Create NSO Docker Environments

Introduction

In this activity, you will learn how to set up and create the NSO in Docker environment to use for development and production purposes. The main reason for using Docker for NSO lies in ensuring a consistent, customizable, portable environment, that can be used for both development, testing, and production purposes.

First, you will build two Docker images—base and dev. The former is used as a production NSO image and contains the bare necessities that the NSO needs to operate, and the latter is used as a development image and contains various tools and compilers.
You will then use these base images to create a dedicated NSO in a Docker-based project for which you will develop, deploy, and automatically test a simple NSO service.

In addition to the project's Docker image, you will also learn how to create netsim Docker images that can be used as simulated virtual devices for your NSO in the Docker environment.

After completing this activity, you will be able to meet these objectives:

- Build NSO Docker images.
- Add dependencies for NSO Docker images.
- Develop and deploy packages with NSO in Docker.
- Create tests for packages using NSO in Docker.

Job Aids

The following job aid is available to help you complete the lab activities:

- This Lab Guide

The following table contains passwords that you might need.

Device	Username	Password
Student-VM	student	1234QWer
NSO application	admin	admin

Required Resources

The following resources and equipment are required for completing the activities in this lab guide:

- PC or laptop with a web browser
- Access to the Internet

Command Syntax Reference

This lab guide uses the following conventions for command syntax:

Formatting	Description and Examples
show running config	Commands in steps use this formatting.
Example	Type **show running config**
Example	Use the **name** command.
`show running config`	Commands in CLI outputs and configurations use this formatting.
highlight	CLI output that is important is highlighted.

Example	`student@student-vm:~$ ncs --version`
	`5.3.2`

Example	Save your current configuration as the default **startup config**. `Router Name# copy running startup`
brackets ([ ])	Indicates optional element. You can choose one of the options.
Example:	`(config-if)# frame-relay lmi-type {ansi|cisco|q933a}`

Formatting	Description and Examples
italics font	Arguments for which you supply values.
Example	Open file **ip tcp window-size** *bytes*
angle brackets (<>)	In contexts that do not allow italics, arguments for which you supply values are enclosed in angle brackets [<>]. Do not type the brackets when entering the command.
Example	If the command syntax is **ping** <*ip_address*>, you enter ping *192.32.10.12*
string	A non-quoted set of characters. Type the characters as-is.
Example	(config)# **hostname MyRouter**
vertical line (|)	Indicates that you enter one of the choices. The vertical line separates choices. Do not type the vertical line when entering the command.
Example	If the command syntax is **show ip route|arp**, you enter either **show ip route** or **show ip arp**, but not both.

Command List

The following are the most common commands that you will need.

Linux Shell:

Command	Comment
source /opt/ncs/ncs-5.3.2/ncsrc	Source NSO environmental variable in Docker container.
ls|ll	Display contents of the current directory.
cd	Move directly to user home directory.
cd ..	Exit out of current directory.
cd test	Move into folder "test" which is a subfolder of the current directory.
cd /home/student/nso300	Move into folder "nso300" by specifying direct path to it starting from the root of directory system.
ncs_cli -C -u admin	Log in to NSO CLI directly from local server.

NSO CLI:

Command	Comment
switch cli	Change CLI style.
show ?	Display all command options for current mode.
configure	Enter configuration mode.
commit	Commit new configuration (configuration mode only command).
show configuration	Display new configuration that has not yet been committed (configuration mode only command).

Makefile commands for Docker environment:

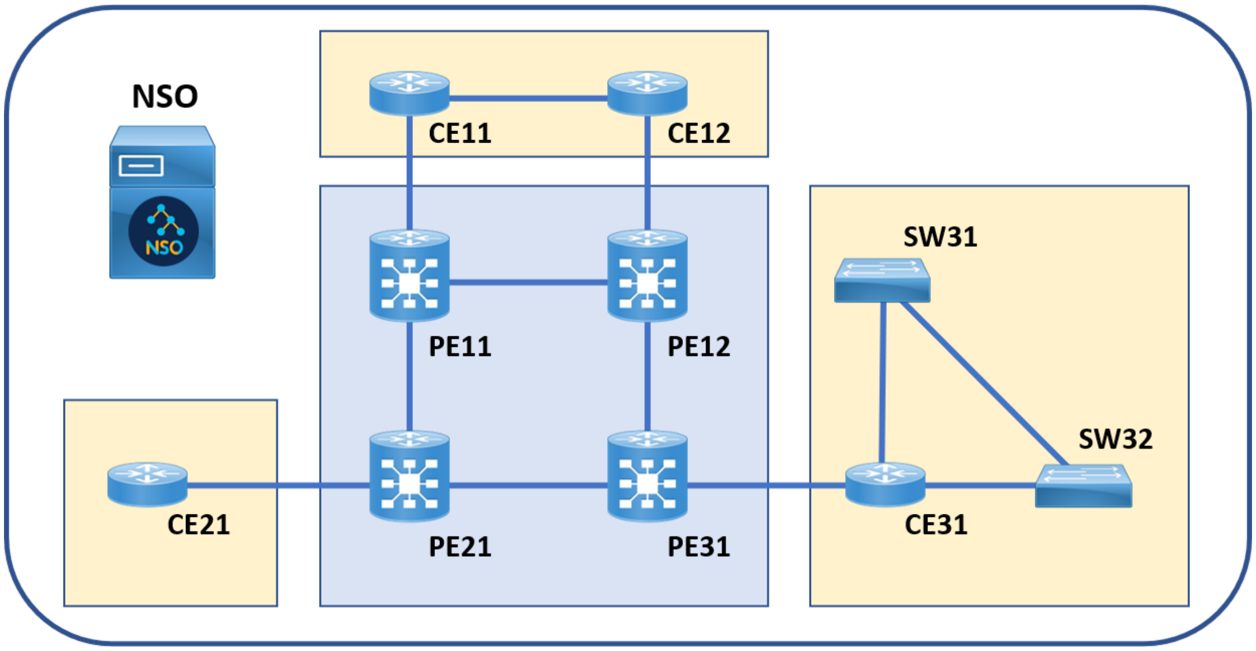
Command	Comment
make build	Builds the main NSO Docker image.
make testenv-start	Starts the NSO Docker environment.
make testenv-stop	Stops the NSO Docker environment.
make testenv-build	Recompiles and reloads the NSO packages.
make testenv-cli	Enters the NSO CLI of the NSO Docker container.
make testenv-shell	Enters the Linux shell of the NSO Docker container.
make dev-shell	Enters the Linux shell of the NSO Docker development container.

Lab Topology Information

Your lab session is your own personal sandbox. Whatever you do in your session will not be reflected in anyone else's session. There are two topologies. The general topology is your lab environment with a Linux server (Student-VM). On the Linux server within that topology is your second topology, a Docker environment, which consists of an NSO System container with your NSO installation, together with numerous Docker containers of NetSim routers and switches that are logically grouped into a network topology. This will be the network that you will orchestrate with your NSO.

- Network topology is designed in a way to cover both Service Provider and Enterprise use cases. It is a simulated NetSim network—devices have no Control or Data Plane. Devices will, however, accept or reject configurations sent by the NSO, just as real devices would.

Topology



Docker Topology

The following figure shows the topology of NSO Docker containers in relation to the build process that is triggered by the **make testenv-build** command.

The NSO Development container bind mounts the *~/nso300/packages* folder from the Student-VM machine to its */src/packages* folder.

The NSO System container bind mounts the packages to an intermediate build container, where the packages are recompiled. The recompiled packages are then copied to a volume, mounted on the NSO System container with the rsync tool, which is used to incrementally copy only the files that have been changed.

Task 1: Set Up the nso-docker Project and Images

In this task, you will set up an nso-docker project, then create and build development and testing NSO Docker images that will then be used as a base for your NSO deployment.

The goal of this activity is to introduce Docker and to produce a working NSO Docker environment. All the subsequent labs are based on an environment produced by this process.

Activity

Complete these steps:

Step 1

Connect to the Student-VM server by clicking the icon labelled NSO in the topology.

Step 2

Open the terminal window using the Terminal icon on the taskbar.

```
student@student-vm:~$
```

Step 3

Make sure that Docker is installed on the system, using the **docker -v** command.

The command should display your Docker version.

```
student@student-vm:~$ docker -v
Docker version 19.03.5, build 633a0ea838
student@student-vm:~$
```

Step 4

List the contents of your home directory with the **ls** command. Make sure that both the NSO installer and the nso-docker project are present.

```
student@student-vm:~$ ls
Desktop                                  nso-docker         snap
GNUstep                                  packages           solutions
lab                                      Pictures           thinclient_drives
nso300                                   scripts            vscode-server.tgz
nso-5.3.2.linux.x86_64.signed.bin        set_resolution.sh
student@student-vm:~$
```

The nso-docker project is already included in this lab, but can otherwise be found at https://github.com/NSO-developer/nso-docker. It is a repository that is run by Cisco and contains the files that are required to set up an NSO project in Docker. It also contains several project skeletons that can be used for different types of NSO projects—system, NED, and package.

Step 5

Unpack the NSO installer.

