

Student: Pham Dinh Quy
Student ID: 313540040

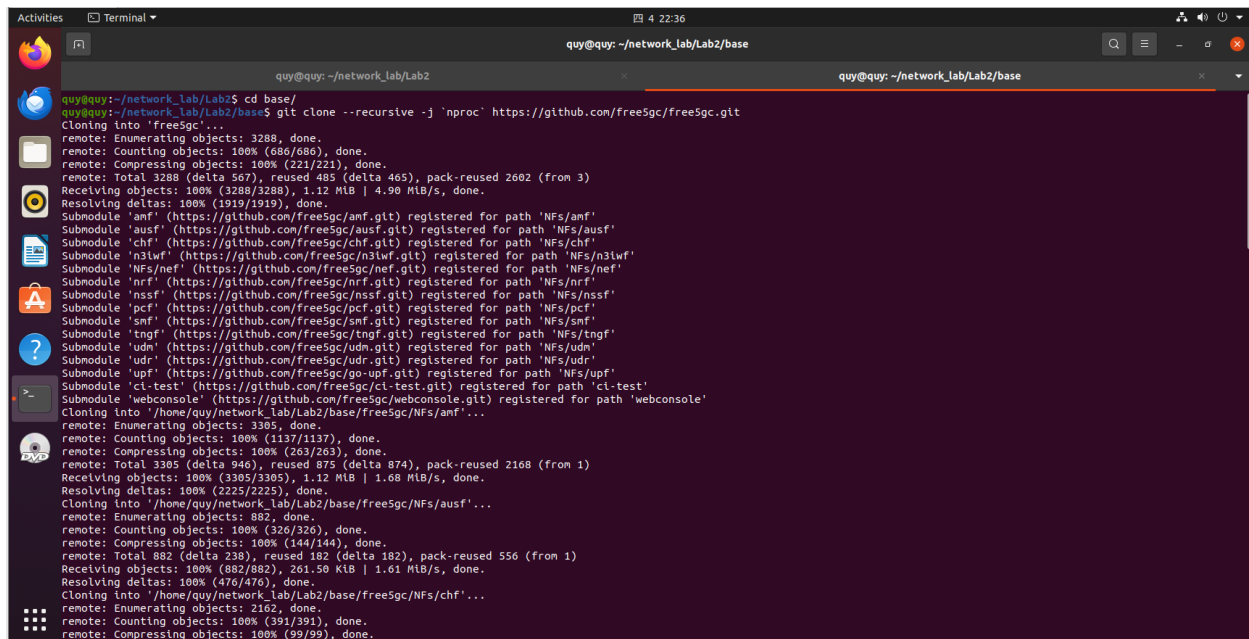
Lab2 Report

A. Building the system

1. Clone project from github repository

git clone <https://github.com/NYCU-CSCS20047-PoCaWN/Lab2.git>

git clone --recursive -j `nproc` <https://github.com/free5gc/free5gc.git>



```
quy@quy: ~/network_lab/Lab2$ cd base/
quy@quy: ~/network_lab/Lab2/base$ git clone --recursive -j `nproc` https://github.com/free5gc/free5gc.git
Cloning into 'free5gc'...
remote: Enumerating objects: 3288, done.
remote: Counting objects: 100% (686/686), done.
remote: Compressing objects: 100% (221/221), done.
remote: Total 3288 (delta 567), reused 485 (delta 465), pack-reused 2602 (from 3)
Receiving objects: 100% (3288/3288), 1.12 MiB | 4.90 MiB/s, done.
Resolving deltas: 100% (1919/1919), done.
Submodule 'anf' (https://github.com/free5gc/anf.git) registered for path 'Nfs/anf'
Submodule 'ausf' (https://github.com/free5gc/ausf.git) registered for path 'Nfs/ausf'
Submodule 'chf' (https://github.com/free5gc/chf.git) registered for path 'Nfs/chf'
Submodule 'n3lwf' (https://github.com/free5gc/n3lwf.git) registered for path 'Nfs/n3lwf'
Submodule 'nfs/nef' (https://github.com/free5gc/nfs/nef.git) registered for path 'Nfs/nef'
Submodule 'nrf' (https://github.com/free5gc/nrf.git) registered for path 'Nfs/nrf'
Submodule 'nssf' (https://github.com/free5gc/nssf.git) registered for path 'Nfs/nssf'
Submodule 'pcf' (https://github.com/free5gc/pcf.git) registered for path 'Nfs/pcf'
Submodule 'snf' (https://github.com/free5gc/snf.git) registered for path 'Nfs/snf'
Submodule 'tngf' (https://github.com/free5gc/tngf.git) registered for path 'Nfs/tngf'
Submodule 'udm' (https://github.com/free5gc/udm.git) registered for path 'Nfs/udm'
Submodule 'udr' (https://github.com/free5gc/udr.git) registered for path 'Nfs/udr'
Submodule 'upf' (https://github.com/free5gc/upf.git) registered for path 'Nfs/upf'
Submodule 'ci-test' (https://github.com/free5gc/ci-test.git) registered for path 'ci-test'
Submodule 'webconsole' (https://github.com/free5gc/webconsole.git) registered for path 'webconsole'
Cloning into '/home/quy/network_lab/Lab2/base/free5gc/Nfs/anf'...
remote: Enumerating objects: 100% (1137/1137), done.
remote: Compressing objects: 100% (263/263), done.
remote: Total 3305 (delta 946), reused 875 (delta 874), pack-reused 2168 (from 1)
Receiving objects: 100% (3305/3305), 1.12 MiB | 1.68 MiB/s, done.
Resolving deltas: 100% (2225/2225), done.
Cloning into '/home/quy/network_lab/Lab2/base/free5gc/Nfs/ausf'...
remote: Enumerating objects: 882, done.
remote: Counting objects: 100% (326/326), done.
remote: Compressing objects: 100% (144/144), done.
remote: Total 882 (delta 230), reused 182 (delta 182), pack-reused 556 (from 1)
Receiving objects: 100% (882/882), 261.50 KiB | 1.61 MiB/s, done.
Resolving deltas: 100% (476/476), done.
Cloning into '/home/quy/network_lab/Lab2/base/free5gc/Nfs/chf'...
remote: Enumerating objects: 2162, done.
remote: Counting objects: 100% (391/391), done.
remote: Compressing objects: 100% (99/99), done.
```

