb20wggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDvkHL2qdRC2kZmpg4qbXRW1wQX
DwR24A9QazpyEfNPyLjfI+YLO/3YMous10EhgaMVVNWuMaLTJ26UcZ9PNGSk2F8T
7kG6qpg7lhgSJxB/DXvAPEn8/LNnqrgS9ge/W14rSsUjjgp8426YjDldF+iYSGhM
7S4aCBYZwq1f0jugLvEY9JLtRG7rhBUFcp0FVGQqqT+mO2HBftLgYBf97DMsXz65
ehFkwNm8+D80TjfNxkaSA7iZrBm7NRuAAhs2iiouv9bfOuJeeE0egy4wfydQMGcm
idCqdy/rjFAxmzUgfGliSzNdnAGfnUs1+F1j0Cc3ZSuC45e3gBHBTb9DUZiHAgMB
AAGjITAfMB0GA1UdDgQWBBSQnNSaffb2VpY83t1IjKn2g0/GVjANBgkqhkiG9w0B
AQsFAAOCAQEASmEPi8pXjAcC7ElsdWy2HjahiIHLPvXuHPJJ1jvGzkVrwWFGBHxt
Uo+bLmffjaSB/gm1cRmM1SMbRh7Fy6fxZEQdIPS2bjFB+UZ9uP1icplMKlCQeLyx
26lAKcUeTD21dny1TFRgP/BjZUHEbHG7r0CFYtvBIYvnJCALV9U9eK5tgyUGK3Pi
/Ou2LMSmmNpMZ5I+Tvxua4D7VuMbanqM5f0EAta0vy5NDTVbWnGdQ4fH1SYQRg1T
MKoglbKsmXxBoca7LSaTta8PQWFEz8mhscI5tRfpy2Q0D54+BvsNS+UAv9CwpFuc
JUwMioabzLNp+n2ULCgvKGC22swPKfm50g==
-----END CERTIFICATE-----
subject=CN=test.utimaco.com
issuer=CN=test.utimaco.com
---
No client certificate CA names sent
Peer signing digest: SHA256
Peer signature type: RSA-PSS
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 1301 bytes and written 394 bytes
Verification error: self-signed certificate
---
New, TLSv1.3, Cipher is TLS_AES_256_GCM_SHA384
Server public key is 2048 bit
This TLS version forbids renegotiation.
Compression: NONE
Expansion: NONE
No ALPN negotiated
Early data was not sent
Verify return code: 18 (self-signed certificate)
---
---
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: 8F56101DF56CAEABDCF71EE192FA8282A10A897C0D79B82E9689AAC336EDF18F
    Session-ID-ctx:
    Resumption PSK: 69462F3D01EBE38AF02F15D6E4C4EBC5FBB2B28849BF1115F917B38E22AC996C4F9F75164AB248256BA8357A344FC7B0
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 49 1e 3f 2d 4f b7 03 f7-a4 d8 bf a8 93 08 4f b0   I.?-O.........O.
    0010 - 52 1b 40 3e ca 49 d2 c4-3f 49 e9 7a bf af b2 d4   R.@>.I..?I.z....
    0020 - b8 d2 66 2d 51 61 25 c5-fb a4 8d 05 c5 81 05 99   ..f-Qa%.........
    0030 - 53 1f 12 c2 b5 f6 5a 72-52 1c a2 34 74 c1 30 f5   S.....ZrR..4t.0.
    0040 - 64 a9 ee 35 c8 5c ae 94-74 bd c9 50 74 a4 76 41   d..5.\..t..Pt.vA
    0050 - e1 22 39 67 71 cd 3c 62-5b f5 e5 20 c4 4a 95 8f   ."9gq.<b[.. .J..
    0060 - 7c 2a 5e 7c 32 2e 75 ee-6c 74 4c 14 32 04 cd af   |*^|2.u.ltL.2...
    0070 - 25 c0 b3 2c ae fa 7e df-03 d8 ec d4 9c 7f ba 34   %..,..~........4
    0080 - 7f 76 0d 63 98 55 59 5a-a6 1c 94 c4 6c ea 3e 2c   .v.c.UYZ....l.>,
    Start Time: 1761651899
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
---
```

