

nShield Security World

nShield Containers v13.9.9 User Guide

09 November 2026

© Entrust Corporation. All rights reserved.

Table of Contents

1. Introduction . . . 1
2. Outline . . . 2
3. Software prerequisites . . . 3
4. Hardserver containers . . . 4
    4.1. Creating the hardserver image . . . 4
    4.2. Configuring the hardserver container . . . 4
    4.3. Running the hardserver container with Docker . . . 5
5. Application containers . . . 6
    5.1. Creating the application image . . . 6
    5.2. Running the application container with Docker . . . 6
        5.2.1. Java API support . . . 7
6. Kubernetes deployment . . . 8
    6.1. Example setup . . . 8
7. Licensing . . . 9
    7.1. Authorized Use . . . 9
    7.2. License Restrictions . . . 9

1. Introduction

nShield Containers enables application developers to access the cryptographic functionality of an nShield Connect HSM in a container-based environment. This release has been tested with Docker.

2. Outline

One container runs the hardserver. It is supplied with configuration to connect to one or more network HSMs. It exposes the hardserver via an `AF_UNIX` socket.

Access to the hardserver socket must be restricted to trusted users.

Application instances are containers that run applications using the nShield software stack. These applications are supplied with the socket used to connect to the hardserver and access to the Security World key management data files and the associated cryptographic keys.

The key management data files, including encrypted copies of keys, are located in `NFAST_KMLOCAL`. A container that mounts `NFAST_KMLOCAL` as a volume can spoof the nShield Connect client. Therefore, access to files in `NFAST_KMLOCAL` must be restricted to trusted users.

3. Software prerequisites

nShield Containers requires Docker to be installed before you can use the nShield Container tools.

The hardserver and application images can be created without access to an HSM. Before you can begin using the containers in a deployment, you must complete the following steps:

1. Set up the HSM(s). For instructions, see the Installation Guide for your HSM(s).
2. Configure your container host machine as a client of the HSM(s). The container host machine is the machine on which you will run the nShield hardserver and application containers.
3. To access and use cryptographic keys from a Security World, load or create the Security World on the HSM, and then map the key management data folder (`kmdata`) from the container host machine into the running application containers.

For more information about configuring and managing nShield HSMs, Security Worlds, and Remote File Systems, see the User Guide.

4. Hardserver containers

The hardserver container controls communication between the configured nShield Connect HSMs and application containers. Only one hardserver container is **required** per deployment, regardless of the number of nShield Connects or application containers. For information about how many hardserver containers are **permitted** in a deployment, see License restrictions.

4.1. Creating the hardserver image

Create the hardserver image by using either the builder tool or the hardserver Dockerfile template located in `/opt/nfast/containers/templates`. The only required arguments are the path to a directory containing the nShield packages and the type of container (hardserver). The base image and tag can be supplied.

For example:

```
$ python3 /opt/nfast/containers/builder.py \
          --tag nshield-hardserver \
          --base redhat/ubi9-micro:9.8 \
          nshield-packages hardserver
```

```
$ docker build --build-arg "PACKAGES_DIR=." \
               --build-arg "BASE_IMAGE=redhat/ubi9-micro:9.8" \
               --file "/opt/nfast/containers/templates/hardserver/Dockerfile" \
               --target hardserver \
               --tag nshield-hardserver \
               nshield-packages
```

The default base image for nShield hardserver containers is Red Hat UBI10-Micro. When you use the builder tool, the default tag reflects the type of container built (nshield-container/hardserver). See `builder.py --help` for more options.

4.2. Configuring the hardserver container

The hardserver kneti file can be generated as follows:

```
/opt/nfast/sbin/hardserver --generate-kneti <path to kneti file>
```

The hardserver container's configuration file can be created from the default template generated with `cfg-mkdefault -f FILE`. The `nethsm_imports` section must contain the details of the HSMs to be used by the hardserver container:

```
[nethsm_imports]
local_module=0
remote_esn=1111-2222-3333
remote_ip=127.0.0.1
remote_port=9004
keyhash=0000000000000000000000000000000000000000
privileged=0
```

The keyhash can be retrieved with the `anonkneti` command.