2. Build image from local

Using <sudo make>:

This command automatically compile and build programs according to instructions in Makefile. The Makefile includes commands that build a base Docker image for Free5GC, with steps such as installing dependencies, copying the Free5GC source code, and compiling the required modules., preparing for deploy the docker image in the next step.

```
Activities Terminal 4 23:43
quy@quy: ~/network_lab/Lab2
quy@quy: ~/network_lab/Lab2/config
quy@quy:~/network_lab/Lab2$ cd ..
quy@quy:~/network_lab/Lab2$ sudo make
[sudo] password for quy:
docker build -t 'free5gc'/'base':'latest' ./base
[+] Building 170.8s (77%) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 479B
=> [internal] load metadata for docker.io/library/golang:1.21.8-bullseye
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/golang:1.21.8-bullseye@sha256:81d98548f08e22a59c5c0be814acc0997f3dfd3313487adcf4e1d3d962af3a43
=> => resolve docker.io/library/golang:1.21.8-bullseye@sha256:81d98548f08e22a59c5c0be814acc0997f3dfd3313487adcf4e1d3d962af3a43
=> sha256:2fdd729b343a8e184d0d6daf378ff3393bf97d2b59e97f40ac0e73d3bed33de0 2.88kB / 2.88kB
=> sha256:ec335f17d0c74f7a270925cb1bbd29acc72ae984c6f4570f9ae369e3eebb64ed 55.08MB / 55.08MB
=> sha256:81d98548f08e22a59c5c0be814acc0997f3dfd3313487adcf4e1d3d962af3a43 1.64kB / 1.64kB
=> sha256:d2b475e1910dc07f5c0bfeddb5a8634024593a0edf3b91f0c723330ef10585 15.76MB / 15.76MB
=> sha256:7f67b1746a83d257a639acf8eac47bfaf1854670097ea4234f12857cfc7d5932 54.59MB / 54.59MB
=> sha256:2b24e533430ee0e12544f2155ff8b93557bdeee898e08b91045f279c9cfac 1.79kB / 1.79kB
=> sha256:fbf73e3517bc8fa1e9863e448b6ad3b54c5482345b15ce6ad28439cf1f617523 86.11MB / 86.11MB
=> sha256:0635831cef590cc18d45a50a86211bcd0b7e2dc2aeac5a3d1967104e0b7e3d9 67.01MB / 67.01MB
=> extracting sha256:ec335f17d0c74f7a270925cb1bbd29acc72ae984c6f4570f9ae369e3eebb64ed 24.65
=> sha256:579c95fcd9f09d4b635a801a036ce19e0793e293b61b4621bac859996c4bbd47 173B / 173B
=> sha256:4f4fb790ef54401cfa02571ae0db9a0dc1e0c0b5577484a6d75e68dc38e8acc1 32B / 32B
=> extracting sha256:d2b475e1910dc07f5c0bfeddb5a8634024593a0edf3b91f0c723330ef10585 3.85
=> extracting sha256:7f67b1746a83d257a639acf8eac47bfaf1854670097ea4234f12857cfc7d5932 22.85
=> extracting sha256:fbf73e3517bc8fa1e9863e448b6ad3b54c5482345b15ce6ad28439cf1f617523 20.15
=> extracting sha256:0635831cef590cc18d45a50a86211bcd0b7e2dc2aeac5a3d1967104e0b7e3d9 45.05
=> extracting sha256:579c95fcd9f09d4b635a801a036ce19e0793e293b61b4621bac859996c4bbd47 0.05
=> extracting sha256:4f4fb790ef54401cfa02571ae0db9a0dc1e0c0b5577484a6d75e68dc38e8acc1 0.05
=> [2/3] RUN apt-get update && apt-get -y install gcc cmake autoconf libtool pkg-config libnsl-dev libyaml-dev apt-transport-https ca-certificates
=> [3/3] RUN apt-get clean
=> exporting to image
=> writing image sha256:b9701e9bbff5a1d18a47dbc9e8dd66aba5ccc682d63b5c16b4855b6dec447159
=> naming to docker.io/free5gc/base:latest
docker image ls 'free5gc'/'base':'latest'
REPOSITORY TAG IMAGE ID CREATED SIZE
free5gc/base latest b9701e9bbff 7 seconds ago 849MB
docker build --build-arg F5GC_MODULE=amf -t 'free5gc'/'amf':'base':'latest' -f ./base/Dockerfile.nf ./base
[+] Building 248.8s (15/15) FINISHED
=> [internal] load build definition from Dockerfile.nf
=> => transferring dockerfile: 780B
=> => naming to docker.io/free5gc/amf:'base':'latest'
docker build -t 'free5gc'/'webconsole':'base':'latest' -f ./base/Dockerfile.nf.webconsole ./base
[+] Building 535.1s (17/17) FINISHED
=> [internal] load build definition from Dockerfile.nf.webconsole
=> => transferring dockerfile: 1.07kB
=> [internal] load metadata for docker.io/free5gc/base:latest
=> [internal] load metadata for docker.io/library/alpine:3.15
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [stage-1 1/7] FROM docker.io/library/alpine:3.15@sha256:19b4bcc4f60e99dd5ebdca0cbce22c503bbcff197549d7e19dab4f22254dc864
=> [my-base 1/4] FROM docker.io/free5gc/base:latest
=> CACHED [stage-1 2/7] WORKDIR /free5gc
=> [stage-1 3/7] RUN mkdir -p cert/ public
=> [internal] load build context
=> => transferring context: 153.11kB
=> CACHED [my-base 2/4] COPY free5gc/ /go/src/free5gc/
=> [my-base 3/4] RUN curl -fsSL https://deb.nodesource.com/setup_20.x | bash - && apt update && apt install nodejs -y
=> [my-base 4/4] RUN cd /go/src/free5gc && apt-get update && apt-get -y install sudo && make webconsole
=> [stage-1 4/7] COPY --from=my-base /go/src/free5gc/webconsole/bin/webconsole ./webui
=> [stage-1 5/7] COPY --from=my-base /go/src/free5gc/webconsole/public ./public
=> [stage-1 6/7] COPY --from=my-base /go/src/free5gc/config/* ./config/
=> [stage-1 7/7] COPY --from=my-base /go/src/free5gc/cert/* ./cert/
=> exporting to image
=> exporting layers
=> writing image sha256:c61a7adf027919a1ddeaeafb2076162f4be43bf7cd93b924e90b15e231df958d
=> naming to docker.io/free5gc/webconsole-base:latest
quy@quy:~/network_lab/Lab2$
```

Using docker build:

The docker compose build command will build all Docker images from the services defined in docker-compose file such as amf, nrf, webui, ueransim,... and network layer such as N2, N4, N6,... The docker compose file will also point to the Docker file located in corresponding directory to get the data.