```
Post-Handshake New Session Ticket arrived:
SSL-Session:
    Protocol  : TLSv1.3
    Cipher    : TLS_AES_256_GCM_SHA384
    Session-ID: 763BEB20C124F4BFCE3714909436957C5F4A599A93C23A7CB238CE816E0F1C71
    Session-ID-ctx:
    Resumption PSK: 374F0B966A821C4103D278A53AEFD6EE07FB7A8B7325F01A3B03A95A7CE625D829A65E1E093A391EAA553A5C3EDA8F96
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - 49 1e 3f 2d 4f b7 03 f7-a4 d8 bf a8 93 08 4f b0   I.?-O.........O.
    0010 - a0 78 36 30 f4 79 c3 6e-74 7f cd d5 e6 76 af 35   .x60.y.nt....v.5
    0020 - 4b 0c a2 4c 16 af dd c3-a1 2b 33 94 67 33 31 b5   K..L.....+3.g31.
    0030 - 1b ad 2a ea a9 f6 c1 00-90 57 dc bf 6f 48 c7 6e   ..*......W..oH.n
    0040 - c5 12 9d 9f 5d 08 ff f9-92 b9 a7 54 0d 93 2d 81   ....]......T..-.
    0050 - 07 30 a7 50 65 98 77 f8-4a 06 3a da 94 81 31 7b   .0.Pe.w.J.:...1{
    0060 - 7e 35 07 42 18 c1 ff 43-8d fd e0 59 72 5c fb 38   ~5.B...C...Yr\.8
    0070 - 98 ee af 7c 8b be 15 c3-de 41 c2 10 e3 21 d7 05   ...|.....A...!..
    0080 - 3b c3 8e a2 92 cc ce 3a-b1 93 57 57 8c 46 64 9c   ;......:..WW.Fd.
    0090 - a1 3d 92 f3 95 5b ea c5-6e 8c 40 99 38 3e 0e 88   .=...[..n.@.8>..
    00a0 - 7d a2 98 d9 f3 5e 32 be-f4 c1 8b eb cc 14 eb 27   }....^2........'
    00b0 - 7f 4c 94 9e 22 f0 7d 79-17 8e f3 cd e6 6a 43 9e   .L.."."}y.....jC.
    00c0 - 8a ab 06 a7 db 0a 1a de-70 a8 1a 44 c1 e1 fa 6a   ........p..D...j
    00d0 - f7 0e 9d 6b 97 15 21 fe-60 b7 46 f4 ba 9b 68 69   ...k..!.`.F...hi

    Start Time: 1761651899
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
closed
```

Figure 20 : Openssl s_client connect output

3. The command should display the certificate details issued by the Utimaco-backed setup. and an output `CONNECTED(00000003)` indicating a successful SSL handshake.

Ignore the above error message as self-signed certificate has been used for demonstration. It is recommended to use CA signed certificate in production environment.

A successful TLS handshake using the `openssl s_client` command confirms that the NGINX server in the second pod is correctly utilizing the private key stored in the Utimaco HSM via the PKCS#11 interface to establish a secure HTTPS connection. This serves as the final validation that the Utimaco HSM and OpenShift integration is functioning as intended, providing secure, hardware-backed encryption for containerized applications.

6.2 Logs and Validation Steps

6.2.1 PKCS#11 Logs

Enabling PKCS#11 logging to facilitate easier testing and troubleshooting is recommended. This can be done by configuring the Logging Loglevel and LogPath parameters in the configuration file (`utimaco-cm.yaml`).

- LogPath should point to a writable directory (not a specific file) where log files can be stored.
- Logging Loglevel controls the verbosity of the logs:
  - Set it to 1 for basic logging.
  - For detailed testing and debugging, increase the level to 4.

The generated log file will be named `cs_pkcs11_R3.log` and located in the directory `/var/log/utimaco` of the securityserver pod specified by LogPath. Reviewing this log file can help identify and resolve issues that arise during testing.

Once testing is complete, it is advisable to reduce the Logging Loglevel to 1 or 2 to limit logging to only critical or important messages, thereby optimizing performance and reducing unnecessary log data.