Different configuration files can be used for different container deployments.

Verify the configuration file values against those displayed on the front panel or obtained through another trusted method.

4.3. Running the hardserver container with Docker

To run the hardserver container, you must:

- Supply the generated hardserver configuration to the container
- Mount a volume for the `/opt/nfast/sockets` folder
- Mount a hardserver kneti file in `/opt/nfast/hardserver.d/kneti-hardserver`

For example:

```
$ docker volume create nfast_socket
$ docker run \
    -v nfast_socket:/opt/nfast/sockets \
    -v <path to config>:/opt/nfast/kmdata/config/config:ro \
    -v <path to kneti file>:/opt/nfast/hardserver.d/kneti-hardserver:ro \
    nshield-hardserver
```

This makes the hardserver available via the sockets in the Docker volume nfast_socket.

The config volume should contain the hardserver's configuration file. The kmlocal volume should be a writable persistent volume for the keys, softcards and cardset files used by the application containers.

5. Application containers

An nShield application container is a container with the nShield Security World software installed. These images are intended to serve as examples and as base (`FROM`) images for pro duction applications.

5.1. Creating the application image

Create the application image using the builder tool or directly from the template application Dockerfile located in `/opt/nfast/containers/templates`. The only required arguments are the path to a directory containing the nShield packages and the type of container (app). The base image and tag can be supplied.

```
$ python3 /opt/nfast/containers/builder.py \
          --tag nshield-app \
          --base redhat/ubi9-micro:9.8 \
          nshield-packages app
```

```
$ docker build --build-arg "PACKAGES_DIR=." \
               --build-arg "BASE_IMAGE=redhat/ubi9-micro:9.8" \
               --file "/opt/nfast/containers/templates/app/Dockerfile" \
               --target app \
               --tag nshield-app \
               nshield-packages
```

The default base image for nShield application containers is Red Hat UBI10-Micro. When you use the builder tool, the default tag reflects the type of container built (nshield-container/app).

5.2. Running the application container with Docker

To run the base application container, you must:

- Supply a kmlocal folder (if you want to perform operations that require a Security World)
- Mount a volume for the `/opt/nfast/sockets` folder (same as the hardserver container)

Different application containers can use different kmdata folders.

```
$ docker run -it --rm \
    -v nfast_socket:/opt/nfast/sockets \
    -v <path to kmlocal>:/opt/nfast/kmdata/local \
    nshield-app
```

You can then copy the required Security World and module files for your application into the mounted kmlocal folder.

5.2.1. Java API support

To run Java application containers, create the Java base application image from the template provided. The base image does not include a JDK, so you must install one in any image derived from it. Java 21 or later are supported. If using earlier versions of Java, please contact support.

For example, a Dockerfile for a new Java application might look as follows:

```
FROM nshield-container/java:latest
RUN apt install myjdkprovider
COPY java-app/app.jar /
ENTRYPOINT "java -jar /app.jar RunApp"
```

6. Kubernetes deployment

nShield Containers can be used in a Kubernetes deployment.

6.1. Example setup

A set of example Kubernetes configuration files has been provided in `/opt/nfast/containers/kubernetes`. These example files can be used to create a deployment containing a pod made up of a hardserver and application container. The deployment uses volume claims for data and secrets for sensitive material. Each container has health checks configured using the `healthcheck` utility. This executes `Cmd_GetModuleState` on all configured modules.