This installer will be used to install NSO on Docker images.

```
student@student-vm:~$ ./nso-5.3.2.linux.x86_64.signed.bin
Unpacking...
Verifying signature...
Downloading CA certificate from http://www.cisco.com/security/pki/
certs/crcam2.cer ...
Successfully downloaded and verified crcam2.cer.
Downloading SubCA certificate from http://www.cisco.com/security/pki/
certs/innerspace.cer ...
Successfully downloaded and verified innerspace.cer.
Successfully verified root, subca and end-entity certificate chain.
Successfully fetched a public key from tailf.cer.
Successfully verified the signature of
nso-5.3.2.linux.x86_64.installer.bin using tailf.cer
student@student-vm:~$
```

Step 6

Copy the NSO installer into your nso-docker repository.

```
student@student-vm:~$ cp nso-5.3.2.linux.x86_64.installer.bin nso-
docker/nso-install-files/
student@student-vm:~$
```

Step 7

Set the environmental variables. Set the DOCKER_REGISTRY, IMAGE_PATH and NSO_IMAGE_PATH to "**nso300.gitlab.local/**" and the NSO_VERSION to "**5.3.2**".

These variables are used to build and tag the NSO docker images. Usually, they are set within project-specific Makefiles, because multiple versions and locations of images can be used on a single machine.

```
student@student-vm:~$ export DOCKER_REGISTRY=nso300.gitlab.local/
student@student-vm:~$ export IMAGE_PATH=nso300.gitlab.local/
student@student-vm:~$ export NSO_IMAGE_PATH=nso300.gitlab.local/
student@student-vm:~$ export NSO_VERSION=5.3.2
```

This lab activity does not include an actual remote Docker image registry. Instead, you need to make sure that all the images needed are built locally and correctly tagged so that the build process simulates an actual development or production environment.

Step 8

Enter the *nso-docker* directory and build the two NSO images using the **make** command. This command takes some time to complete.

The **make** command executes a set of commands listed in the Makefile, that builds the base NSO image. The image should be built successfully.

```
student@student-vm:~$ cd nso-docker
student@student-vm:~/nso-docker$ make
The default make target will build Docker images out of all the NSO
versions found in nso-install-files/. To also run the test
suite for the built images, run 'make test-all'
make build-all

< … Output Omitted … >

Successfully built 3147c6c654aa
Successfully tagged nso300.gitlab.local/cisco-nso-base:5.3.2-student

< … Output Omitted … >

Successfully built 238ff3c386d7
Successfully tagged nso300.gitlab.local/cisco-nso-dev:5.3.2-student

rm -f *.bin
make[3]: Leaving directory '/home/student/nso-docker/docker-images'
make[2]: Leaving directory '/home/student/nso-docker'
make[1]: Leaving directory '/home/student/nso-docker'
student@student-vm:~/nso-docker$
```

Step 9

Verify that your local Docker registry now contains a *base* and a *dev* image by using the **docker images** command.

These images can be used as a base for NSO development and/or production.

```
student@student-vm:~/nso-docker$ docker images
REPOSITORY                           TAG                 IMAGE ID
CREATED              SIZE
nso300.gitlab.local/cisco-nso-base   5.3.2-student       eded19b9090b
About an hour ago    534MB
<none>                               <none>              ba882f85d30e
About an hour ago    706MB
nso300.gitlab.local/cisco-nso-dev    5.3.2-student       d97b14f50ee8
About an hour ago    1GB
<none>                               <none>              2a633be1d096
About an hour ago    706MB
debian                               buster              1b686a95ddbf
4 weeks ago          114MB
student@student-vm:~/nso-docker$
```

Docker generates Image and Container IDs randomly. The IDs in your registry will not match the IDs in the CLI outputs, but the image names should match.

Step 10

Re-tag the images using the **make tag-release** command. This action prepares the image for general use, not just for the current user.

```
student@student-vm:~/nso-docker$ make tag-release
docker tag nso300.gitlab.local/cisco-nso-dev:5.3.2-student
nso300.gitlab.local/cisco-nso-dev:5.3.2
docker tag nso300.gitlab.local/cisco-nso-base:5.3.2-student
nso300.gitlab.local/cisco-nso-base:5.3.2
student@student-vm:~/nso300$ docker images
REPOSITORY                           TAG                 IMAGE ID
CREATED              SIZE
nso300.gitlab.local/cisco-nso-base   5.3.2               eded19b9090b
2 hours ago          534MB
nso300.gitlab.local/cisco-nso-base   5.3.2-student       eded19b9090b
2 hours ago          534MB
<none>                               <none>              ba882f85d30e
2 hours ago          706MB
nso300.gitlab.local/cisco-nso-dev    5.3.2               d97b14f50ee8
2 hours ago          1GB
nso300.gitlab.local/cisco-nso-dev    5.3.2-student       d97b14f50ee8
2 hours ago          1GB
<none>                               <none>              2a633be1d096
2 hours ago          706MB
debian                               buster              1b686a95ddbf
4 weeks ago          114MB
student@student-vm:~/nso300$
```

Step 11

Enter the *nso300* folder in the home directory. This directory is used as the home directory for your NSO Docker project.

```
student@student-vm:~/nso-docker$ cd ~/nso300
student@student-vm:~/nso300$
```

Step 12

Copy the contents of the nso-system project skeleton from the nso-docker project into your *nso300* directory.

This is how the directory structure should appear:

```
student@student-vm:~/nso300$ cp -r ~/nso-docker/skeletons/system/* .
student@student-vm:~/nso300$ ls
Dockerfile.in  includes  nid          nidsystem.mk  README.nid-
system.org
extra-files    Makefile  nidcommon.mk  packages      test-packages
student@student-vm:~/nso300$
```

Step 13

Open the Makefile.

The Makefile from this project skeleton is used to set up, start, and test your project-specific NSO Docker environment. Now it contains nothing relevant. Most of the complexity is hidden in the pre-built *nidsystem.mk* library.

The following output shows how the file should appear when you open it for the first time:

```
student@student-vm:~/nso300$ vi Makefile
# You can set the default NSO_IMAGE_PATH & PKG_PATH to point to your
docker
# registry so that developers don't have to manually set these
variables.
# Similarly for NSO_VERSION you can set a default version. Note how the
?=
# operator only sets these variables if not already set, thus you can
easily
# override them by explicitly setting them in your environment and they
will be
# overridden by variables in CI.
# TODO: uncomment and fill in values for your environment
# Default variables:
#export NSO_IMAGE_PATH ?= registry.example.com:5000/my-group/nso-
docker/
#export PKG_PATH ?= registry.example.com:5000/my-group/
#export NSO_VERSION ?= 5.4

# Include standard NID (NSO in Docker) system Makefile that defines all
standard
# make targets
```

```
include nidsystem.mk

# The rest of this file is specific to this repository.

# For development purposes it is useful to be able to start a testenv
once and
# then run the tests, defined in testenv-test, multiple times,
adjusting the
# code in between each run. That is what a normal development cycle
looks like.
# There is usually some form of initial configuration that we want to
apply
# once, after the containers have started up, but avoid applying it for
each
# invocation of testenv-test. Such configuration can be placed at the
end of
# testenv-start-extra. You can also start extra containers with
# testenv-start-extra, for example netsims or virtual routers.

# TODO: you should modify the make targets below for your package
# TODO: clean up your Makefile by removing comments explaining how to
do things

# Start extra containers or place things you want to run once, after
startup of
# the containers, in testenv-start-extra.
testenv-start-extra:
	@echo "\n== Starting repository specific testenv"
# Start extra things, for example a netsim container by doing:
# docker run -td --name $(CNT_PREFIX)-my-netsim --network-alias
mynetsim1 $(DOCKER_ARGS) $(IMAGE_PATH)my-ned-repo/netsim:$(DOCKER_TAG)
# Use --network-alias to give it a name that will be resolvable from
NSO and
# other containers in our testenv network, i.e. in NSO, the above
netsim should
# be configured with the address 'mynetsim1'.
# Make sure to include $(DOCKER_ARGS) as it sets the right docker
network and
# label which other targets, such as testenv-stop, operates on. If you
start an
# extra NSO container, use $(DOCKER_NSO_ARGS) and give a unique name
but
# starting with '-nso', like so:
# docker run -td --name $(CNT_PREFIX)-nsofoo --network-alias nsofoo
$(DOCKER_NSO_ARGS) $(IMAGE_PATH)$(PROJECT_NAME)/nso:$(DOCKER_TAG)
#
# Add things to be run after startup is complete. If you want to
configure NSO,
# be sure to wait for it to start, using e.g.:
#docker exec -t $(CNT_PREFIX)-nso bash -lc 'ncs --wait-started 600'
#
# For example, to load an XML configuration file:
# docker cp test/initial-config.xml $(CNT_PREFIX)-nso:/tmp/initial-
config.xml
#	$(MAKE) testenv-runcmdJ CMD="configure\n load merge /tmp/
initial-config.xml\n commit"
```

```
# Place your tests in testenv-test. Feel free to define a target per
test case
# and call them from testenv-test in case you have more than a handful
of cases.
# Sometimes when there is a "setup" or "preparation" part of a test, it
can be
# useful to separate into its own target as to make it possible to run
that
# prepare phase and then manually inspect the state of the system. You
can
# achieve this by further refining the make targets you have.
testenv-test:
        @echo "\n== Running tests"
        @echo "TODO: Fill in your tests here"
# Some examples for how to run commands in the ncs_cli:
#       $(MAKE) testenv-runcmdJ CMD="show packages"
#       $(MAKE) testenv-runcmdJ CMD="request packages reload"
# Multiple commands in a single session also works - great for
configuring stuff:
#       $(MAKE) testenv-runcmdJ CMD="configure\n set foo bar\n commit"
# We can test for certain output by combining show commands in the CLI
with for
# example grep:
#       $(MAKE) testenv-runcmdJ CMD="show configuration foo" | grep bar
```

Step 14

Locate the environmental variables section and replace them with the variables that are required for this project. Set the NSO_VERSION to the NSO version that you are using (**5.3.2**) and set the NSO_IMAGE_PATH and IMAGE_PATH to that of your local Docker image registry.