```
[root@master-node hsm-build]# oc exec -it securityserver-59d9dc4f66-br121 -- cat /var/log/utimaco/cs_pkcs11_R3.log
Defaulted container "nginx-app" out of: nginx-app, utimaco-init (init)
28.10.2025 05:47:43.850 | [00000001:00000001] initialize                       | I: CryptoServer PKCS#11 Library R3 version 6.2.0.0 (Ju
l  1 2025)
28.10.2025 05:47:43.850 | [00000001:00000001] initialize                       | I: Configfile: /etc/utimaco/cs_pkcs11_R3.cfg
28.10.2025 05:47:43.850 | [00000001:00000001] initialize                       | I: Device List:
28.10.2025 05:47:43.850 | [00000001:00000001] initialize                       | I: Device 3001@172.31.1.62
28.10.2025 05:47:43.850 | [00000001:00000001] initialize                       | I: Multi-threaded access support enabled: true
28.10.2025 05:47:43.861 | [00000001:00000001] open_plugin                      | I: Opening KeyStorePlugin 'Ephemeral Storage' (config:
 )
28.10.2025 05:47:43.861 | [00000001:00000001] set_default_plugin_id            | I: Set new default KeyStorePlugin 'Legacy PKCS#11 Stor
age'
28.10.2025 05:47:43.882 | [00000001:00000001] open_plugin                      | I: Opening KeyStorePlugin 'Ephemeral Storage' (config:
 )
28.10.2025 05:47:43.882 | [00000001:00000001] set_default_plugin_id            | I: Set new default KeyStorePlugin 'Legacy PKCS#11 Stor
age'
28.10.2025 05:47:43.898 | [00000001:00000001] open_plugin                      | I: Opening KeyStorePlugin 'Ephemeral Storage' (config:
 )
28.10.2025 05:47:43.898 | [00000001:00000001] set_default_plugin_id            | I: Set new default KeyStorePlugin 'Legacy PKCS#11 Stor
age'
28.10.2025 05:47:43.911 | [00000001:00000001] open_plugin                      | I: Opening KeyStorePlugin 'Ephemeral Storage' (config:
 )
28.10.2025 05:47:43.911 | [00000001:00000001] set_default_plugin_id            | I: Set new default KeyStorePlugin 'Legacy PKCS#11 Stor
```

Figure 21 : Sample PKCS#11 Log

7 Optional Features

7.1 Scalability Recommendations

To verify that the Utimaco HSM integration scales seamlessly with additional application pods in the OpenShift cluster, increase the number of replicas and repeat the functional checks.

1. Scale the Deployment.

```
[root@master-node hsm-build]# oc scale deployment securityserver --replicas=5
deployment.apps/securityserver scaled
```

Figure 22 : Scale Pods

2. Get the list of running pods.

```
[root@master-node hsm-build]# oc get pods -o wide
NAME                              READY   STATUS      RESTARTS       AGE    IP            NODE                 NOMINATED NODE   READINESS GATES
key-generator-job-9mgjd           0/1     Completed   0              82m    10.128.1.20   00-50-56-86-32-b6    <none>           <none>
securityserver-6bb57d6dbd-6pf4w   1/1     Running     0              2m1s   10.128.1.59   00-50-56-86-32-b6    <none>           <none>
securityserver-6bb57d6dbd-9mzsm   1/1     Running     0              64m    10.128.1.30   00-50-56-86-32-b6    <none>           <none>
securityserver-6bb57d6dbd-b8fx2   1/1     Running     0              64m    10.128.1.31   00-50-56-86-32-b6    <none>           <none>
securityserver-6bb57d6dbd-bjv9p   1/1     Running     0              2m1s   10.128.1.58   00-50-56-86-32-b6    <none>           <none>
securityserver-6bb57d6dbd-gnbh2   1/1     Running     1 (113s ago)   2m1s   10.128.1.57   00-50-56-86-32-b6    <none>           <none>
```

Figure 23 : List Running Pods

3. Access one of the newly created pods and verify HSM connectivity by listing keys.