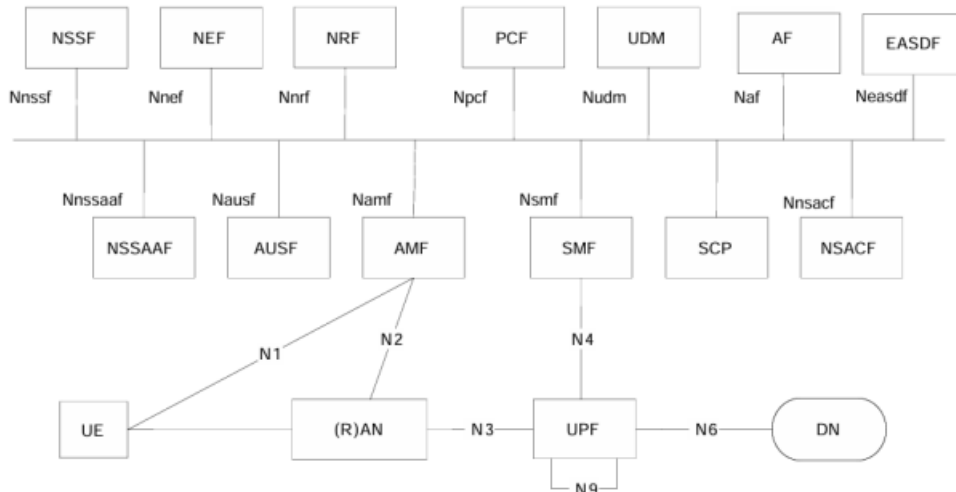
```
Activities Terminal 5 00:21
quy@quy: ~/network_lab/Lab2
quy@quy: ~/network_lab/Lab2$ sudo docker compose -f 313540040.yaml build
[sudo] password for quy:
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 2079.0s (147/147) FINISHED
=> [freeSgc-ngw internal] load build definition from Dockerfile
=> == transferring dockerfile: 383B 0.1s
=> [freeSgc-nrf internal] load build definition from Dockerfile 0.1s
=> == transferring dockerfile: 642B 0.1s
=> [freeSgc-upf internal] load build definition from Dockerfile 0.0s
=> == transferring dockerfile: 936B 0.0s
=> [freeSgc-smf internal] load metadata for docker.io/bitnami/minideb:bullseye 6.9s
=> [freeSgc-upf internal] load metadata for docker.io/freeSgc/upf-base:latest 0.0s
=> [freeSgc-nrf internal] load metadata for docker.io/freeSgc/base:latest 0.0s
=> [freeSgc-chf internal] load metadata for docker.io/freeSgc/nrf-base:latest 0.0s
=> [freeSgc-nrf internal] load metadata for docker.io/library/alpine:3.15 5.0s
=> [freeSgc-nrf internal] load .dockerignore 0.0s
=> == transferring context: 2B 0.0s
=> [freeSgc-chf stage-1 1/6] FROM docker.io/library/alpine:3.15@sha256:19b4bcc4f60e99dd5ebdca0cbce22c503bbcff197549d7e19dab4f22254dc864 0.0s
=> [freeSgc-nrf builder 1/1] FROM docker.io/freeSgc/nrf-base:latest 0.4s
=> CACHED [freeSgc-chf stage-1 2/6] WORKDIR /freeSgc 45.4s
=> [freeSgc-nrf stage-1 3/6] RUN mkdir -p config/ log/ cert/ 3.5s
=> [freeSgc-ngw internal] load .dockerignore 0.2s
=> == transferring context: 2B 0.0s
=> [freeSgc-upf internal] load .dockerignore 0.4s
=> == transferring context: 2B 0.1s
=> CACHED [ueransim stage-1 1/7] FROM docker.io/bitnami/minideb:bullseye@sha256:d2e21765766b393fc43f250e632fb8b07722ed94126d94f91b37687d4e4dbfe8 0.1s
=> == resolve docker.io/bitnami/minideb:bullseye@sha256:d2e21765766b393fc43f250e632fb8b07722ed94126d94f91b37687d4e4dbfe8 45.4s
=> == sha256:d2e21765766b393fc43f250e632fb8b07722ed94126d94f91b37687d4e4dbfe8 741B / 741B 0.0s
=> == sha256:4619e708f8bc7ec6751cf41cf64b24fa1fadd7cfa73742ea3a3ff97f61e0df8 528B / 528B 0.0s
=> == sha256:93af0c23b703296f7ba0725f47c66423a8b5b1e0da491fc38ee23a31f88a0df 910B / 910B 0.0s
=> == sha256:17be8885e9d6ab819673d2186d50a8384ee935c7e046ae028a51beeac05264 2.9s
=> == extracting sha256:17be8885e9d6ab819673d2186d50a8384ee935c7e046ae028a51beeac05264 30.96MB / 30.96MB 38.0s
=> CACHED [freeSgc-upf my-base 1/2] FROM docker.io/freeSgc/base:latest 0.1s
=> [freeSgc-upf builder 1/1] FROM docker.io/freeSgc/upf-base:latest 0.8s
=> [freeSgc-upf my-base 2/2] RUN git clone https://github.com/freeSgc/go-gtp5gnl.git && mkdir "go-gtp5gnl/bin" && cd "go-gtp5gnl/cmd/gotgp5g-tunnel" && go build -o "/" 161.8s
=> [freeSgc-nrf stage-1 4/6] COPY --from=builder /freeSgc/nrf ./ 0.5s
=> [freeSgc-nrf stage-1 5/6] COPY --from=builder /freeSgc/cert/nrf.pem ./cert/ 0.4s
=> [freeSgc-nrf stage-1 6/6] COPY --from=builder /freeSgc/cert/nrf.key ./cert/ 0.3s
=> [freeSgc-nrf] exporting to image 0.7s
=> == exporting layers 0.6s
=> == writing image sha256:89a1a57c7f3da323fba69887b3d658fd54f52b195592d5bcc447ec8a0d5c413 0.0s
```