To make the file easier to work with, remove all the comments.

```


export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"

testenv-test:
        @echo "\n== Running tests"
        @echo "TODO: Fill in your tests here"
```

Step 15

Save the file and exit the file editor.

Step 16

Build the NSO project image using the **make build** command.

```
student@student-vm:~/nso300$ make build
Checking NSO in Docker images are available...
-- Generating Dockerfile
cp Dockerfile.in Dockerfile
for DEP_NAME in $(ls includes/); do export DEP_URL=$(awk '{ print
"echo", $0 }' includes/${DEP_NAME} | /bin/sh -); awk "/DEP_END/ { print
\"FROM ${DEP_URL} AS ${DEP_NAME}\" }; /DEP_INC_END/ { print \"COPY --
from=${DEP_NAME} /var/opt/ncs/packages/ /var/opt/ncs/packages/\" }; 1"
Dockerfile > Dockerfile.tmp; mv Dockerfile.tmp Dockerfile; done
docker build --target nso -t nso300.gitlab.local/nso300/nso:5.3.2-
student --build-arg NSO_IMAGE_PATH=nso300.gitlab.local/ --build-arg
NSO_VERSION=5.3.2 --build-arg PKG_FILE=nso300.gitlab.local/nso300/
package:5.3.2-student .
Sending build context to Docker daemon  57.86kB
Step 1/12 : ARG NSO_IMAGE_PATH
Step 2/12 : ARG NSO_VERSION

< ... Output Omitted ... >

Successfully built 20a0e3a3eb84
Successfully tagged nso300.gitlab.local/nso300/nso:5.3.2-student
student@student-vm:~/nso300$
```

Step 17

Start the NSO Docker environment using the **make testenv-start** command.

This command starts the NSO System container. This container includes the NSO installation and is based on the base (production) NSO image.

```
student@student-vm:~/nso300$ make testenv-start
docker network inspect testenv-nso300-5.3.2-student >/dev/null 2>&1 ||
docker network create testenv-nso300-5.3.2-student
7687cb31bfc8498fb06ebd4bef4b25024d713b787cbd6aa847be55a2d614555f
docker run -td --name testenv-nso300-5.3.2-student-nso --network-alias
nso --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student --label nidtype=nso --volume /var/opt/ncs/packages
-e ADMIN_PASSWORD=NsoDocker1337 ${NSO_EXTRA_ARGS} nso300.gitlab.local/
nso300/nso:5.3.2-student
c921186ec65848902c7956f8c0090101cba791b1f01154cbfb41878f9f76ad35
make testenv-start-extra
make[1]: Entering directory '/home/student/nso300'

== Starting repository specific testenv
make[1]: Leaving directory '/home/student/nso300'
docker exec -t testenv-nso300-5.3.2-student-nso bash -lc 'ncs --wait-
started 600'
student@student-vm:~/nso300$
```

Step 18

Verify that the NSO System container is running. Use the command **docker ps -a** command.

```
student@student-vm:~/nso300$ docker ps -a
CONTAINER ID        IMAGE
COMMAND             CREATED             STATUS
PORTS                                       NAMES
c921186ec658        nso300.gitlab.local/nso300/nso:5.3.2-student   "/
run-nso.sh"        About a minute ago    Up About a minute (healthy)
22/tcp, 80/tcp, 443/tcp, 830/tcp, 4334/tcp   testenv-nso300-5.3.2-
student-nso
student@student-vm:~/nso300$
```

Step 19

Enter the NSO CLI using the **make testenv-cli** command.

This command takes you into the test NSO System container and executes the **ncs_cli** command, which takes you directly to the NSO CLI.

```
student@student-vm:~/nso300$ make testenv-cli
docker exec -it testenv-nso300-5.3.2-student-nso bash -lc 'ncs_cli -u
admin'

admin connected from 127.0.0.1 using console on c921186ec658
admin@ncs>
```

To enter the NSO System container Linux shell instead of NSO CLI, use the **make testenv-shell** command.

Step 20

Exit the NSO CLI. This action also exits the Docker container.

```
admin@ncs> exit
student@student-vm:~/nso300$
```

Activity Verification

You have completed this task when you attain these results:

- Two NSO Docker images are built.
- You can enter the NSO System container.

Task 2: Create Dependencies Using Project Skeletons

In this task, you will create NSO system image dependencies such as NEDs and set up netsim devices for your NSO environment inside Docker containers. Using dependencies allows you to quickly and consistently set up and edit your NSO environment.

Activity

Complete these steps:

Step 1

Enter the NSO System container again, switch CLI mode and list the devices and packages using **show** commands.

No devices are currently added to NSO and no packages are loaded.

```
student@student-vm:~/nso300$ make testenv-cli
docker exec -it testenv-nso300-5.3.2-student-nso bash -lc 'ncs_cli -u
admin'

admin connected from 127.0.0.1 using console on c921186ec658
admin@ncs> switch cli
admin@ncs# show devices brief
NAME  ADDRESS  DESCRIPTION  NED ID
----------------------------------
admin@ncs# show packages
% No entries found.
admin@ncs#
```

Step 2

Exit the NSO Docker container.

```
admin@ncs# exit
student@student-vm:~/nso300$
```

Step 3

Create a *neds* folder in home directory and navigate there.

```
student@student-vm:~/nso300$ mkdir ~/neds
student@student-vm:~/nso300$ cd ~/neds/
student@student-vm:~/neds$
```

Step 4

Create four directories there, one for each type of NED you will use in this lab: ned-ios, ned-iosxr, ned-nx, and ned-asa.

```
student@student-vm:~/neds$ mkdir ned-ios
student@student-vm:~/neds$ mkdir ned-iosxr
student@student-vm:~/neds$ mkdir ned-nx
student@student-vm:~/neds$ mkdir ned-asa
student@student-vm:~/neds$
```

Step 5

Copy the NED project skeleton contents from the nso-docker repository to each one of the directories.

```
student@student-vm:~/neds$ cp -r ~/nso-docker/skeletons/ned/* ned-ios/
student@student-vm:~/neds$ cp -r ~/nso-docker/skeletons/ned/* ned-
iosxr/
student@student-vm:~/neds$ cp -r ~/nso-docker/skeletons/ned/* ned-nx/
student@student-vm:~/neds$ cp -r ~/nso-docker/skeletons/ned/* ned-asa/
student@student-vm:~/neds$
```

Instead of creating a netsim lab locally with the command **ncs-netsim**, you build the images using a NED project skeleton, include them as dependencies to NSO Docker images and start them within your test environment.

Step 6

Copy the NEDs from *~/packages* into the *packages* subdirectory of their respective NED folder.

```
student@student-vm:~/neds$ cp -r ~/packages/cisco-ios-cli-6.54 ned-ios/
packages/
student@student-vm:~/neds$ cp -r ~/packages/cisco-iosxr-cli-7.26 ned-
iosxr/packages/
student@student-vm:~/neds$ cp -r ~/packages/cisco-nx-cli-5.15 ned-nx/
packages/
student@student-vm:~/neds$ cp -r ~/packages/cisco-asa-cli-6.10 ned-asa/
packages/
student@student-vm:~/neds$
```

Step 7

Enter the *ios* NED directory and build the image using the **make build** command. Tag the image for release with the **make tag-release** command.

The **make build** command should produce 3 images—*netsim*, *testnso, and package*. The netsim image is used to host a netsim device, the testnso image is used in automated testing of the NED, and the package image contains the compiled NED package.

```
student@student-vm:~/neds$ cd ned-ios
student@student-vm:~/neds/ned-ios$ make build
Checking NSO in Docker images are available...
-- Generating Dockerfile

< … Output Omitted … >

Successfully built 448d4417b49d
Successfully tagged nso300.gitlab.local/ned-ios/netsim:5.3.2-student

< … Output Omitted … >

Successfully built 5e2d220af242
Successfully tagged nso300.gitlab.local/ned-ios/testnso:5.3.2-student

< … Output Omitted … >

Successfully built 243ad64b1f4e
Successfully tagged nso300.gitlab.local/ned-ios/package:5.3.2-student
student@student-vm:~/neds/ned-ios$ make tag-release
docker tag nso300.gitlab.local/ned-ios/package:5.3.2-student
nso300.gitlab.local/ned-ios/package:5.3.2
docker tag nso300.gitlab.local/ned-ios/netsim:5.3.2-student
nso300.gitlab.local/ned-ios/netsim:5.3.2
student@student-vm:~/neds/ned-ios$
```

Step 8

Repeat the previous step for *iosxr*, *nx,* and *asa* directories. Build and tag the images using the **make build** and **make tag-release** commands.

```
student@student-vm:~/neds/ned-ios$ cd ../ned-iosxr
student@student-vm:~/neds/ned-iosxr$ make build
Checking NSO in Docker images are available...
-- Generating Dockerfile

< … Output Omitted … >

student@student-vm:~/neds/ned-iosxr$ make tag-release

< … Output Omitted … >

student@student-vm:~/neds/ned-iosxr$ cd ../ned-nx
student@student-vm:~/neds/ned-nx$ make build
Checking NSO in Docker images are available...
-- Generating Dockerfile

< … Output Omitted … >

student@student-vm:~/neds/ned-nx$ make tag-release

< … Output Omitted … >

student@student-vm:~/neds/ned-nx$ cd ../ned-asa
student@student-vm:~/neds/ned-asa$ make build
Checking NSO in Docker images are available...
-- Generating Dockerfile

< … Output Omitted … >

student@student-vm:~/neds/ned-asa$ make tag-release

< … Output Omitted … >
```

```
student@student-vm:~/neds/ned-asa$
```

Step 9

Clean up the intermediate Docker images with the **docker system prune -f** command.

