5G Communication and Network Lab

CS462

PROJECT REPORT

Topic : 5G Network deployment - Ansible with Docker, Kubernetes and OVS

Diya Garg 202251044
Garima Singh 202251047
Nimisha Kushwaha 202251079
Pankaj 202251083

Part 1 : 5G Core Network Deployment using Docker + Ansible

This project is about deploying a containerized 5G Core Network setup using Docker and orchestrated by Ansible. Here's what's happening:

Components:

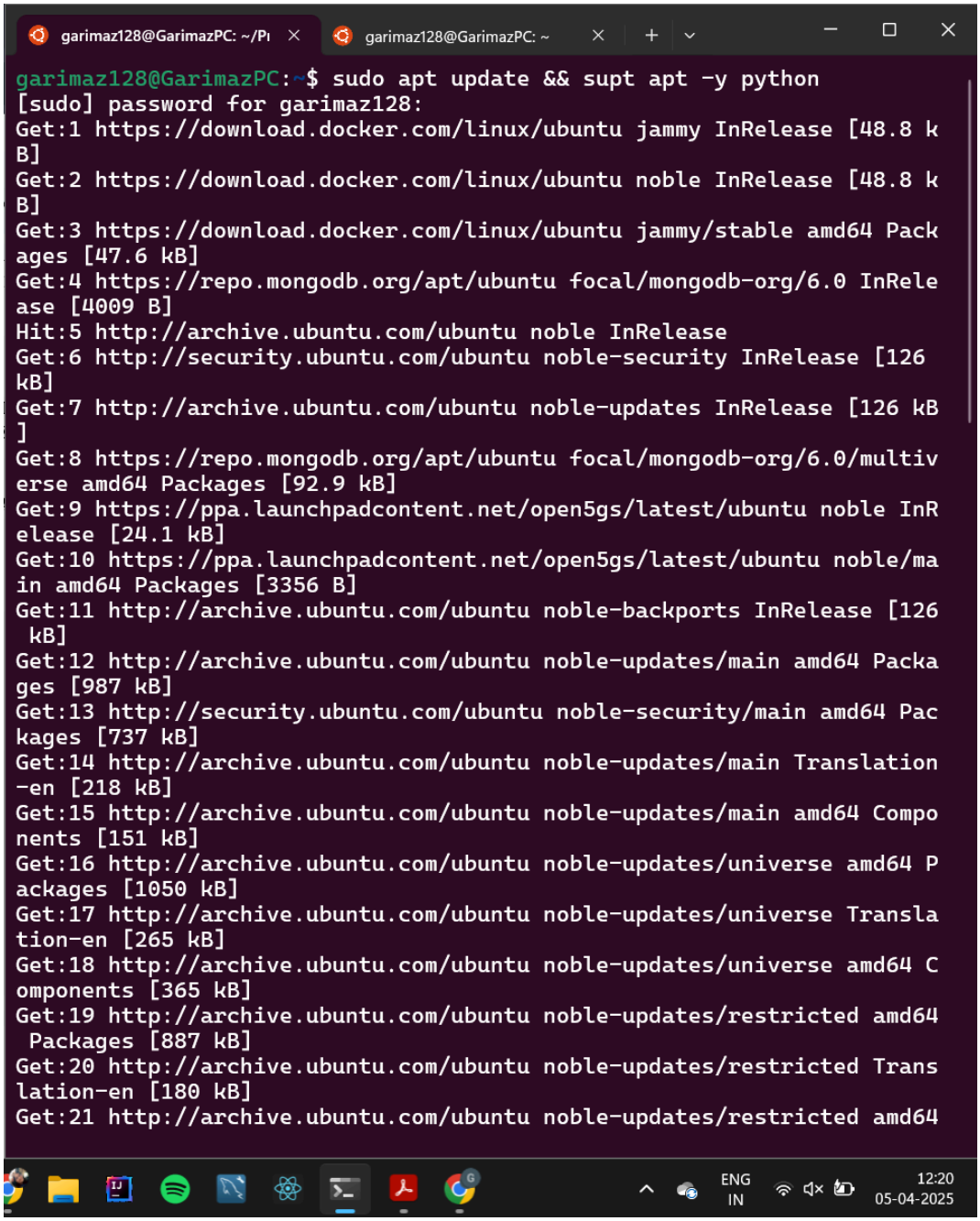
- Containers used :
    - ue: User Equipment simulator
    - enb: eNodeB (radio access part of LTE)
    - webui: Web-based interface for Free5GC
    - pcrf: Policy Control and Charging Rules Function
    - hss: Home Subscriber Server
    - smf: Session Management Function
    - upf: User Plane Function
    - amf: Access and Mobility Function
    - mongodb-s: MongoDB for Free5GC data storage