```
[root@master-node hsm-build]# oc rsh securityserver-6bb57d6dbd-bjv9p
Defaulted container "nginx-app" out of: nginx-app, utimaco-init (init)
sh-5.1# p11tool2 LoginUser=12345678 ListObjects


CKO_PUBLIC_KEY:

+ 1.1
  CKA_KEY_TYPE                  = CKK_RSA
  CKA_UNIQUE_ID                 = 284CEDEE-D469-46DA-AA20-8B5CBCA01374
  CKA_LABEL                     = RSAKey
  CKA_ID                        =



CKO_PRIVATE_KEY:

+ 2.1
  CKA_KEY_TYPE                  = CKK_RSA
  CKA_UNIQUE_ID                 = 7852007E-4AEF-4E11-8F62-E4CDC9F40770
  CKA_SENSITIVE                 = CK_TRUE
  CKA_EXTRACTABLE               = CK_FALSE
  CKA_LABEL                     = RSAKey
  CKA_ID                        =
```

Figure 24 : List Objects

4. From the new pod, perform an HTTPS connection to another pod.

```
[root@master-node hsm-build]# oc rsh securityserver-6bb57d6dbd-bjv9p
Defaulted container "nginx-app" out of: nginx-app, utimaco-init (init)
sh-5.1# openssl s_client -connect 10.128.1.57:443
Connecting to 10.128.1.57
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN=test.utimaco.com
verify error:num=18:self-signed certificate
verify return:1
depth=0 CN=test.utimaco.com
verify return:1
---
Certificate chain
 0 s:CN=test.utimaco.com
   i:CN=test.utimaco.com
   a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256
   v:NotBefore: Oct 27 14:11:19 2025 GMT; NotAfter: Oct 27 14:11:19 2026 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIC5TCCAc2gAwIBAgIUG5Ql5EJPaMiGD6iwEUQF0KQMIG8wDQYJKoZIhvcNAQEL
BQAwGzEZMBcGA1UEAwwQdGVzdC51dGltYWNvLmNvbTAeFw0yNTEwMjcxNDExMTla
Fw0yNjEwMjcxNDExMTlaMBsxGTAXBgNVBAMMEHRlc3QudXRpbWFjby5jb20wggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDvkHL2qdRC2kZmpg4qbXRW1wQX
DwR24A9QazpyEfNPyLjfI+YLO/3YMous10EhgaMVVNWuMaLTJ26UcZ9PNGSk2F8T
7kG6qpg7lhgSJxB/DXvAPEn8/LNnqrgS9ge/W14rSsUjjgp8426YjDldF+iYSGhM
7S4aCBYZwq1f0jugLvEY9JLtRG7rhBUFcp0FVGQqqT+mO2HBftLgYBf97DMsXz65
ehFkwNm8+D80TjfNxkaSA7iZrBm7NRuAAhs2iiouv9bfOuJeeE0egy4wfydQMGcm
idCqdy/rjFAxmzUgfGliSzNdnAGfnUs1+F1j0Cc3ZSuC45e3gBHBTb9DUZiHAgMB
AAGjITAfMB0GA1UdDgQWBBSQnNSaffb2VpY83t1IjKn2g0/GVjANBgkqhkiG9w0B
AQsFAAOCAQEASmEPi8pXjAcC7ElsdWy2HjahiIHLPvXuHPJJ1jvGzkVrwWFGBHxt
Uo+bLmffjaSB/gm1cRmM1SMbRh7Fy6fxZEQdIPS2bjFB+UZ9uP1icplMKlCQeLyx
26lAKcUeTD21dny1TFRgP/BjZUHEbHG7r0CFYtvBIYvnJCALV9U9eK5tgyUGK3Pi
/Ou2LMSmmNpMZ5I+Tvxua4D7VuMbanqM5f0EAta0vy5NDTVbWnGdQ4fH1SYQRg1T
MKoglbKsmXxBoca7LSaTta8PQWFEz8mhscI5tRfpy2Q0D54+BvsNS+UAv9CwpFuc
```

Figure 25 : Openssl s_client connect output

5. Successful output indicates that the new pod can securely communicate using the HSM-backed private key.