This command removes unused data in your Docker environment, that is, unused containers and networks, dangling and unreferenced images and volumes.

```
student@student-vm:~/neds/ned-asa$ docker system prune -f
Deleted Containers:
dfe970704b85545f95f153e24ad2f0918743abb81a6920aedd415900881ae839
4abff6802b3943903317f81d7967eb3598bdcf9f9b3a19628ccad4e9422642f6

< … Output Omitted … >

student@student-vm:~/neds/ned-asa$
```

Step 10

Verify that the *netsim* and *package* images with tag 5.3.2 exist for every NED by listing the Docker images with the **docker images** command.

These are the eight Docker images that should be present for the setup of the dependencies:

```
student@student-vm:~/neds/ned-asa$ docker images
REPOSITORY                              TAG                 IMAGE ID
CREATED             SIZE
nso300.gitlab.local/ned-nx/package      5.3.2
44721a15a45b        45 seconds ago      22.8MB
nso300.gitlab.local/ned-nx/package      5.3.2-student
44721a15a45b        45 seconds ago      22.8MB
nso300.gitlab.local/ned-nx/testnso      5.3.2-student
f1bb18d24c8f        47 seconds ago      557MB
nso300.gitlab.local/ned-nx/netsim       5.3.2
d395c0343a58        50 seconds ago      1.03GB
nso300.gitlab.local/ned-nx/netsim       5.3.2-student
d395c0343a58        50 seconds ago      1.03GB
nso300.gitlab.local/ned-asa/package     5.3.2
fcc07bdf43cf        2 minutes ago       17.8MB
nso300.gitlab.local/ned-asa/package     5.3.2-student
fcc07bdf43cf        2 minutes ago       17.8MB
nso300.gitlab.local/ned-asa/testnso     5.3.2-student
0724750d3e84        2 minutes ago       552MB
nso300.gitlab.local/ned-asa/netsim      5.3.2
cb6c7c74d201        2 minutes ago       1.02GB
nso300.gitlab.local/ned-asa/netsim      5.3.2-student
cb6c7c74d201        2 minutes ago       1.02GB
nso300.gitlab.local/ned-iosxr/package   5.3.2
62794ea4f0af        3 minutes ago       131MB
nso300.gitlab.local/ned-iosxr/package   5.3.2-student
62794ea4f0af        3 minutes ago       131MB
```

```
nso300.gitlab.local/ned-iosxr/testnso   5.3.2-student
143808231d20        3 minutes ago       665MB
nso300.gitlab.local/ned-iosxr/netsim    5.3.2
0937703968a3        3 minutes ago       1.14GB
nso300.gitlab.local/ned-iosxr/netsim    5.3.2-student
0937703968a3        3 minutes ago       1.14GB
nso300.gitlab.local/ned-ios/package     5.3.2
135789184d9b        4 minutes ago       187MB
nso300.gitlab.local/ned-ios/package     5.3.2-student
135789184d9b        4 minutes ago       187MB
nso300.gitlab.local/ned-ios/testnso     5.3.2-student
8f07b25292a2        5 minutes ago       721MB
nso300.gitlab.local/ned-ios/netsim      5.3.2
ea3ef54f71b5        5 minutes ago       1.19GB
nso300.gitlab.local/ned-ios/netsim      5.3.2-student
ea3ef54f71b5        5 minutes ago       1.19GB
nso300.gitlab.local/nso300/nso          5.3.2-student
20a0e3a3eb84        3 hours ago         534MB
nso300.gitlab.local/cisco-nso-base      5.3.2
eded19b9090b        4 hours ago         534MB
nso300.gitlab.local/cisco-nso-base      5.3.2-student
eded19b9090b        4 hours ago         534MB
nso300.gitlab.local/cisco-nso-dev       5.3.2
d97b14f50ee8        4 hours ago         1GB
nso300.gitlab.local/cisco-nso-dev       5.3.2-student
d97b14f50ee8        4 hours ago         1GB
debian                                  buster
1b686a95ddbf        4 weeks ago         114MB
student@student-vm:~/neds/ned-nx$
```

Step 11

Include the images as dependencies to the nso300 project. Enter the *nso300/includes* directory and create an image dependency for each NED type. The contents of the files should list the image name + tag for each NED.

By including these images with the main NSO system image, the packages are added to the NSO running folder when building or rebuilding the image.

```
student@student-vm:~/neds/ned-asa$ cd ~/nso300/includes/
student@student-vm:~/nso300/includes$ echo "${NSO_IMAGE_PATH}ned-ios/
package:${NSO_VERSION}" >> ned-ios
student@student-vm:~/nso300/includes$ echo "${NSO_IMAGE_PATH}ned-iosxr/
package:${NSO_VERSION}" >> ned-iosxr
student@student-vm:~/nso300/includes$ echo "${NSO_IMAGE_PATH}ned-nx/
package:${NSO_VERSION}" >> ned-nx
student@student-vm:~/nso300/includes$ echo "${NSO_IMAGE_PATH}ned-asa/
package:${NSO_VERSION}" >> ned-asa
```

Step 12

List the contents of the *includes* directory. You should now have four files there.

```
student@student-vm:~/nso300/includes$ ls
```

```
ned-asa  ned-ios  ned-iosxr  ned-nx
student@student-vm:~/nso300/includes$
```

Step 13

Start the NED netsim images as a part of the test environment. You can do this by editing the Makefile of the nso300 project.

```
student@student-vm:~/nso300/includes$ cd ..
student@student-vm:~/nso300 $ vi Makefile
```

Step 14

Add an IOS device that is named CE11. You add the device by creating a Docker container from the devices' NED image using the **docker run** command.

```
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"
        docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)

testenv-test:
        @echo "\n== Running tests"
        @echo "TODO: Fill in your tests here"
```

Make sure you only use TAB spaces when indenting commands inside Makefiles. If you use normal whitespaces, you encounter errors. You can learn more about the **docker run** command from the following link https://docs.docker.com/engine/reference/run.

Step 15

Add the additional devices. Create a total of eight Cisco IOS devices (CE11, CE12, CE21, CE31, PE11, PE12, PE31, and SW31), one Cisco IOS XR NetSim device (PE21), and one Cisco NX device (SW32). Include an extra Cisco ASA device, named ASA41.

Because the NSO project skeleton provides a **--network** parameter within ${DOCKER_ARGS} to the **docker run** command, all the containers are in the same Docker network and are therefore able to communicate with each other.

```
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"
        docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE12 --network-alias CE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE21 --network-alias CE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE31 --network-alias CE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE11 --network-alias PE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE12 --network-alias PE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE21 --network-alias PE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-iosxr/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE31 --network-alias PE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW31 --network-alias SW31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW32 --network-alias SW32
$(DOCKER_ARGS) $(IMAGE_PATH)ned-nx/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-ASA41 --network-alias ASA41
$(DOCKER_ARGS) $(IMAGE_PATH)ned-asa/netsim:$(DOCKER_TAG)

testenv-test:
        @echo "\n== Running tests"
        @echo "TODO: Fill in your tests here"
```

Step 16

Save the file and exit the file editor.

Step 17

Rebuild the test image to include the NED packages.

```
student@student-vm:~/nso300 $ make build

< … Output omitted … >

Successfully built 36dffa1e9b9f
Successfully tagged nso300.gitlab.local/nso300/nso:5.3.2-student
student@student-vm:~/nso300 $
```

Step 18

Stop and start the Docker environment again.