This structure mimics a Free5GC-based mobile core network where all functional blocks (5GC NFs) are containerized and managed using Docker. This setup enables:

- Simulation of a 5G/4G core network
- Testing service orchestration, routing, and user connectivity
- Easy deployment and teardown using Ansible automation

STEP BY STEP IMPLEMENTATION FOR THE DEPLOYMENT OF 5G NETWORK USING DOCKER

1. INSTALLING AND UPDATING PYTHON AND OTHER RELATED LIBRARIES



2. **Cloning the repository that contains all the yml files and directories (ansible playbook) to get them on the machine and getting into the repository**

```
Project5G-ansible-deployment.git
fatal: destination path 'Project5G-ansible-deployment' already exists
and is not an empty directory.
garimaz128@GarimazPC:~$ cd Project5G-ansible-deployment
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ ls
'Docker deployment'       google6905b9cd306365f9.html
'Kubernetes deployment'   ovs-cni.yml
 README.md                ovs-net-crd.yaml
'Scenario setup'          unix-daemonset.yaml
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo apt -y insta
ll ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ansible is already the newest version (9.2.0+dfsg-0ubuntu5).
The following packages were automatically installed and are no longer
required:
  bridge-utils dns-root-data dnsmasq-base ubuntu-fan
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 72 not upgraded.
```

3. **Checking out the files inside the ‘docker deployment’ folder inside the repo , which contains the yml file to run the ansible playbook (through “ls” commands)**

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ cd Docker deploym
ent
-bash: cd: too many arguments
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ cd "Docker deploy
ment"
garimaz128@GarimazPC:~/Project5G-ansible-deployment/Docker deployment$
 ls
5g-docker-deployment.yml  README.md  ansible_env  images
garimaz128@GarimazPC:~/Project5G-ansible-deployment/Docker deployment$
```

4. **Checking the IP of the interface connected through the internet , which lies under eth0 : using command “ip a”**

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment/Docker deployment$
 ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gr
oup 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
    inet 10.255.255.254/32 brd 10.255.255.254 scope global lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
    link/ether 00:15:5d:a8:17:67 brd ff:ff:ff:ff:ff:ff
    inet 172.31.71.95/20 brd 172.31.79.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fea8:1767/64 scope link
       valid_lft forever preferred_lft forever
```

5. **Deploying 5G core network components using docker compose based on Open5GS**

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment/Docker deployment$
 head -n 5 5g-docker-deployment.yml
version: "3.8"