This test confirms that all newly created pods can securely connect to the Utimaco HSM using the configured PKCS#11 library, access the same cryptographic keys generated during the initial deployment, and successfully perform SSL/TLS operations demonstrating that the HSM integration scales seamlessly as additional replicas are added to the OpenShift deployment.

8 Troubleshooting

8.1 Common Issues and How to Resolve them

1. If pods get stuck in **CrashLoopBackOff** or **Init:Error**, verify that all ConfigMaps and Secrets are correctly applied and their names match those referenced in the deployment file.
2. If you encounter **Permission denied** errors on `/certs` or mounted volumes, ensure that proper permissions are set using an initContainer or appropriate security context for OpenShift's random UID policy.
3. If **SSL/TLS handshake fails** in NGINX pods, confirm that the SSL certificate and key paths are correctly mounted and that the PKCS#11 configuration points to the correct key label and PIN.
4. If the `p11tool2` **command is not found** inside the container, check that the Utimaco binaries were properly copied from the initContainer to `/opt/utimaco/bin`.
5. If there are **HSM connection errors or timeouts**, verify that the `Devices` parameter of `cs_pkcs11_R3.cfg` present in the `utimaco-cm.yaml` file points to the correct HSM IP and port and ensures that the HSM is reachable from the OpenShift node.
6. If the **Secret is missing** during deployment, make sure the key-generation Job completed successfully and created the `ssl-cert` Secret before applying the main deployment.
7. If **new pods fail to connect to the HSM after scaling**, confirm that Utimaco libraries and configuration volumes are mounted correctly and that the pods can list HSM objects using `p11tool2`.

8.2 Log Locations and Interpretation

8.2.1 PKCS#11 API Logging

1. The Utimaco PKCS#11 library supports detailed logging for all cryptographic operations (e.g., key generation, slot login, and certificate handling).
2. Logging is configured through the Utimaco ConfigMap (utimaco-config.yaml), which provides the file cs_pkcs11_R3.cfg mounted inside the container at /etc/utimaco/cs_pkcs11_R3.cfg.
3. To enable logging, set the following parameters:

- Logging = 1
- Logpath = /var/log/utimaco/

4. The log file cs_pkcs11_R3.log records API-level activities and helps diagnose issues such as PIN authentication failures, slot enumeration errors, or missing key labels.

8.2.2 OpenShift Job and Pod Logging

1. The Key Generator Job log provides insight into the certificate creation and secret management process within the OpenShift environment.
2. Key events include:

- Successful initialization of the PKCS#11 engine.
- RSA key pair and certificate generation within the HSM.
- Creation or update of the ssl-cert Kubernetes Secret.

3. Logs can be viewed using the command oc logs <key-generator-job-name>