Stopping and starting the NSO Docker environment deletes and recreates the Docker containers. This action means that any CDB configuration is lost, because the NSO System container will be deleted. Your packages remain, though, because they are not located in that container, but merely mounted.

```
student@student-vm:~/nso300$ make testenv-stop
docker ps -aq --filter label=testenv-nso300-5.3.2-student | xargs --no-
run-if-empty docker rm -vf
c921186ec658
docker network rm testenv-nso300-5.3.2-student
testenv-nso300-5.3.2-student
student@student-vm:~/nso300$ make testenv-start
docker network inspect testenv-nso300-5.3.2-student >/dev/null 2>&1 ||
docker network create testenv-nso300-5.3.2-student
aaca04d0c9e8d5edd9eda585cfb0c16885d9e73e910feeba4bc065f4ab751943
docker run -td --name testenv-nso300-5.3.2-student-nso --network-alias
nso --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student --label nidtype=nso --volume /var/opt/ncs/packages
-e ADMIN_PASSWORD=NsoDocker1337 ${NSO_EXTRA_ARGS} nso300.gitlab.local/
nso300/nso:5.3.2-student
4e553741a013c5bba4719456294040961d1cfcf6a93eaf09568506af83c782d1
make testenv-start-extra
make[1]: Entering directory '/home/student/nso300'

== Starting repository specific testenv
docker run -td --name testenv-nso300-5.3.2-student-CE11 --network-alias
CE11 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
0875a55f9000d3a9c3252d07a7c43d58d60173855625b49fbc6b505b76222aed
docker run -td --name testenv-nso300-5.3.2-student-CE12 --network-alias
CE12 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
e92acb795163ebfd9f5dfa28a01c355f5674157812965ac7c97946a096323f86
docker run -td --name testenv-nso300-5.3.2-student-CE21 --network-alias
CE21 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
8f4f5c676f088667669f0370b376b9bde4790cc30dc44f53ab2c5a8b6eb63eed
docker run -td --name testenv-nso300-5.3.2-student-CE31 --network-alias
CE31 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
2ef4fd0567f10bb6a8878440b8ab11f952a4c5e1056419c3fd2a958616943e02
docker run -td --name testenv-nso300-5.3.2-student-PE11 --network-alias
PE11 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
cb724f58bd7577a6b0ed6171adcc8a132847b905922c866bcfe510f04510c630
docker run -td --name testenv-nso300-5.3.2-student-PE12 --network-alias
PE12 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
a6b8bf2fb7e71a17ca41935dabc4d9c004b39777874c0e1699ce14d9f6c0f07c
docker run -td --name testenv-nso300-5.3.2-student-PE21 --network-alias
PE21 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-iosxr/netsim:5.3.2-student
9c96472944ecae56c09c7ac0b4802af59f4ee0f9fc6b4f9e404c1e29a8c17eaa
docker run -td --name testenv-nso300-5.3.2-student-PE31 --network-alias
PE31 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
aa7807c51b8a9d8faadd7b1190baabab543de5663951a5c9ea09fa6b0da7e41f
docker run -td --name testenv-nso300-5.3.2-student-SW31 --network-alias
SW31 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-ios/netsim:5.3.2-student
```

```
0362d3bcf060b5b6c801f901e8bdedac3be74d26156ec07a0d1e80693494f9b6
docker run -td --name testenv-nso300-5.3.2-student-SW32 --network-alias
SW32 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-nx/netsim:5.3.2-student
f268ce76df1575798fdd0e053a52ce9129811821a01811007754eef98e4c359a
docker run -td --name testenv-nso300-5.3.2-student-ASA41 --network-
alias ASA41 --network testenv-nso300-5.3.2-student --label testenv-
nso300-5.3.2-student nso300.gitlab.local/ned-asa/netsim:5.3.2-student
31ea0c591badf0308e818aedcc32f498fa662e645dbca423e3d0d995e3a01aa7
make[1]: Leaving directory '/home/student/nso300'
docker exec -t testenv-nso300-5.3.2-student-nso bash -lc 'ncs --wait-
started 600'
student@student-vm:~/nso300$
```

If you forget to stop the test environment before starting it again, a conflict in container or network names occurs.

Step 19

Enter the NSO System container shell and ping the CE11 device, which is in the same internal Docker network.

The netsim images in your Docker environment are now online; however, they are not yet added to NSO.

```
student@student-vm:~/nso300$ make testenv-shell
docker exec -it testenv-nso300-5.3.2-student-nso bash -l
root@4e553741a013:/# ping CE11
PING CE11 (172.27.0.3) 56(84) bytes of data.
64 bytes from testenv-nso300-5.3.2-student-CE11.testenv-nso300-5.3.2-
student (172.27.0.3): icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from testenv-nso300-5.3.2-student-CE11.testenv-nso300-5.3.2-
student (172.27.0.3): icmp_seq=2 ttl=64 time=0.082 ms
64 bytes from testenv-nso300-5.3.2-student-CE11.testenv-nso300-5.3.2-
student (172.27.0.3): icmp_seq=3 ttl=64 time=0.066 ms
64 bytes from testenv-nso300-5.3.2-student-CE11.testenv-nso300-5.3.2-
student (172.27.0.3): icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from testenv-nso300-5.3.2-student-CE11.testenv-nso300-5.3.2-
student (172.27.0.3): icmp_seq=5 ttl=64 time=0.068 ms
--- CE11 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 89ms
rtt min/avg/max/mdev = 0.063/0.074/0.095/0.016 ms
root@4e553741a013:/#
```

Step 20

Exit the NSO System container shell and open and display the *~/lab/devices.xml* file.

The *devices.xml* file contains the CDB configuration for the NetSim images. It can be compiled by manually exporting each NetSim device configuration but it is provided in this case.

```
root@4e553741a013:/# exit
```

```
student@student-vm:~/nso300$ cat ~/lab/devices.xml
<devices xmlns="http://tail-f.com/ns/ncs">
    <authgroups>
        <group>
            <name>netsim</name>
            <default-map>
                <remote-name>admin</remote-name>
                <remote-password>admin</remote-password>
            </default-map>
        </group>
    </authgroups>
    <device>
        <name>CE11</name>
        <address>CE11</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-cli-6.54="http://tail-f.com/ns/
ned-id/cisco-ios-cli-6.54">cisco-ios-cli-6.54:cisco-ios-cli-6.54</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
    <device>
        <name>CE12</name>
        <address>CE12</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-cli-6.54="http://tail-f.com/ns/
ned-id/cisco-ios-cli-6.54">cisco-ios-cli-6.54:cisco-ios-cli-6.54</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
    <device>
        <name>CE21</name>
        <address>CE21</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-cli-6.54="http://tail-f.com/ns/
ned-id/cisco-ios-cli-6.54">cisco-ios-cli-6.54:cisco-ios-cli-6.54</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
    <device>
        <name>CE31</name>
        <address>CE31</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-cli-6.54="http://tail-f.com/ns/
ned-id/cisco-ios-cli-6.54">cisco-ios-cli-6.54:cisco-ios-cli-6.54</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
    <device>
        <name>PE11</name>
        <address>PE11</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-cli-6.54="http://tail-f.com/ns/
ned-id/cisco-ios-cli-6.54">cisco-ios-cli-6.54:cisco-ios-cli-6.54</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
    <device>
        <name>PE12</name>
        <address>PE12</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-cli-6.54="http://tail-f.com/ns/
ned-id/cisco-ios-cli-6.54">cisco-ios-cli-6.54:cisco-ios-cli-6.54</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
    <device>
        <name>PE21</name>
        <address>PE21</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-iosxr-cli-7.26="http://tail-f.com/
ns/ned-id/cisco-iosxr-cli-7.26">cisco-iosxr-cli-7.26:cisco-iosxr-
cli-7.26</ned-id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
    <device>
        <name>PE31</name>
        <address>PE31</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-cli-6.54="http://tail-f.com/ns/
ned-id/cisco-ios-cli-6.54">cisco-ios-cli-6.54:cisco-ios-cli-6.54</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
     <device>
        <name>SW31</name>
        <address>SW31</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                                <ned-id xmlns:cisco-ios-
cli-6.54="http://tail-f.com/ns/ned-id/cisco-ios-cli-6.54">cisco-ios-
cli-6.54:cisco-ios-cli-6.54</ned-id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
     <device>
        <name>SW32</name>
        <address>SW32</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-ios-nx-5.15="http://tail-f.com/ns/
ned-id/cisco-nx-cli-5.15">cisco-nx-cli-5.15:cisco-nx-cli-5.15</ned-id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
     </device>
     <device>
        <name>ASA41</name>
        <address>ASA41</address>
        <authgroup>netsim</authgroup>
        <device-type>
            <cli>
                <ned-id xmlns:cisco-asa-cli-6.10="http://tail-f.com/ns/
ned-id/cisco-asa-cli-6.10">cisco-asa-cli-6.10:cisco-asa-cli-6.10</ned-
id>
            </cli>
        </device-type>
        <state>
            <admin-state>unlocked</admin-state>
        </state>
```

```
    </device>
</devices>
```

You can also import the devices by hand through the NSO CLI.

Step 21

Copy the *devices.xml* file to *~/nso300/extra-files* directory. This directory is used to include extra files with the NSO system image. These files are placed into the root of the resulting image.

```
student@student-vm:~/nso300$ cp ~/lab/devices.xml extra-files/
```

Step 22

Stop, build, and start the Docker environment again for the extra files to appear in the NSO System container.

```
student@student-vm:~/nso300$ make testenv-stop

< … Output Omitted … >

student@student-vm:~/nso300$ make build

< … Output Omitted … >

student@student-vm:~/nso300$ make testenv-start

< … Output Omitted … >
```

Step 23

Open the Docker environment's Makefile and add a *testenv-configure* target. This target will be used to import and configure devices in your Docker environment.

```
student@student-vm:~/nso300$ vi Makefile
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"
        docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE12 --network-alias CE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE21 --network-alias CE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
```

```
        docker run -td --name $(CNT_PREFIX)-CE31 --network-alias CE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE11 --network-alias PE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE12 --network-alias PE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE21 --network-alias PE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-iosxr/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE31 --network-alias PE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW31 --network-alias SW31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW32 --network-alias SW32
$(DOCKER_ARGS) $(IMAGE_PATH)ned-nx/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-ASA41 --network-alias ASA41
$(DOCKER_ARGS) $(IMAGE_PATH)ned-asa/netsim:$(DOCKER_TAG)

testenv-configure:
        @echo "Configuring test environment"

testenv-test:
        @echo "\n== Running tests"
        @echo "TODO: Fill in your tests here"
```

Step 24

Add the NSO commands to import device configuration, fetch SSH keys, and synchronize the configuration from the devices. You can either use the Cisco- or Juniper-style CLI, by using the **testenv-runcmdC** or **testenv-runcmdJ** targets respectively to execute commands inside the NSO System container. The commands should simulate the input that the user creates through the NSO CLI. You can use the new-line character **\n** to simulate the user pressing the Enter key.

```
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"
        docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE12 --network-alias CE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE21 --network-alias CE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE31 --network-alias CE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE11 --network-alias PE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE12 --network-alias PE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE21 --network-alias PE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-iosxr/netsim:$(DOCKER_TAG)
```

```
        docker run -td --name $(CNT_PREFIX)-PE31 --network-alias PE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW31 --network-alias SW31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW32 --network-alias SW32
$(DOCKER_ARGS) $(IMAGE_PATH)ned-nx/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-ASA41 --network-alias ASA41
$(DOCKER_ARGS) $(IMAGE_PATH)ned-asa/netsim:$(DOCKER_TAG)

testenv-configure:
        @echo "Configuring test environment"
        $(MAKE) testenv-runcmdJ CMD="configure \n load merge /
devices.xml \n commit \n exit"
        $(MAKE) testenv-runcmdJ CMD="request devices fetch-ssh-host-
keys"
        $(MAKE) testenv-runcmdJ CMD="request devices sync-from"

testenv-test:
        @echo "\n== Running tests"
        @echo "TODO: Fill in your tests here"
```

Remember to make sure you only use TAB spaces when indenting commands inside Makefiles. If you use normal whitespaces, you will encounter errors.