services:
  mongodb:
    image: mongo:latest
garimaz128@GarimazPC:~/Project5G-ansible-deployment/Docker deployment$
 docker-compose -f 5g-docker-deployment.yml up -d
WARN[0000] /home/garimaz128/Project5G-ansible-deployment/Docker deploy
ment/5g-docker-deployment.yml: the attribute `version` is obsolete, it
 will be ignored, please remove it to avoid potential confusion
[+] Running 6/6
 ✔Container mongodb-s  Started                                    1.9s
 ✔Container amf        Started                                    0.6s
 ✔Container hss        Started                                    0.7s
 ✔Container smf        Started                                    0.6s
 ✔Container upf        Started                                    0.6s
 ✔Container webui      Started                                    1.7s
```

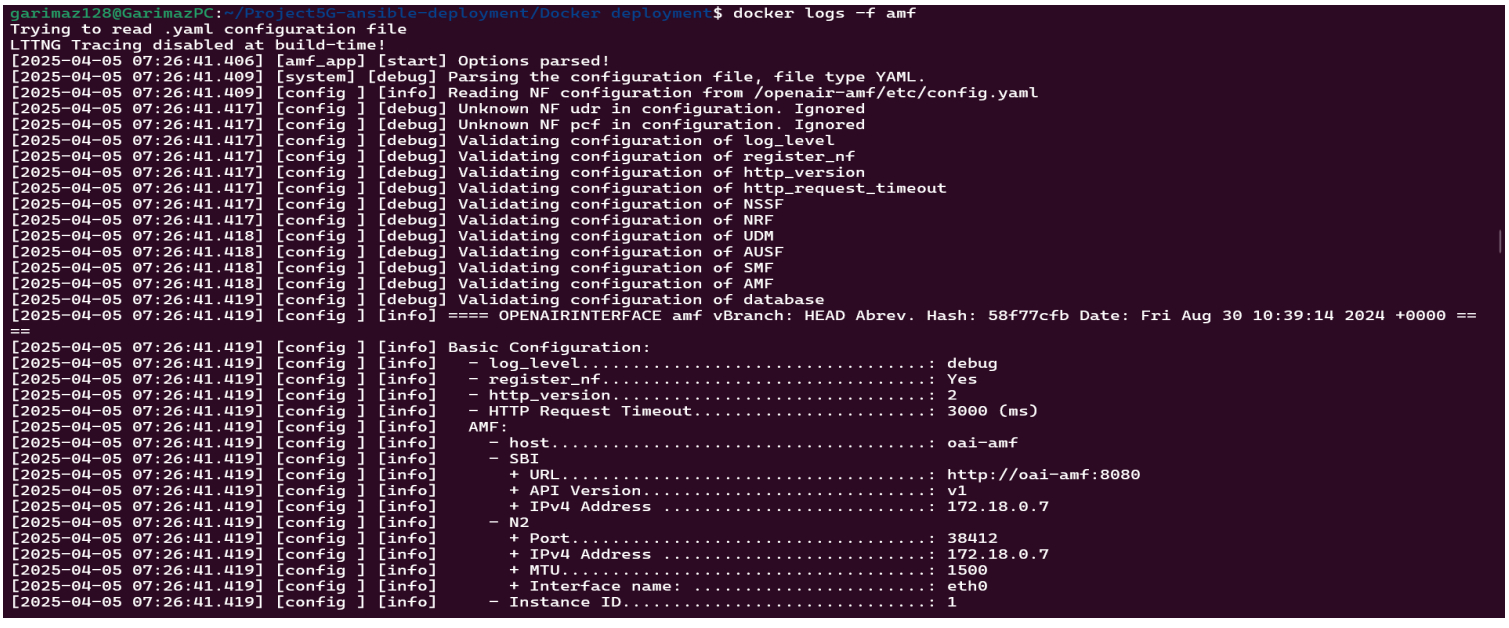
Working of the project till this part : deployment is simulating a 5G core network using docker containers for testing purpose , each launched component is doing the following work :

1. AMF : managing UE connection
2. SMF : handling sessions and allocating IP address
3. UPF : manages data forwarding
4. HSS : stores user related data
5. WebUI: web dashboard to monitor the network performance
6. MongoDB : backend database

6. Checking the container status after getting started with all the containers