ve the mouse pointer inside or press Ctrl+G.

```
Activities Terminal 5 00:20
quy@quy: ~/network_lab/Lab2
quy@quy: ~/network_lab/Lab2$
=> == extracting sha256:8c51bf5a7bd72d9b6eb9e7417b838d37e0786ad44f0833863bae467f7239a059 0.0s
=> == extracting sha256:71c2d0da5d28e6d960493c4e643061708eae3001808dbd6d6498670d08e5658d 0.5s
=> [ueransim stage-1 2/10] RUN apt-get update && apt-get install libscpt-dev lksctp-tools iproute2 iputils-ping procs psmisc -y 60.5s
=> [freeSgc-smf stage-1 3/7] WORKDIR /freeSgc 0.2s
=> [freeSgc-smf stage-1 4/7] RUN mkdir -p config/ log/ cert/ 5.4s
=> [freeSgc-smf stage-1 5/7] COPY --from=builder /freeSgc/smf ./ 1.1s
=> [ueransim stage-1 3/10] RUN apt-get update && apt-get install -y vim strace net-tools iputils-ping curl netcat tcpdump 44.7s
=> [freeSgc-smf stage-1 6/7] COPY --from=builder /freeSgc/cert/smf.pem ./cert/ 0.3s
=> [freeSgc-smf stage-1 7/7] COPY --from=builder /freeSgc/cert/smf.key ./cert/ 0.5s
=> [freeSgc-smf] exporting to image 3.9s
=> == exporting layers 3.6s
=> == writing image sha256:fbe8961f1052f838f02ado1cd0a40e48efc149cba9f3ff9d559c36edd02b14 0.0s
=> == naming to docker.io/library/lab2-freeSgc-smf 0.1s
=> [freeSgc-smf] resolving provenance for metadata file 0.1s
=> [ueransim stage-1 4/10] WORKDIR /ueransim 0.1s
=> [ueransim stage-1 5/10] RUN mkdir -p config/ binder/ 3.0s
=> [ueransim builder 2/2] RUN apt-get update && apt-get install libscpt-dev lksctp-tools iproute2 -y && wget https://github.com/Kitware/CMake/releases/download/v3.160.2 160.2s
=> [ueransim stage-1 6/10] COPY --from=builder /UERANSIM/build/nr-gnb . 0.3s
=> [ueransim stage-1 7/10] COPY --from=builder /UERANSIM/build/nr-ue . 0.3s
=> [ueransim stage-1 8/10] COPY --from=builder /UERANSIM/build/nr-ctrl . 0.1s
=> [ueransim stage-1 9/10] COPY --from=builder /UERANSIM/build/nr-binder binder/ 0.2s
=> [ueransim stage-1 10/10] COPY --from=builder /UERANSIM/build/libdevbnd.so binder/ 0.1s
=> [ueransim] exporting to image 10.6s
=> == exporting layers 10.5s
=> == writing image sha256:ed04d92560748ba18a8ca326f58e9577849f1f87b12fd22dc6c33908b386c04 0.0s
=> == naming to docker.io/library/lab2-ueransim 0.0s
=> [ueransim] resolving provenance for metadata file 0.1s
[+] Building 13/13
✓ freeSgc-smf Built 0.0s
✓ freeSgc-ausf Built 0.0s
✓ freeSgc-chf Built 0.0s
✓ freeSgc-ngw Built 0.0s
✓ freeSgc-nrf Built 0.0s
✓ freeSgc-nssf Built 0.0s
✓ freeSgc-pcf Built 0.0s
✓ freeSgc-smf Built 0.0s
✓ freeSgc-udm Built 0.0s
✓ freeSgc-udr Built 0.0s
✓ freeSgc-upf Built 0.0s
✓ freeSgc-webui Built 0.0s
✓ ueransim Built 0.0s
quy@quy: ~/network_lab/Lab2$
```

3. Run container

Before I run up the container, I need to modify the config file in `./config` directory.
Here is the 5G core network architecture:



5G core network architecture

Because we need to configure Docker networks for 5G interfaces (N1/N2, N3, N4, N6), we should configure the such files: `amfcfg.yaml`, `smfcfg.yaml`, `upfcfg.yaml`, and `gnbcfg.yaml`:

- In `amfcfg.yaml`, `ngapIpList` is the list of IP addresses that AMF will listen on to establish NGAP connections (N2 interface) from gNB, In `313540040.yaml`, `amf` container is using alias `amf.n2.org`, so I will set this value to FDQN of `ngapIpList` in `amfcfg.yaml` (line8).

```
configuration:
  amfName: AMF # the name of this AMF
  ngapIpList: # the IP list of N2 interfaces on this AMF
    - amf.n2.org
  ngapPort: 38412 # the SCTP port listened by NGAP
  sbi: # Service-based interface information
```