For more information on targets, open and study the *nidsystem.mk* file.

Step 25

Save and exit the file.

Step 26

Import the devices with the **make testenv-configure** command that you created in the Makefile. Make sure that the devices end up in-sync.

```
student@student-vm:~/nso300$ make testenv-configure
Configuring test environment
make testenv-runcmdJ CMD="configure\nload merge /
devices.xml\ncommit\nexit"
make[1]: Entering directory '/home/student/nso300'
docker exec -t testenv-nso300-5.3.2-student-nso bash -lc 'echo -e
"configure\nload merge /devices.xml\ncommit\nexit" | ncs_cli -Ju admin'
Commit complete.
make[1]: Leaving directory '/home/student/nso300'
make testenv-runcmdJ CMD="request devices fetch-ssh-host-keys"
make[1]: Entering directory '/home/student/nso300'
docker exec -t testenv-nso300-5.3.2-student-nso bash -lc 'echo -e
"request devices fetch-ssh-host-keys" | ncs_cli -Ju admin'
fetch-result {
    device ASA41
    result updated
```

```
        fingerprint {
            algorithm ssh-rsa
            value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
        }
    }
    fetch-result {
        device CE11
        result updated
        fingerprint {
            algorithm ssh-rsa
            value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
        }
    }
    fetch-result {
        device CE12
        result updated
        fingerprint {
            algorithm ssh-rsa
            value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
        }
    }
    fetch-result {
        device CE21
        result updated
        fingerprint {
            algorithm ssh-rsa
            value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
        }
    }
    fetch-result {
        device CE31
        result updated
        fingerprint {
            algorithm ssh-rsa
            value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
        }
    }
    fetch-result {
        device PE11
        result updated
        fingerprint {
            algorithm ssh-rsa
            value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
        }
    }
    fetch-result {
        device PE12
        result updated
        fingerprint {
            algorithm ssh-rsa
            value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
        }
    }
    fetch-result {
        device PE21
        result updated
        fingerprint {
            algorithm ssh-rsa
```

```
          value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
      }
  }
  fetch-result {
      device PE31
      result updated
      fingerprint {
          algorithm ssh-rsa
          value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
      }
  }
  fetch-result {
      device SW31
      result updated
      fingerprint {
          algorithm ssh-rsa
          value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
      }
  }
  fetch-result {
      device SW32
      result updated
      fingerprint {
          algorithm ssh-rsa
          value 01:2d:00:cb:09:eb:89:e4:db:f0:8c:c6:32:0d:90:72
      }
  }
  make[1]: Leaving directory '/home/student/nso300'
  make testenv-runcmdJ CMD="request devices sync-from"
  make[1]: Entering directory '/home/student/nso300'
  docker exec -t testenv-nso300-5.3.2-student-nso bash -lc 'echo -e
  "request devices sync-from" | ncs_cli -Ju admin'
  sync-result {
      device ASA41
      result true
  }
  sync-result {
      device CE11
      result true
  }
  sync-result {
      device CE12
      result true
  }
  sync-result {
      device CE21
      result true
  }
  sync-result {
      device CE31
      result true
  }
  sync-result {
      device PE11
      result true
  }
  sync-result {
      device PE12
```

```
    result true
}
sync-result {
    device PE21
    result true
}
sync-result {
    device PE31
    result true
}
sync-result {
    device SW31
    result true
}
sync-result {
    device SW32
    result true
}
make[1]: Leaving directory '/home/student/nso300'
student@student-vm:~/nso300$
```

Step 27

Enter the NSO System container, switch the CLI type, and list the devices.

All eleven devices should be present.

```
student@student-vm:~/nso300$ make testenv-cli
docker exec -it testenv-nso300-5.3.2-student-nso bash -lc 'ncs_cli -u
admin'

admin connected from 127.0.0.1 using console on 1712c0a242e5
admin@ncs> switch cli
admin@ncs# show devices list
NAME   ADDRESS  DESCRIPTION  NED ID               ADMIN STATE
--------------------------------------------------------------
ASA41  ASA41    -            cisco-asa-cli-6.10    unlocked
CE11   CE11     -            cisco-ios-cli-6.54    unlocked
CE12   CE12     -            cisco-ios-cli-6.54    unlocked
CE21   CE21     -            cisco-ios-cli-6.54    unlocked
CE31   CE31     -            cisco-ios-cli-6.54    unlocked
PE11   PE11     -            cisco-ios-cli-6.54    unlocked
PE12   PE12     -            cisco-ios-cli-6.54    unlocked
PE21   PE21     -            cisco-iosxr-cli-7.26  unlocked
PE31   PE31     -            cisco-ios-cli-6.54    unlocked
SW31   SW31     -            cisco-ios-cli-6.54    unlocked
SW32   SW32     -            cisco-nx-cli-5.15     unlocked
admin@ncs#
```

Step 28

Exit the NSO System container.

```
admin@ncs# exit
```

Activity Verification

You have completed this task when you attain these results:

- You successfully created and imported the netsim devices using Docker images.

Task 3: Develop an NSO Package

In this task, you will learn how to develop an NSO package using NSO in Docker.

Activity

Complete these steps:

Step 1

Create and run a development container using the **make dev-shell** command.

This creates a container that contains an ncsc YANG compiler and a Java compiler, has **ncs** commands added to the path and includes other useful tools that you can use for NSO package development.

```
student@student-vm:~/nso300$ make dev-shell
docker run -it -v $(pwd):/src nso300.gitlab.local/cisco-nso-dev:5.3.2
root@98e9a053bede:/#
```

Step 2

Navigate to the */src/packages* folder.

```
root@98e9a053bede:/# cd src/packages/
root@98e9a053bede:/src/packages#
```

Step 3

Create a template-based **hostname** package using the **ncs-make-package** command.

```
root@98e9a053bede:/src/packages# ncs-make-package --service-skeleton
template hostname
```

Step 4

List the contents of the current folder. It should contain the **hostname** package.

```
root@98e9a053bede:/src/packages# ls
hostname
```

Step 5

Change the ownership of the package to a non-root user.

Owner with the UID 1000 is the user 'student' in this case.

```
root@98e9a053bede:/src/packages# chown -Rv 1000:1000 hostname
changed ownership of 'hostname/test/internal/Makefile' from root:root
to 1000:1000
changed ownership of 'hostname/test/internal/lux/basic/Makefile' from
root:root to 1000:1000
changed ownership of 'hostname/test/internal/lux/basic/run.lux' from
root:root to 1000:1000
changed ownership of 'hostname/test/internal/lux/basic' from root:root
to 1000:1000
changed ownership of 'hostname/test/internal/lux/Makefile' from
root:root to 1000:1000
changed ownership of 'hostname/test/internal/lux' from root:root to
1000:1000
changed ownership of 'hostname/test/internal' from root:root to
1000:1000
changed ownership of 'hostname/test/Makefile' from root:root to
1000:1000
changed ownership of 'hostname/test' from root:root to 1000:1000
changed ownership of 'hostname/templates/hostname-template.xml' from
root:root to 1000:1000
changed ownership of 'hostname/templates' from root:root to 1000:1000
changed ownership of 'hostname/src/Makefile' from root:root to
1000:1000
changed ownership of 'hostname/src/yang/hostname.yang' from root:root
to 1000:1000
changed ownership of 'hostname/src/yang' from root:root to 1000:1000
changed ownership of 'hostname/src' from root:root to 1000:1000
changed ownership of 'hostname/package-meta-data.xml' from root:root to
1000:1000
changed ownership of 'hostname' from root:root to 1000:1000
```

Step 6

Exit the development container.

```
root@98e9a053bede:/src/packages# exit
logout
student@student-vm:~/nso300$
```

Step 7

List the contents of the *packages* directory.