```
[+] Running 7/7
 ✔Network dockerdeployment_free5gc-net  Created                0.0s
 ✔Container webui                       St...                  0.5s
 ✔Container smf                         Star...                1.4s
 ✔Container amf                         Star...                1.5s
 ✔Container upf                         Star...                1.0s
 ✔Container hss                         Star...                0.5s
 ✔Container mongodb-s                   Started                0.3s
garimaz128@GarimazPC:~/Project5G-ansible-deployment/Docker deployment$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS                    PORTS
      NAMES
011be53fdce5   mongo:latest                         "docker-entrypoint.s…"   33 seconds ago   Up 32 seconds             27017/tcp
      mongodb-s
7fda817a7f65   oaisoftwarealliance/oai-amf:latest   "/openair-amf/bin/oa…"   33 seconds ago   Up 32 seconds (healthy)   80/tcp, 9090/tcp, 38412/
sctp   amf
ddd5fcfb2df1   oaisoftwarealliance/oai-smf:latest   "/openair-smf/bin/oa…"   33 seconds ago   Up 32 seconds (healthy)   80/tcp, 8080/tcp, 8805/u
dp     smf
```

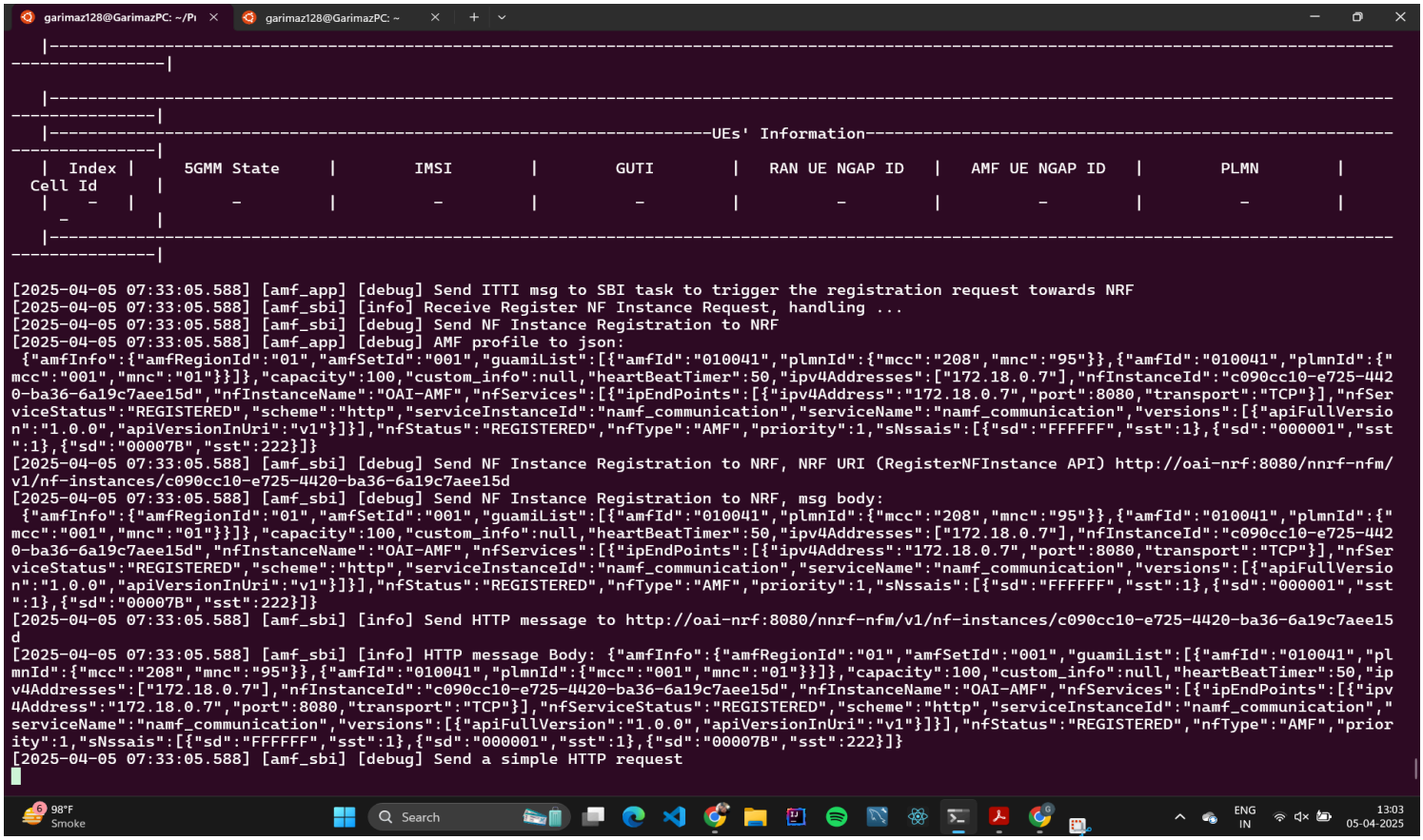
7. Checking the logs for a specific container service (just in case)