- In smfcfg.yaml, we have to modify the SMF and UPF properties. UPF connect to SMF through N4 interface. I set SMF nodeId is smf.free5gc.org, listenAddr, externalAddr is amf.n4.org (line 52-54). The UPF nodeId and UPF N4 interface IP is upf.n4.org (line 66-67). The N3 interface endpoint (line 75) will be upf N3 static ip defined in 313540040.yaml: 10.100.3.100.

```

pfcf: # the IP address of N4 interface on this SMF (PFCP)
# addr config is deprecated in smf config v1.0.3, please use the following config
nodeID: smf.free5gc.org # the Node ID of this SMF
listenAddr: smf.n4.org # the IP/FQDN of N4 interface on this SMF (PFCP)
externalAddr: smf.n4.org # the IP/FQDN of N4 interface on this SMF (PFCP)
userplaneInformation: # list of userplane information
upNodes: # information of userplane node (AN or UPF)
  gNB1: # the name of the node
    type: AN # the type of the node (AN or UPF)
  UPF: # the name of the node
    type: UPF # the type of the node (AN or UPF)
    nodeID: upf.n4.org # the Node ID of this UPF
    addr: upf.n4.org # the IP/FQDN of N4 interface on this UPF (PFCP)
    sNssaiUpfInfos: # S-NSSAI information list for this UPF

  interfaces: # Interface list for this UPF
    - interfaceType: N3 # the type of the interface (N3 or N9)
      endpoints: # the IP address of this N3/N9 interface on this UPF
        - 10.100.3.100
      networkInstances: # Data Network Name (DNN)
        - internet
  links: # the topology graph of userplane, A and B represent the two nodes of each link

```

- In the upfcfg.yaml, I need to set the N4 interface of UPF in line 6-7, so I change it to upf.n4.org. The N3 address will be upf N3 static ip defined in 313540040.yaml: 10.100.3.100 (line 14)

```

pfcf:
  addr: upf.n4.org # IP addr for listening
  nodeID: upf.n4.org # External IP or FQDN can be reached
  retransTimeout: 1s # retransmission timeout
  maxRetrans: 3 # the max number of retransmission

gtpu:
  forwarder: gtp5g
  # The IP list of the N3/N9 interfaces on this UPF
  # If there are multiple connection, set addr to 0.0.0.0 or list all the addresses
  ifList:
    - addr: 10.100.3.100
      type: N3

```


- In gnbcfg.yaml, I set the gNB's local IP address for N2 and N3 Interface is gnb.n2.org and gnb.n3.org (line 9-10) and amf address is amf.n2.org.

```
linkIp: 127.0.0.1 # gNB's local IP address for Radio Link Simulation (Usually same with local IP)
ngapIp: gnb.n2.org #local IP address for N2 Interface (Usually same with local IP)
gtpIp: gnb.n3.org # gNB's local IP address for N3 Interface (Usually same with local IP)

# List of AMF address information
amfConfigs:
- address: amf.n2.org
  port: 38412
```

After modifying the config file, I run up the container by <docker compose up>:

This command will run up all the containers defined in 313540040.yaml file, image is take from the previous step.

```
quydingh@quydingh:~/computer_network/Lab2$ sudo docker compose -f 313540040.yaml up -d
[sudo] password for quydingh:
[+] Running 14/14
 ✓ Container n6gw      Started
 ✓ Container upf       Started
 ✓ Container mongodb   Started
 ✓ Container nrf       Started
 ✓ Container pcf       Started
 ✓ Container smf       Started
 ✓ Container webui     Started
 ✓ Container chf       Started
 ✓ Container nssf      Started
 ✓ Container udm       Started
 ✓ Container udr       Started
 ✓ Container ausf      Started
 ✓ Container amf       Started
 ✓ Container ueransim  Started
```

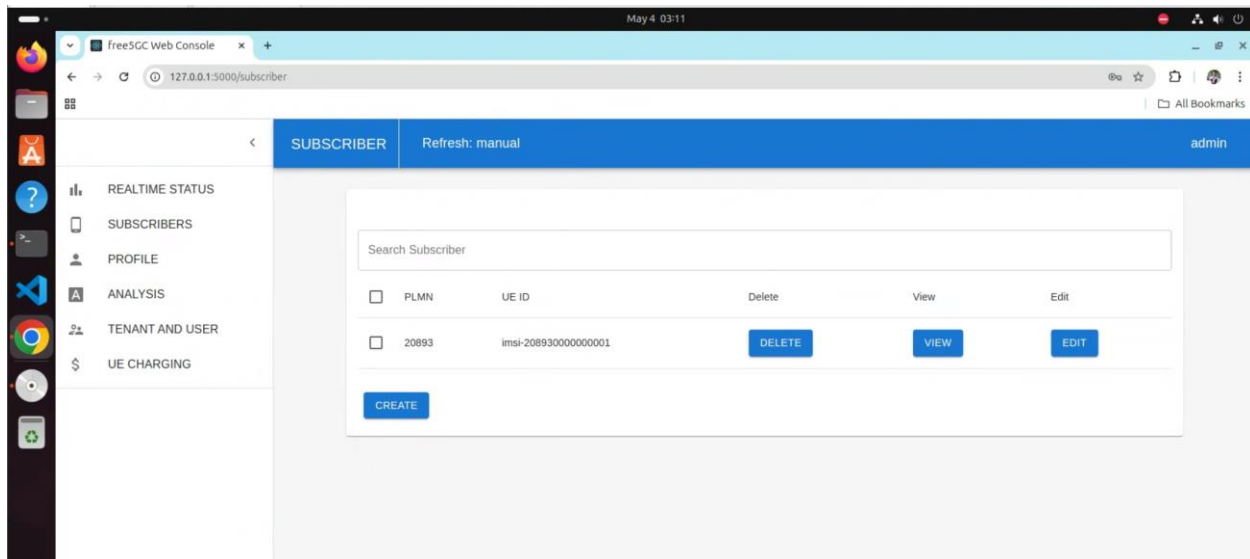
```
quydingh@quydingh:~/computer_network/Lab2$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
380d00093fc	lab2-ueransim	"./nr-gnb -c ./confi..." ueransim	7 seconds ago	Up 6 seconds	
e5f860acb3bd	lab2-free5gc-ausf	"./ausf -c ./config/..." ausf	7 seconds ago	Up 6 seconds	8000/tcp
360860d66ae2	lab2-free5gc-webui	"./webui -c ./config/..." webui	7 seconds ago	Up 6 seconds	0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
59f0ffbeeb66	lab2-free5gc-udr	"./udr -c ./config/u..." udr	7 seconds ago	Up 6 seconds	8000/tcp
19bb014dd1eb	lab2-free5gc-pcf	"./pcf -c ./config/p..." pcf	7 seconds ago	Up 6 seconds	8000/tcp
9b73bc2dc127	lab2-free5gc-smf	"./smf -c ./config/s..." smf	7 seconds ago	Up 6 seconds	8000/tcp
f721f5a061d4	lab2-free5gc-nssf	"./nssf -c ./config/..." nssf	7 seconds ago	Up 6 seconds	8000/tcp
a671bcb43872	lab2-free5gc-udm	"./udm -c ./config/u..." udm	7 seconds ago	Up 6 seconds	8000/tcp
f3e918534134	lab2-free5gc-amf	"./amf -c ./config/a..." amf	7 seconds ago	Up 6 seconds	8000/tcp
f5392169225c	lab2-free5gc-nrf	"./nrf -c ./config/n..." nrf	7 seconds ago	Up 6 seconds	8000/tcp
19b6b61c68bb	mongo	"docker-entrypoint.s..." mongod	7 seconds ago	Up 7 seconds	27017/tcp
84cb38f1567f	lab2-free5gc-upf	"bash -c './upf-ipta..." upf	7 seconds ago	Up 7 seconds	
69b32b7e392d	lab2-free5gc-n6gw	"bash -c './n6gw-ipt..." n6gw	7 seconds ago	Up 7 seconds	
11e48d3ad7f7	portainer/portainer-ce	"/portainer"	6 days ago	Up 22 hours	0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp, 9443/tcp, 0.0.0.0:80->80/tcp