The hostname package is present here too. This is because the *packages* directory is bind mounted to the NSO Docker development container.

```
student@student-vm:~/nso300$ ls packages
hostname
student@student-vm:~/nso300$
```

Step 8

Enter the NSO CLI in the NSO system container and switch the CLI type.

```
student@student-vm:~/nso300$ make testenv-cli
docker exec -it testenv-nso300-5.3.2-student-nso bash -lc 'ncs_cli -u
admin'

admin connected from 127.0.0.1 using console on 1712c0a242e5
admin@ncs> switch cli
admin@ncs#
```

Step 9

Configure the hostname of one Cisco-IOS, Cisco-IOSXR, Cisco-NX, and Cisco ASA device.

```
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# devices device CE11 config hostname
CE11.nso300.local
admin@ncs(config-config)# top
admin@ncs(config)# devices device PE21 config hostname
PE21.nso300.local
admin@ncs(config-config)# top
admin@ncs(config)# devices device SW32 config hostname
SW32.nso300.local
admin@ncs(config-config)# top
admin@ncs(config)# devices device ASA41 config hostname
ASA41.nso300.local
admin@ncs(config-config)# top
```

Step 10

Make a dry run of the commit and save the configuration output to clipboard or to a text editor. This configuration will be used to create a service template.

```
admin@ncs(config-config)# commit dry-run outformat xml
result-xml {
    local-node {
        data <devices xmlns="http://tail-f.com/ns/ncs">
               <device>
                 <name>ASA41</name>
                 <config>
                   <hostname xmlns="http://cisco.com/ned/
asa">ASA41.nso300.local</hostname>
```

```
                  </config>
                </device>
                <device>
                  <name>CE11</name>
                  <config>
                    <hostname xmlns="urn:ios">CE11.nso300.local</
hostname>
                  </config>
                </device>
                <device>
                  <name>PE21</name>
                  <config>
                    <hostname xmlns="http://tail-f.com/ned/cisco-ios-
xr">PE21.nso300.local</hostname>
                  </config>
                </device>
                <device>
                  <name>SW32</name>
                  <config>
                    <hostname xmlns="http://tail-f.com/ned/cisco-
nx">SW32.nso300.local</hostname>
                  </config>
                </device>
              </devices>
    }
}
```

Step 11

Exit the NSO CLI and NSO System container.

```
admin@ncs(config)# abort
admin@ncs# exit
student@student-vm:~/nso300$
```

Step 12

Open the packages/hostname/src/yang/hostname.yang file.

```
student@student-vm:~/nso300$ vi packages/hostname/src/yang/
hostname.yang
```

This is how the YANG model should appear when you open it for the first time.

```
module hostname {
  namespace "http://com/example/hostname";
  prefix hostname;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-ncs {
    prefix ncs;
  }

  list hostname {
    key name;

    uses ncs:service-data;
    ncs:servicepoint "hostname";

    leaf name {
      type string;
    }

    // may replace this with other ways of refering to the devices.
    leaf-list device {
      type leafref {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }

    // replace with your own stuff here
    leaf dummy {
      type inet:ipv4-address;
    }
  }
}
```

Step 13

Remove the dummy code and the name leaf, import the tailf-common module, and add a hostname leaf.

```
module hostname {
  namespace "http://com/example/hostname";
  prefix hostname;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-ncs {
    prefix ncs;
  }
  import tailf-common {
    prefix tailf;
  }

  list hostname {
    key name;

    uses ncs:service-data;
    ncs:servicepoint "hostname";

    leaf-list device {
      type leafref {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }

    leaf hostname {
      tailf:info "Device hostname";
      type string;
    }

  }
}
```

Step 14

Change the device YANG type from leaf-list to a leaf and make it the service key.

```
module hostname {
  namespace "http://com/example/hostname";
  prefix hostname;

  import ietf-inet-types {
    prefix inet;
  }
  import tailf-ncs {
    prefix ncs;
  }
  import tailf-common {
    prefix tailf;
  }

  list hostname {
    key device;

    uses ncs:service-data;
    ncs:servicepoint "hostname";

    leaf device {
      type leafref {
        path "/ncs:devices/ncs:device/ncs:name";
      }
    }

    leaf hostname {
      tailf:info "Device hostname";
      type string;
    }

  }
}
```

Step 15

Save and exit the file.

Step 16

Open the packages/hostname/template/hostname-template.xml file.

```
student@student-vm:~/nso300$ vi packages/hostname/templates/hostname-
template.xml
```

This is how the file should appear when you open it for the first time.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="hostname">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <!--
          Select the devices from some data structure in the service
          model. In this skeleton the devices are specified in a leaf-
list.
          Select all devices in that leaf-list:
      -->
      <name>{/device}</name>
      <config>
        <!--
            Add device-specific parameters here.
            In this skeleton the service has a leaf "dummy"; use that
            to set something on the device e.g.:
            <ip-address-on-device>{/dummy}</ip-address-on-device>
        -->
      </config>
    </device>
  </devices>
</config-template>
```

Step 17

Remove the comments and replace the device configuration with the configuration produced by the **dry-run** of the commit a few steps back.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="hostname">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>ASA41</name>
      <config>
        <hostname xmlns="http://cisco.com/ned/asa">ASA41.nso300.local</
hostname>
      </config>
    </device>
    <device>
      <name>CE11</name>
      <config>
        <hostname xmlns="urn:ios">CE11.nso300.local</hostname>
      </config>
    </device>
    <device>
      <name>PE21</name>
      <config>
        <hostname xmlns="http://tail-f.com/ned/cisco-ios-
xr">PE21.nso300.local</hostname>
      </config>
    </device>
    <device>
      <name>SW32</name>
      <config>
        <hostname xmlns="http://tail-f.com/ned/cisco-
nx">SW32.nso300.local</hostname>
      </config>
    </device>
  </devices>
</config-template>
```

Step 18

Replace the hardcoded values with variables used in the YANG model.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="hostname">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{/device}</name>
      <config>
        <hostname xmlns="http://cisco.com/ned/asa">{/hostname}</
hostname>
      </config>
    </device>
    <device>
      <name>{/device}</name>
      <config>
        <hostname xmlns="urn:ios">{/hostname}</hostname>
      </config>
    </device>
    <device>
      <name>{/device}</name>
      <config>
        <hostname xmlns="http://tail-f.com/ned/cisco-ios-xr">{/
hostname}</hostname>
      </config>
    </device>
    <device>
      <name>{/device}</name>
      <config>
        <hostname xmlns="http://tail-f.com/ned/cisco-nx">{/hostname}</
hostname>
      </config>
    </device>
  </devices>
</config-template>
```

Step 19

Optimize the template since most of the elements are duplicate. Keep only one device element in which the hostname is set depending on the namespace.

```
<config-template xmlns="http://tail-f.com/ns/config/1.0"
                 servicepoint="hostname">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{/device}</name>
      <config>
        <hostname xmlns="http://cisco.com/ned/asa">{/hostname}</
hostname>
        <hostname xmlns="urn:ios">{/hostname}</hostname>
        <hostname xmlns="http://tail-f.com/ned/cisco-ios-xr">{/
hostname}</hostname>
        <hostname xmlns="http://tail-f.com/ned/cisco-nx">{/hostname}</
hostname>
      </config>
    </device>
  </devices>
</config-template>
```

Step 20

Save the file and exit the file editor.

Step 21

Rebuild the package using the **make testenv-build** command. This command can figure out what (if any) kinds of changes have been done to the packages. It then performs a package reload or just a package re-deploy accordingly.

```
student@student-vm:~/nso300$ make testenv-build

< … Output Omitted … >

-- Reloading packages for NSO testenv-nso300-5.3.2-student-nso

>>> System upgrade is starting.
>>> Sessions in configure mode must exit to operational mode.
>>> No configuration changes can be performed until upgrade has
completed.
>>> System upgrade has completed successfully.
reload-result {
    package cisco-asa-cli-6.10
    result true
}
reload-result {
    package cisco-ios-cli-6.54
    result true
}
reload-result {
    package cisco-iosxr-cli-7.26
    result true
}
reload-result {
    package cisco-nx-cli-5.15
    result true
}
reload-result {
    package hostname
    result true
}
student@student-vm:~/nso300$
```

Instead of using the **make testenv-build** command, you can reload the packages by recompiling them using the **make** command from the *<package>/src* directory and then reloading them by using the **reload packages** command from the NSO CLI.

Step 22

Enter the NSO CLI in NSO System container and switch the CLI type.

```
student@student-vm:~/nso300$ make testenv-cli
docker exec -it testenv-nso300-5.3.2-student-nso bash -lc 'ncs_cli -u
admin'

admin connected from 127.0.0.1 using console on 1712c0a242e5
admin@ncs> switch cli
admin@ncs#
```

Step 23

Configure an instance of the hostname service for the PE21 device and verify the dry-run configuration.

```
admin@ncs# config
Entering configuration mode terminal
admin@ncs(config)# hostname PE21 hostname PE21.nso300.local
admin@ncs(config-hostname-PE21)# top
admin@ncs(config)# commit dry-run
cli {
    local-node {
        data  devices {
                  device PE21 {
                      config {
             +            hostname PE21.nso300.local;
                      }
                  }
              }
             +hostname PE21 {
             +    hostname PE21.nso300.local;
             +}
    }
}
admin@ncs(config)#
```

Step 24

Commit the transaction and exit the NSO CLI.

```
admin@ncs(config)# commit
Commit complete.
admin@ncs(config)# exit
admin@ncs# exit
student@student-vm:~/nso300$
```

Step 25

Enter the NSO System Linux shell with the **make testenv-shell** command.

```
student@student-vm:~/nso300$ make testenv-shell
docker exec -it testenv-nso300-5.3.2-student-nso bash -l
root@1712c0a242e5:/#
```

Step 26

Connect to the PE21 netsim device. Use the **ssh** command with the username **admin** and password **admin** to establish an SSH connection.

```
root@1712c0a242e5:/# ssh admin@PE21
The authenticity of host 'pe21 (172.29.0.9)' can't be established.
RSA key fingerprint is
SHA256:wu9ZN8zMo4GMCMvcysRacc/67ECvC8TM3pmyyl2JyyQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'pe21,172.29.0.9' (RSA) to the list of known
hosts.
admin@pe21's password:

admin connected from 172.29.0.2 using ssh on 762447fefa19
dev#
```

Step 27

Verify that the configuration (hostname) has been successfully applied to the device.

