CNaaS-NMS Workshop - 1: Minimal setup

This workshop will guide you to the steps of setting up a very minimal CNaaS-NMS environment without any connected network devices. You will be able to use the API and get a feeling for how to work with the git repositories to generate a configuration, but we will not be installing the configuration on any network devices.

This guide will use docker and docker-compose to get the environment up and running quickly. We will also be creating local git repositories on your machine.

Install required tools: docker, docker-compose, git, curl, jq

Starting the API

Create a new directory named "cnaas", and then create a docker-compose.yaml file with the following contents in it:

docker-compose.yaml

```
---
version: '3.7'
services:
  cnaas_api:
    image: docker.sunet.se/cnaas/api:stable
    ports:
      - 443:1443
    networks:
      - cnaas
    environment:
      - GITREPO_TEMPLATES=/opt/git/cnaas-templates-origin.git
      - GITREPO_SETTINGS=/opt/git/cnaas-settings-origin.git
      - GITREPO_ETC=/opt/git/cnaas-etc-origin.git
      - USERNAME_DHCP_BOOT=admin
      - PASSWORD_DHCP_BOOT=abc123abc123
      - USERNAME_DISCOVERED=admin
      - PASSWORD_DISCOVERED=abc123abc123
      - USERNAME_INIT=admin
      - PASSWORD_INIT=abc123abc123
      - USERNAME_MANAGED=admin
      - PASSWORD_MANAGED=abc123abc123
      - TEMPLATE_SECRET_ADMIN_HASH
    depends_on:
      - "cnaas_postgres"
      - "cnaas_redis"
    volumes:
      - type: volume
        source: cnaas-templates
        target: /opt/cnaas/templates
      - type: volume
        source: cnaas-settings
        target: /opt/cnaas/settings
      - type: volume
        source: cnaas-jwtcert
        target: /opt/cnaas/jwtcert
      - type: volume
        source: cnaas-cacert
        target: /opt/cnaas/cacert
      - type: bind
        source: ./git
        target: /opt/git
  cnaas_postgres:
    image: docker.sunet.se/cnaas/postgres:latest
    volumes:
      - cnaas-postgres-data:/var/lib/postgresql/data
    environment:
      - POSTGRES_USER=cnaas
      - POSTGRES_PASSWORD=cnaas
      - POSTGRES_DB=cnaas
    networks:
      - cnaas
  cnaas_redis:
```

```
    image: redis:latest
    networks:
      - cnaas

networks:
  cnaas:
    driver: bridge
    name: cnaas
    ipam:
      config:
      - subnet: 172.30.0.0/24
    driver_opts:
      com.docker.network.bridge.name: br-cnaas

volumes:
  cnaas-templates:
  cnaas-settings:
  cnaas-postgres-data:
  cnaas-jwtcert:
  cnaas-cacert:
```

From the directory where you saved the docker-compose.yaml file run:

```
docker-compose up -d
```

This will download the docker images which are several gigabytes of data so it might take a while.

Next we need to set up authentication for the API. The easiest way to do this in a lab environment is to just download a precreated "JWT-certificate" and associated authentication token like so:

```
wget https://raw.githubusercontent.com/SUNET/cnaas-nms/develop/docker/jwt-cert/public.pem
docker cp public.pem cnaas_cnaas_api_1:/opt/cnaas/jwtcert/public.pem
```

The docker container name might be different on your system, check with "docker ps". Without this JWT certificate the API will not start, you can check that the API starts up correctly and does not continue to endlessly crash/restart by checking the logs:

```
docker logs -f cnaas_cnaas_api_1
```

Export the authentication token:

```
export JWT_AUTH_TOKEN="eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiJ9.
eyJpYXQiOjE1NzEwNTk2MTgsIm5iZiI6MTU3MTA1OTYxOCwianRpIjoiNTQ2MDk2YTUtZTNmOS00NzFlLWE2NTctZWFlYTZkNzA4NmVhIiwic3ViIj
oiYWRtaW4iLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MifQ.
Sfffg9oZg_Kmoq7Oe8IoTcbuagpP6nuUXOQzqJpgDfqDq_GM_4zGzt7XxByD4G0q8g4gZGHQnV14TpDer2hJXw"
```