```
quydingh@quydingh:~/computer_network/Lab2$
```

4. Test by exec to ueramsim container

Go into WebUI and create Subscriber:



Executing to ueramsim container and run “./nr-ue -c config/uecfg.yaml” to register ueransim to the 5g core network.

```
^Croot@380d00093fc:/ueransim# quydinh@quydinh:~/computer_network/Lab2$ sudo docker exec -it ueransim /bin/bash
root@380d00093fc:/ueransim# ./nr-ue -c config/uecfg.yaml
UERANSIM v3.2.7
[2025-05-03 19:10:56.839] [nas] [info] UE switches to state [MM-DEREGISTERED/PLMN-SEARCH]
[2025-05-03 19:10:56.839] [rrc] [debug] New signal detected for cell[1], total [1] cells in coverage
[2025-05-03 19:10:56.839] [nas] [info] Selected plmn[208/93]
[2025-05-03 19:10:56.839] [rrc] [info] Selected cell plmn[208/93] tac[1] category[SUITABLE]
[2025-05-03 19:10:56.839] [nas] [info] UE switches to state [MM-DEREGISTERED/PS]
[2025-05-03 19:10:56.839] [nas] [info] UE switches to state [MM-DEREGISTERED/NORMAL-SERVICE]
[2025-05-03 19:10:56.839] [nas] [debug] Initial registration required due to [MM-DEREG-NORMAL-SERVICE]
[2025-05-03 19:10:56.840] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_sig]
[2025-05-03 19:10:56.840] [nas] [debug] Sending Initial Registration
[2025-05-03 19:10:56.841] [nas] [info] UE switches to state [MM-REGISTER-INITIATED]
[2025-05-03 19:10:56.841] [rrc] [debug] Sending RRC Setup Request
[2025-05-03 19:10:56.842] [rrc] [info] RRC connection established
[2025-05-03 19:10:56.842] [rrc] [info] UE switches to state [RRC-CONNECTED]
[2025-05-03 19:10:56.842] [nas] [info] UE switches to state [CM-CONNECTED]
[2025-05-03 19:10:56.876] [nas] [debug] Authentication Request received
[2025-05-03 19:10:56.876] [nas] [debug] Received SQN [000000000023]
[2025-05-03 19:10:56.876] [nas] [debug] SQN-MS [000000000000]
[2025-05-03 19:10:56.895] [nas] [debug] Security Mode Command received
[2025-05-03 19:10:56.895] [nas] [debug] Selected integrity[2] ciphering[0]
[2025-05-03 19:10:56.964] [nas] [debug] Registration accept received
[2025-05-03 19:10:56.964] [nas] [info] UE switches to state [MM-REGISTERED/NORMAL-SERVICE]
[2025-05-03 19:10:56.964] [nas] [debug] Sending Registration Complete
[2025-05-03 19:10:56.964] [nas] [info] Initial Registration is successful
[2025-05-03 19:10:56.964] [nas] [debug] Sending PDU Session Establishment Request
[2025-05-03 19:10:56.969] [nas] [debug] UAC access attempt is allowed for identity[0], category[MO_sig]
[2025-05-03 19:10:57.170] [nas] [debug] Configuration Update Command received
[2025-05-03 19:10:57.272] [nas] [debug] PDU Session Establishment Accept received
[2025-05-03 19:10:57.272] [nas] [info] PDU Session establishment is successful PSI[1]
[2025-05-03 19:10:57.292] [app] [info] Connection setup for PDU session[1] is successful, TUN interface[uesimtun0, 10.60.0.1] is up.
```

After that I try to ping 8.8.8.8 through uesimtun0 interface. The successful ping mean that ue successfully register in to local core 5G network and can reach out to external networks.