```
dev# show running-config hostname
hostname PE21.nso300.local
dev#
```

Step 28

Terminate the session and exit the NSO System container.

```
dev# exit
Connection to pe21 closed.
```

```
root@1712c0a242e5:/# exit
logout
student@student-vm:~/nso300$
```

Activity Verification

You have completed this task when you attain these results:

- You have successfully developed a package and deployed a service instance using NSO in Docker.

Task 4: Test an NSO Package

In this task, you will create an automated test procedure for an NSO package using NSO in Docker. The test procedure is a result of how quickly this NSO Docker environment is set up and provides you with a simple but powerful framework for testing your NSO packages.

Activity

Complete these steps:

Step 1

Open the Makefile for the nso300 project.

```
student@student-vm:~/nso300$ vi Makefile
```

The following output shows how the file should appear when you open it. In this activity, you will edit the testenv-test target to create a simple automated testing procedure.

```
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"
        docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE12 --network-alias CE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE21 --network-alias CE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE31 --network-alias CE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE11 --network-alias PE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE12 --network-alias PE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
```

```
	docker run -td --name $(CNT_PREFIX)-PE21 --network-alias PE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-iosxr/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-PE31 --network-alias PE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-SW31 --network-alias SW31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-SW32 --network-alias SW32
$(DOCKER_ARGS) $(IMAGE_PATH)ned-nx/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-ASA41 --network-alias ASA41
$(DOCKER_ARGS) $(IMAGE_PATH)ned-asa/netsim:$(DOCKER_TAG)

testenv-configure:
	@echo "Configuring test environment"
	$(MAKE) testenv-runcmdJ CMD="configure \n load merge /
devices.xml \n commit \n exit"
	$(MAKE) testenv-runcmdJ CMD="request devices fetch-ssh-host-
keys"
	$(MAKE) testenv-runcmdJ CMD="request devices sync-from"

testenv-test:
	@echo "\n== Running tests"
	@echo "TODO: Fill in your tests here"
```

Step 2

Create a set of commands that will deploy an instance of the hostname service on the SW32 device and verify that the hostname has been set. To verify this, you can grep the CLI output by piping the CLI output and searching it with the **grep** command.

```
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
	@echo "\n== Starting repository specific testenv"
	docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-CE12 --network-alias CE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-CE21 --network-alias CE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-CE31 --network-alias CE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-PE11 --network-alias PE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-PE12 --network-alias PE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-PE21 --network-alias PE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-iosxr/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-PE31 --network-alias PE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-SW31 --network-alias SW31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
	docker run -td --name $(CNT_PREFIX)-SW32 --network-alias SW32
```

```
$(DOCKER_ARGS) $(IMAGE_PATH)ned-nx/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-ASA41 --network-alias ASA41
$(DOCKER_ARGS) $(IMAGE_PATH)ned-asa/netsim:$(DOCKER_TAG)

testenv-configure:
        @echo "Configuring test environment"
        $(MAKE) testenv-runcmdJ CMD="configure \n load merge /
devices.xml \n commit \n exit"
        $(MAKE) testenv-runcmdJ CMD="request devices fetch-ssh-host-
keys"
        $(MAKE) testenv-runcmdJ CMD="request devices sync-from"

testenv-test:
        @echo "\n== Running tests"
        @echo "Hostname package tests"
        $(MAKE) testenv-runcmdJ CMD="configure \n set hostname SW32
hostname SW32.nso300.local \n commit"
        $(MAKE) testenv-runcmdJ CMD="show configuration devices device
SW32 config hostname" | grep "SW32.nso300.local"
```

If your tests require some pre-existing device configuration, you can always use the **ncs_load** command to load some configuration into CDB before running the tests.

Step 3

Add an extra command that ensures that the hostname is not set beforehand. You can do this action by counting the number of occurrences that **grep** finds by using the **-c** flag and making sure that number is 0.

```
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"
        docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE12 --network-alias CE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE21 --network-alias CE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE31 --network-alias CE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE11 --network-alias PE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE12 --network-alias PE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE21 --network-alias PE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-iosxr/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE31 --network-alias PE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
```

```
        docker run -td --name $(CNT_PREFIX)-SW31 --network-alias SW31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW32 --network-alias SW32
$(DOCKER_ARGS) $(IMAGE_PATH)ned-nx/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-ASA41 --network-alias ASA41
$(DOCKER_ARGS) $(IMAGE_PATH)ned-asa/netsim:$(DOCKER_TAG)

testenv-configure:
        @echo "Configuring test environment"
        $(MAKE) testenv-runcmdJ CMD="configure \n load merge /
devices.xml \n commit \n exit"
        $(MAKE) testenv-runcmdJ CMD="request devices fetch-ssh-host-
keys"
        $(MAKE) testenv-runcmdJ CMD="request devices sync-from"

testenv-test:
        @echo "\n== Running tests"
        @echo "Hostname package tests"
        $(MAKE) testenv-runcmdJ CMD="show configuration devices device
SW32 config hostname" | grep -c "SW32.nso300.local" | grep 0
        $(MAKE) testenv-runcmdJ CMD="configure \n set hostname SW32
hostname SW32.nso300.local \n commit"
        $(MAKE) testenv-runcmdJ CMD="show configuration devices device
SW32 config hostname" | grep "SW32.nso300.local"
```

Step 4

Finally, add a set of commands to clean up the configuration after the test has been completed. Ensure that the hostname is not set anymore at the end with the command from the previous step.

```
export NSO_VERSION=5.3.2
export NSO_IMAGE_PATH=nso300.gitlab.local/
export IMAGE_PATH=nso300.gitlab.local/

include nidsystem.mk

testenv-start-extra:
        @echo "\n== Starting repository specific testenv"
        docker run -td --name $(CNT_PREFIX)-CE11 --network-alias CE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE12 --network-alias CE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE21 --network-alias CE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-CE31 --network-alias CE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE11 --network-alias PE11
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE12 --network-alias PE12
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE21 --network-alias PE21
$(DOCKER_ARGS) $(IMAGE_PATH)ned-iosxr/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-PE31 --network-alias PE31
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW31 --network-alias SW31
```

```
$(DOCKER_ARGS) $(IMAGE_PATH)ned-ios/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-SW32 --network-alias SW32
$(DOCKER_ARGS) $(IMAGE_PATH)ned-nx/netsim:$(DOCKER_TAG)
        docker run -td --name $(CNT_PREFIX)-ASA41 --network-alias ASA41
$(DOCKER_ARGS) $(IMAGE_PATH)ned-asa/netsim:$(DOCKER_TAG)

testenv-configure:
        @echo "Configuring test environment"
        $(MAKE) testenv-runcmdJ CMD="configure \n load merge /
devices.xml \n commit \n exit"
        $(MAKE) testenv-runcmdJ CMD="request devices fetch-ssh-host-
keys"
        $(MAKE) testenv-runcmdJ CMD="request devices sync-from"

testenv-test:
        @echo "\n== Running tests"
        @echo "Hostname package tests"
        $(MAKE) testenv-runcmdJ CMD="show configuration devices device
SW32 config hostname" | grep -c "SW32.nso300.local" | grep 0
        $(MAKE) testenv-runcmdJ CMD="configure \n set hostname SW32
hostname SW32.nso300.local \n commit"
        $(MAKE) testenv-runcmdJ CMD="show configuration devices device
SW32 config hostname" | grep "SW32.nso300.local"
        $(MAKE) testenv-runcmdJ CMD="configure \n delete hostname SW32
\n commit"
        $(MAKE) testenv-runcmdJ CMD="show configuration devices device
SW32 config hostname" | grep -c "SW32.nso300.local" | grep 0
        @echo "Hostname package test completed!"
```

Step 5

Save the file and exit the file editor.

Step 6

Run the tests by using the **make testenv-test** command.

Verify that the tests have successfully completed. If any of the intermediate commands fail, the execution of Makefile will stop at that point.

```
student@student-vm:~/nso300$ make testenv-test

== Running tests
Hostname package tests
make testenv-runcmdJ CMD="show configuration devices device SW32 config
hostname" | grep -c "SW32.nso300.local" | grep 0
0
make testenv-runcmdJ CMD="configure \n set hostname SW32 hostname
SW32.nso300.local \n commit"
make[1]: Entering directory '/home/student/nso300'
docker exec -t testenv-nso300-5.3.2-student-nso bash -lc 'echo -e
"configure \n set hostname SW32 hostname SW32.nso300.local \n commit" |
ncs_cli -Ju admin'
Commit complete.
make[1]: Leaving directory '/home/student/nso300'
make testenv-runcmdJ CMD="show configuration devices device SW32 config
hostname" | grep "SW32.nso300.local"
```

```
hostname SW32.nso300.local;
make testenv-runcmdJ CMD="configure \n delete hostname SW32 \n commit"
make[1]: Entering directory '/home/student/nso300'
docker exec -t testenv-nso300-5.3.2-student-nso bash -lc 'echo -e
"configure \n delete hostname SW32 \n commit" | ncs_cli -Ju admin'
Commit complete.
make[1]: Leaving directory '/home/student/nso300'
make testenv-runcmdJ CMD="show configuration devices device SW32 config
hostname" | grep -c "SW32.nso300.local" | grep 0
0
Hostname package test completed!
student@student-vm:~/nso300$
```

This is the simplest form of tests you can implement with NSO in Docker. For more advanced tests and more complex services, consider creating a sandbox environment with actual virtual network devices that support end-to-end tests.

Activity Verification

You have completed this task when you attain these results:

- You have successfully designed and executed automated tests for an NSO service using NSO in Docker.