Test some API-calls:

```
curl -ks https://localhost/api/v1.0/system/version | jq .
```

```
curl -ks -H "Authorization: Bearer $JWT_AUTH_TOKEN" https://localhost/api/v1.0/devices | jq .
```

This should show the running version of the API, and return an empty list of devices from the database.

Configuring git repositories

To get CNaaS-NMS to generate any configuration you must first populate templates and settings. This is done via two separate git repositories. In this lab we will create the git repositories locally on your machine and have the API container fetch the contents from there.

Initialize git repository for templates:

```
mkdir git
cd git/
git init --bare cnaas-templates-origin.git
git clone cnaas-templates-origin.git cnaas-templates
```

This will setup a local "origin", then clone/check out that origin in to the directory cnaas-templates where you can work on the contents. Next we need to populate this repository with some data so we will grab some example templates from SUNET github and copy that over to our local repository:

```
git clone https://github.com/SUNET/cnaas-nms-templates sunet-github-templates
cp -r sunet-github-templates/eos cnaas-templates/
cd cnaas-templates/
git add .
git config --global user.email "lab@example.com"
git commit -a -m "init"
git push
```

You have now committed and pushed these templates to your local "origin" repository, but we still need to make the API fetch the contents from your local origin so that it can be used in the docker container. This is done via the API:

```
curl -ks -H "Authorization: Bearer $JWT_AUTH_TOKEN" https://localhost/api/v1.0/repository/templates -d
'{"action": "refresh"}' -X PUT -H "Content-Type: application/json"
```

You should get a response like:

```
{"status": "success", "data": "Cloned new from remote. Last commit 69611c3059889970433aca8e4b8fc46007ed81e0
master by root at 2021-09-28 11:48:34+00:00"}
```

Next we need to do the same things for the settings repository. From the git/ directory:

```
git init --bare cnaas-settings-origin.git
git clone cnaas-settings-origin.git cnaas-settings
git clone git://gitops.sunet.se/cnaas-lab-settings sunet-integrationtest-settings
cp -r sunet-integrationtest-settings/* cnaas-settings
cd cnaas-settings/
git add .
git commit -a -m "init"
git push
```

And ask the API to fetch the settings:

```
curl -ks -H "Authorization: Bearer $JWT_AUTH_TOKEN" https://localhost/api/v1.0/repository/settings -d '{"action":
"refresh"}' -X PUT -H "Content-Type: application/json"
```

Response:

```
{"status": "success", "data": "Cloned new from remote. Last commit c03813ff000293f606f33f48727b5baf3de68486
master by root at 2021-09-28 11:50:15+00:00"}
```

Try generating configuration

We'll add a simple device for testing:

```
curl -ks -H "Authorization: Bearer $JWT_AUTH_TOKEN" https://localhost/api/v1.0/device -X POST -d '{ "hostname":
"eosdist1", "management_ip": "10.100.3.0", "platform": "eos", "state": "MANAGED", "device_type": "DIST"}' -H
"Content-Type: application/json" | jq
```

And generate the configuration:

```
curl -ks -H "Authorization: Bearer $JWT_AUTH_TOKEN" https://localhost/api/v1.0/device/eosdist1/generate_config -H
"Content-Type: application/json" | jq .data.config.generated_config | sed 's/\\n/\n/g'
```

Check available variables that can be used when developing templates:

```
curl -ks -H "Authorization: Bearer $JWT_AUTH_TOKEN" https://localhost/api/v1.0/device/eosdist1/generate_config -H
"Content-Type: application/json" | jq .data.config.available_variables
```

Continue with Workshop 2