```
May 4 03:15
quy dinh@quy dinh: ~/computer_network/Lab2/config
quy dinh@quy dinh: ~/computer_network/Lab2$ sudo docker exec -it /bin/bash ueransim
Error response from daemon: No such container: bin/bash
quy dinh@quy dinh: ~/computer_network/Lab2/config$ sudo docker exec -it ueransim /bin/bash
root@380d00093fc:/ueransim# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0@1f860: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 1a:ff:c6:da:04:36 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.100.2.3/24 brd 10.100.2.255 scope global eth0
        valid_lft forever preferred_lft forever
3: eth1@1f861: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 9e:09:83:30:41:62 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.100.3.2/24 brd 10.100.3.255 scope global eth1
        valid_lft forever preferred_lft forever
4: uesintun0: <POINTOPOINT,PROMISC,NOTRAILERS,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.60.0.1/16 scope global uesintun0
        valid_lft forever preferred_lft forever
    inet6 fe80::972:79a7:c3f2:30e7/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
root@380d00093fc:/ueransim# ping -I uesintun0 8.8.8.8
PING 8.8.8.8 (8.8.8.8) from 10.60.0.1 uesintun0: 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=125 time=8.07 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=125 time=7.49 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=125 time=7.94 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=125 time=10.7 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=125 time=11.2 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=125 time=6.39 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=125 time=11.0 ms
```

B. Question and Answer

1. What are N1, N2, N3, N4, and N6 in the 5G core network?

The N1, N2, N3, N4, N6 interfaces define how different components communicate with each other:

- N1 is interface for signaling and registration (initial access, authentication, etc.) from UE (mobile phone, IoT device,...) to the AMF (Access and Mobility Management Function)
- N2 interface connect between gNB and AMF, using for UE registration, mobility management, handovers.
- N3 interface is connection between gNB and UPF, it handle the actual data traffic (internet, app data) from UE flows.
- N4 interface connect SMF (Session Management Function) to UPF, it tells the UPF how to route packets (e.g., rules, QoS) and controls user plane forwarding behavior
- N6 connect UPF to external network, it handle actual outgoing/incoming data to/from external networks like your UE's internet traffic after it passes through UPF

2. What is bridge network in docker?

A bridge network is the default network driver Docker uses. It acts as a virtual LAN where containers can talk to each other within the same host, it also provide isolation from containers that aren't connected to that bridge network .Docker assigns each container an IP within that network. Containers can also communicate via container name (DNS resolution).

In my lab, there are also bridge networks namely lab2_sbinet:

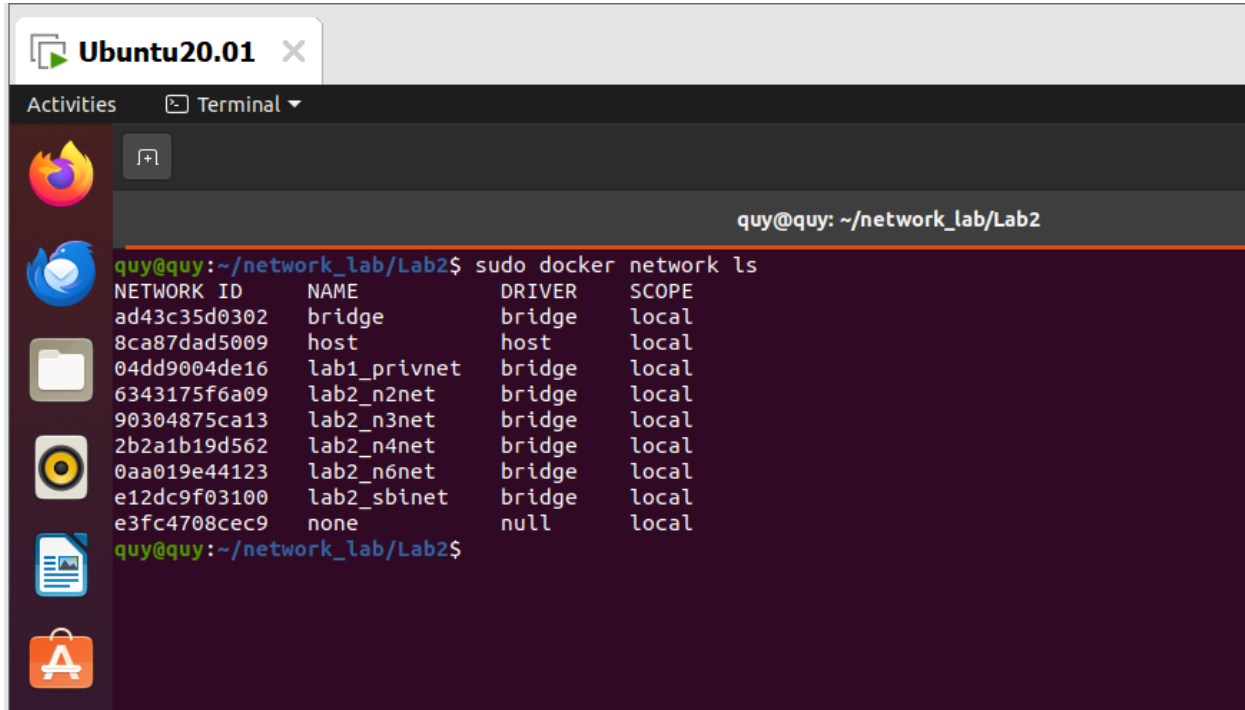
- SBI network (10.100.163.0/24) is defined by user (like me or any administrator) and connect 5G core components such as AMF, SMF, UPF, WebUI,... ensure containers can ping, communicate via TCP/UDP, and route according to the simulated packet flow in the 5G network

3. What is the name of each network you are using and the subnet for each?

To view all the networks, I run: `sudo docker network ls`

To view the network detail, I run: `sudo docker network inspect <network name>`

I use 5 network in this lab: lab2_n2net, lab2_n3net, lab2_n4net, lab2_n6net, lab2_sbinet, the rest are networks created from previous labs.



```
quy@quy: ~/network_lab/Lab2$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
ad43c35d0302        bridge             bridge              local
8ca87dad5009        host               host                local
04dd9004de16        lab1_privnet       bridge              local
6343175f6a09        lab2_n2net         bridge              local
90304875ca13        lab2_n3net         bridge              local
2b2a1b19d562        lab2_n4net         bridge              local
0aa019e44123        lab2_n6net         bridge              local
e12dc9f03100        lab2_sbinet        bridge              local
e3fc4708cec9        none               null                local
quy@quy:~/network_lab/Lab2$
```