Create the hardserver and application images using the commands in Hardserver containers and Application containers, and configure the deployment as shown below. The image tags must match those in the deployment configuration files.

```
$ cd /opt/nfast/containers/kubernetes/
$ kubectl apply -f 00-namespace.yml
$ kubectl apply -f 01-hostpath-volume.yml
$ kubectl apply -f 02-volume-claims.yml
$ bash 03-populate-volumes.sh <path to hardserver kneti> <path to config> <path to kmlocal>
$ kubectl apply -f 03-deployment.yml
$ kubectl wait -f 03-deployment.yml --timeout=3m --for=condition=Available
```

7. Licensing

This section defines the licensing terms and permitted uses for the nShield Containers software.

7.1. Authorized Use

In this document, when we refer to "you", we mean the Entrust customer ("Customer") who has purchased one or more nShield Containers licenses; or an individual authorized by Customer to access components, features or functions of nShield Containers ("Users"). Customer may also grant access to Users who are employees of external contractors, but only to the extent that such Users are using nShield Containers on Customer's behalf in the operation or management of the Customer's own business. Customer is responsible for all of its Users.

nShield Containers is:

- licensed for internal Customer use only (i.e., use for the Customer's own business purposes)
- compatible with all nShield network-attached hardware security modules (HSMs) (including nShield as a Service cloud-hosted HSM deployments)
- licensed on a per HSM basis (i.e., one nShield Containers license allows for the use of nShield Containers on a single network-attached HSM)
- supported for Linux distributions only



Except as may be otherwise specified in an express license agreement signed by Entrust, neither Customer nor any User may use nShield Containers for other companies (e.g. provision of nShield Containers functionality as a "Managed Service Provider" or "Systems Integrator").

7.2. License Restrictions

To use nShield Containers, you must have at least one supported nShield network-attached HSM and nShield Security World client software and Docker must be installed.

Your permitted number of containers will be dictated by the number of HSM client licenses you have purchased for each of your HSMs. nShield Containers uses a weighting factor for calculating the number of HSM client licenses required based on the maximum number of running hardserver containers to be deployed.

You may not alter your nShield Containers license, nor may you circumvent or attempt to circumvent the license limits.

For each network-attached HSM that you have purchased and want to use with nShield Containers, you must have an nShield Containers license and at least one available HSM client license (each purchased HSM includes three free HSM client licenses). You can also purchase a container client license multiplier ("license multiplier") for an HSM to increase the number of permitted hardserver container connections by ten per HSM client license. Each license multiplier applies to a single HSM only and applies to all HSM client licenses on that HSM.

For example, an HSM with an nShield Containers license, three available HSM client licenses, and a license multiplier could connect to up to 30 hardserver containers. If you pur chased a further two HSM client licenses for the HSM later, you would be able to connect to up to 50 hardserver containers.

If you do not purchase a license multiplier, you can only connect to one hardserver con tainer per available HSM client license.

To check your client license usage, and to interpret the `stattree` client counts and the `ClientLicenseLimitExceeded` error that is returned when a container is refused, see Con tainer client licensing.

Regular hardserver connections still require one HSM client license per connection. You can not use the same HSM client license for both a regular hardserver connection and a hard server container connection.

The chart below provides examples of combinations of the maximum number of regular and hardserver container connections to the HSM for 3 HSM client licenses with one license multiplier.

Number of regular hardserver con nections per HSM	Number of hardserver containers per HSM	Total client licenses used
3	0	3
2	10	3
1	20	3
0	30	3



Should you decide to purchase an enterprise HSM client license for your network-attached HSM, then there shall be no limits on the number of hardserver container connections permitted for that HSM.

You are strongly encouraged to keep your nShield Containers up to date with our latest product release.

The following are not included in your nShield Containers license and are sold separately (i.e., they would need to be covered in your Order):

- network-attached HSM (on-premises or cloud)
- other Entrust software or products
- support entitlement
- professional services

You are expected to keep reasonable records of the number of hardserver containers you have connected to your licensed network-attached HSMs and the number of application containers that you have run. Any increases in such numbers shall also be tracked in order to ensure that the requisite number of client licenses are in place.