```
[root@master-node hsm-build]# oc logs key-generator-job-9mgjd
2 Objects deleted
Engine "pkcs11" set.
secret/ssl-cert created
Warning: resource secrets/ssl-cert is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by oc a
pply. oc apply should only be used on resources created declaratively by either oc create --save-config or oc apply. The missing annota
tion will be patched automatically.
secret/ssl-cert configured
```

Figure 26 : Key Generator Pod Logs

9 Contact and Support Information

You can reach us from Monday to Friday, 09.00 a.m. to 05.00 p.m., Central European Time (CET).

Utimaco IS GmbH
Krefelder Straße 220

52070 Aachen
Germany

RMA Query

If you need to send the device back to Utimaco IS GmbH, please open a new RMA case (Return Merchandise Authorization). We request that you use the following web address. RMA cases cannot be opened by email or phone.

https://support.hsm.utimaco.com/support/rma/new

Other Support Queries

- Mail (preferred contact method)
  support@utimaco.com
  Attach the diagnostic information to your email.
- Web portal
  https://support.hsm.utimaco.com/support/cases/new/
  The diagnostic information will be requested in our response if necessary.
- By phone
  AMERICAS +1-844-UTIMACO (+1 844-884-6226)
  EMEA +49 800-627-3081
  APAC +81 800-919-1301
  The diagnostic information will be requested in our response if necessary.

10 Appendices

10.1 References

This document serves as a comprehensive guide for integrating Utimaco u.trust GP HSM Se-Series with OpenShift. For more information on other Utimaco products and offerings, please visit the official Utimaco Website.

The official document for setting up OpenShift Cluster will be in: https://docs.redhat.com/en/documentation/openshift_container_platform/4.19#Install

Title	Document Number
utrust_Anchor_LAN_V5_Operating_Manual	2021-0039
utrust_Anchor_PCIe_Operating_Manual	2020-0042
OpenShift OCP Ngix OpenSSL3 and SecurityServer & CP5	2025-0047
OpenShift OCP Ngix OpenSSL1.1 and SecurityServer & CP5	2025-0048

Table 6: References

10.2 Command Summary

Command	Purpose
`mkdir hsm-build`	To create a working directory to store all build-related files, such as Dockerfiles, YAML manifests, and the `oc` client.

Command	Purpose
`mkdir hsm`	To create a subdirectory inside `hsm-build` to store the Utimaco client binaries and libraries (`.so` , `p11tool2` , etc.).
`docker build -f Dockerfile.sidecar -t <docker_username>/utimaco-sidecar:1`	To build the Utimaco sidecar image containing the PKCS#11 library, Utimaco tools, and `oc` client.
`docker build -f Dockerfile.nginx -t <docker_username>/nginx:1`	To build the NGINX image integrated with the Utimaco HSM via PKCS#11 for SSL/TLS operations.
`docker images`	To list all Docker images available locally to verify that the sidecar and NGINX images were built successfully.
`docker push <docker_username>/utimaco-sidecar:1`	To push the Utimaco sidecar image to the Docker repository for use in the OpenShift cluster.
`docker push <docker_username>/nginx:1`	To push the custom NGINX image to the Docker repository for deployment in OpenShift.
`oc apply -f job-permissions.yaml`	To create the required Role, RoleBinding, and ServiceAccount to allow the key-generation job to create Secrets.
`oc get role,sa,rolebinding`	To verify that the Role, ServiceAccount, and RoleBinding objects were successfully created.

Command	Purpose
`oc apply -f utimaco-cm.yaml`	To apply the Utimaco ConfigMap, which contains the PKCS#11 and OpenSSL configuration files for the HSM integration.
`oc apply -f nginx-cm.yaml`	To apply the NGINX ConfigMap containing the HTTPS configuration that uses the PKCS#11 engine for key access.
`oc get cm`	To list all ConfigMaps in the namespace to verify that `utimaco-config` and `nginx-config` were created.
`oc apply -f key-generation-job.yaml`	To run the Key Generation Job that securely creates an RSA key pair inside the HSM and stores the certificate as a Secret.
`oc get jobs`	To verify the completion status of the key-generation job.
`oc apply -f deploy.yaml`	To deploy the SecurityServer application with NGINX configured to use the HSM via PKCS#11.
`oc get deployment`	To check the deployment status and ensure the pods are created successfully.
`oc get pods`	To view the list of running pods and verify that the NGINX pods are in the running state.

Command	Purpose
`oc get pods -o wide`	To get additional details about the running pods, such as IP addresses and assigned nodes.
`oc rsh securityserver-<pod>`	To open a remote shell session into a running NGINX pod.
`p11tool2 LoginUser=12345678 ListObjects`	To list all keys and objects stored inside the HSM using the PKCS#11 tool to verify key generation.
`openssl s_client -connect <pod_IP>:443`	To test SSL/TLS connectivity between pods to verify that the NGINX server is correctly using the HSM-backed certificate.
`oc scale deployment/securityserver --replicas=5`	To scale the number of NGINX pods (replicas) in the deployment.
`oc exec -it securityserver-<pod> -- cat /tmp/cs_pkcs11_R3.log`	To view PKCS#11 API logs generated inside the container.
`oc logs key-generator-job-<pod>`	To view the logs of the key-generation job to verify successful RSA key pair creation and Secret generation.

Table 7: CLI Commands