```
Activities Terminal 四 12 21:4
quy@quy: ~/network
quy@quy: ~/network_lab/Lab2
quy@quy:~/network_lab/Lab2$ sudo docker network inspect lab2_n2net
[
  {
    "Name": "lab2_n2net",
    "Id": "6343175f6a0958ad1d4928edbb491d7bfd861e1a0d4e654dc4d5979844170977",
    "Created": "2025-04-12T21:45:07.309227634+08:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv4": true,
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "10.100.2.0/24"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "8ebec14c245e60bf1d6fe8806d5c7ebb213f4c4a03570aa9a21c914dba45f554": {
        "Name": "ueransim",
        "EndpointID": "bf9a6a042f32a9f96bb4e8adb09b44f45c277c193b23b43445ea2fe88219ecac",
        "MacAddress": "16:06:86:f6:e8:2f",
        "IPv4Address": "10.100.2.3/24",
        "IPv6Address": ""
      },
      "f19617ad85d0b68c149d731fc0e0b86c5ee81e4c7c07c1ef844681b9cf17aca8": {
        "Name": "amf2",
        "EndpointID": "0a1e2594970b4fa8d23918757ce53f4bf664420fa186d992db659398b48d2887",
        "MacAddress": "da:3a:ab:3c:54:ab",
        "IPv4Address": "10.100.2.2/24",
        "IPv6Address": ""
      }
    },
    "Options": {
      "com.docker.network.bridge.name": "br-n2"
    },
    "Labels": {
      "com.docker.compose.config-hash": "a6a305b88add3265550ce0531a3d72d65aac7ad792db97417e7f8411196c2246",
      "com.docker.compose.network": "n2net",
      "com.docker.compose.project": "Lab2",
      "com.docker.compose.version": "2.34.0"
    }
  }
]
quy@quy:~/network_lab/Lab2$
```

N2 network

- Name: lab2_n2net
- Subnet: "Subnet": "10.100.2.0/24"

N3 network

- Name: lab2_n3net
- Subnet: 10.100.3.0/24

N4 network

- Name: lab2_n4net
- Subnet: 10.100.4.0/24

N6 network

- Name: lab2_n6net
- Subnet: 10.100.6.0/24

SBI network

- Name: lab2_sbinetwork
- Subnet: 10.100.163.0/24

All the subnet of networks is working properly like I defined in the 313540040.yaml file, from line 305 to 345

4. What the IP Address for your UPF? (Also provides screenshot and steps for how you get your answers.)

To see the UPF IP_address, I run command:
docker inspect <upf_container_name>

```

    "description": "free5gc open source 5G Core Network",
    "version": "Stage 3"
  },
  "NetworkSettings": {
    "Bridge": "",
    "SandboxKey": "/var/run/docker/netns/93af28993668",
    "Ports": {},
    "HostPortMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "",
    "Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "MacAddress": "",
    "Networks": {
      "lab2_n6net": {
        "IPAMConfig": {
          "IPv4Address": "10.100.3.100"
        },
        "Links": null,
        "Aliases": [
          "upf2",
          "free5gc-upf",
          "upf.n3.org"
        ],
        "MacAddress": "a6c1e7:15:99:22",
        "DriverOpts": null,
        "GwPriority": 0,
        "NetworkID": "e071833d64e5a6b6d4f6687410c44d1babe1efda9228a23d81eb4117d986f",
        "EndpointID": "38477fb32eb5c92e4e2962c1801ffe1438dc798348773af4842fcb10073eb3",
        "Gateway": "10.100.3.1",
        "IPAddress": "10.100.3.100",
        "IPPrefixLen": 24,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "DNSNames": [
          "upf2",
          "free5gc-upf",
          "upf.n3.org",
          "f37eda5158ac"
        ]
      }
    }
  }
}

lab2_n6net: {
  "IPAMConfig": {
    "Links": null,
    "Aliases": [
      "upf2",
      "free5gc-upf",
      "upf.n4.org"
    ],
    "MacAddress": "86:24:fb:de:56:76",
    "DriverOpts": null,
    "GwPriority": 0,
    "NetworkID": "8336a95d7a29f959dbf639b1d01d8e11b06da899ebc5cd8d136449d42af0",
    "EndpointID": "d8ee3210cd80450a46cabb7721372540a1f302efeb1af8fcfc50303a8fd7c50",
    "Gateway": "10.100.4.1",
    "IPAddress": "10.100.4.2",
    "IPPrefixLen": 24,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "DNSNames": [
      "upf2",
      "free5gc-upf",
      "upf.n4.org",
      "f37eda5158ac"
    ]
  }
}

lab2_n6net: {
  "IPAMConfig": {
    "IPv4Address": "10.100.6.100"
  },
  "Links": null,
  "Aliases": [
    "upf2",
    "free5gc-upf"
  ],
  "MacAddress": "cd31:83:93:14:d1:21",
  "DriverOpts": null,
  "GwPriority": 0,
  "NetworkID": "df4d79e339771db3669d0e49614d708855c5ad0d3048122b792b7db98c4ab22",
  "EndpointID": "d923447ccd2e8212a88eeb1d81eda74ebeb9d11c0753b794f28383af9188de",
  "Gateway": "10.100.6.1",
  "IPAddress": "10.100.6.100",
  "IPPrefixLen": 24,
  "IPv6Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "DNSNames": [
    "upf2",
    "free5gc-upf",
    "f37eda5158ac"
  ]
}

```

N3 interface : 10.100.3.100

N4 interface: 10.100.4.2

N6 interface: 10.100.6.100

The ipv4 addresses are like we defined in the 313540040.yaml file, from line 14 to 23.

5. What protocols are used in N2, N3, N4 interfaces? (Describe each)

N2 Interface

Connects: gNB ↔ AMF (Access and Mobility Management Function)

Protocols:

1. NGAP (NG Application Protocol):
 - Runs over SCTP/IP.
 - Responsible for signaling between gNB and AMF.
 - Supports UE context management, NAS transport, paging, PDU session setup, etc.
2. SCTP (Stream Control Transmission Protocol):
 - Reliable transport layer protocol.
 - Allows multi-streaming and multi-homing between gNB and AMF.

N3 Interface

Connects: gNB ↔ UPF (User Plane Function)

Protocols:

1. GTP-U (GPRS Tunneling Protocol – User Plane):
 - Encapsulates user IP packets inside a tunnel.
 - Enables tunneling over IP for data traffic.
 - Runs over UDP/IP (typically port 2152).
2. UDP/IP: Transport protocol for GTP-U.

N4 Interface

Connects: SMF (Session Management Function) ↔ UPF

Protocols:

1. PFCP (Packet Forwarding Control Protocol):
 - Controls and configures rules on UPF (e.g., forwarding rules, buffer settings, usage reporting).
 - Defines session establishment, modification, deletion.
 - Runs over UDP/IP (typically port 8805).
2. UDP/IP: Transport protocol for PFCP messages.