```
garimaz128@GarimazPC:~/Project5G-ansible-deployment/Docker deployment$ docker logs -f amf
Trying to read .yaml configuration file
LTTNG Tracing disabled at build-time!
[2025-04-05 07:26:41.406] [amf_app] [start] Options parsed!
[2025-04-05 07:26:41.409] [system] [debug] Parsing the configuration file, file type YAML.
[2025-04-05 07:26:41.409] [config ] [info] Reading NF configuration from /openair-amf/etc/config.yaml
[2025-04-05 07:26:41.417] [config ] [debug] Unknown NF udr in configuration. Ignored
[2025-04-05 07:26:41.417] [config ] [debug] Unknown NF pcf in configuration. Ignored
[2025-04-05 07:26:41.417] [config ] [debug] Validating configuration of log_level
[2025-04-05 07:26:41.417] [config ] [debug] Validating configuration of register_nf
[2025-04-05 07:26:41.417] [config ] [debug] Validating configuration of http_version
[2025-04-05 07:26:41.417] [config ] [debug] Validating configuration of http_request_timeout
[2025-04-05 07:26:41.417] [config ] [debug] Validating configuration of NSSF
[2025-04-05 07:26:41.417] [config ] [debug] Validating configuration of NRF
[2025-04-05 07:26:41.418] [config ] [debug] Validating configuration of UDM
[2025-04-05 07:26:41.418] [config ] [debug] Validating configuration of AUSF
[2025-04-05 07:26:41.418] [config ] [debug] Validating configuration of SMF
[2025-04-05 07:26:41.418] [config ] [debug] Validating configuration of AMF
[2025-04-05 07:26:41.419] [config ] [debug] Validating configuration of database
[2025-04-05 07:26:41.419] [config ] [info] ==== OPENAIRINTERFACE amf vBranch: HEAD Abrev. Hash: 58f77cfb Date: Fri Aug 30 10:39:14 2024 +0000 ==
==
[2025-04-05 07:26:41.419] [config ] [info] Basic Configuration:
[2025-04-05 07:26:41.419] [config ] [info]    - log_level.....................................: debug
[2025-04-05 07:26:41.419] [config ] [info]    - register_nf...................................: Yes
[2025-04-05 07:26:41.419] [config ] [info]    - http_version..................................: 2
[2025-04-05 07:26:41.419] [config ] [info]    - HTTP Request Timeout..........................: 3000 (ms)
[2025-04-05 07:26:41.419] [config ] [info]    AMF:
[2025-04-05 07:26:41.419] [config ] [info]      - host.......................................: oai-amf
[2025-04-05 07:26:41.419] [config ] [info]      - SBI
[2025-04-05 07:26:41.419] [config ] [info]        + URL......................................: http://oai-amf:8080
[2025-04-05 07:26:41.419] [config ] [info]        + API Version..............................: v1
[2025-04-05 07:26:41.419] [config ] [info]        + IPv4 Address ............................: 172.18.0.7
[2025-04-05 07:26:41.419] [config ] [info]      - N2
[2025-04-05 07:26:41.419] [config ] [info]        + Port.....................................: 38412
[2025-04-05 07:26:41.419] [config ] [info]        + IPv4 Address ............................: 172.18.0.7
[2025-04-05 07:26:41.419] [config ] [info]        + MTU......................................: 1500
[2025-04-05 07:26:41.419] [config ] [info]        + Interface name: .........................: eth0
[2025-04-05 07:26:41.419] [config ] [info]      - Instance ID..............................: 1
```

AMF is successfully registering with the NRF, and the logs confirm that:

- **The NF (Network Function) Registration to the NRF is working.**
- **The nfType: AMF and nfStatus: REGISTERED confirm it's alive and talking to the NRF.**



Challenges we faced and working on them :

1. The WebUI is an optional tool for managing UE entries through a graphical interface. The main 5G functions are working fine, and WebUI not running does not impact the actual network behavior or testing.

Solution to this is by modifying the docker-deployment.yml file

2. Image pull failures from private container registries
3. Obsolete version: attribute warning in YAML — just a deprecation warning, but not a blocker , this was again solved by modifying the yml file respectively.

PART 2 : Our addition to the project

Network slicing implementation using open vSwitch (OVS)

Network slicing allows a single physical 5g infrastructure to be split into multiple virtual networks , called slices.

Using OVS will provide :

1. Isolated network traffic between slices
2. Software defined networking
3. Route traffic differently based on slice-specific rules

Step by step implementation for Network slicing by OVS in existing network deployed by docker

1. **Installing open vSwitch and other dependencies :**

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo apt install
openvswitch-switch
[sudo] password for garimaz128:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer
required:
  bridge-utils dns-root-data dnsmasq-base ubuntu-fan
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libevent-2.1-7t64 libunbound8 libxdp1 openvswitch-common
  python3-openvswitch python3-sortedcontainers
Suggested packages:
  openvswitch-doc python-sortedcontainers-doc
The following NEW packages will be installed:
  libevent-2.1-7t64 libunbound8 libxdp1 openvswitch-common
  openvswitch-switch python3-openvswitch python3-sortedcontainers
0 upgraded, 7 newly installed, 0 to remove and 50 not upgraded.
Need to get 4354 kB of archives.
After this operation, 13.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 libevent-2.1-7
t64 amd64 2.1.12-stable-9ubuntu2 [145 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libunb
ound8 amd64 1.19.2-1ubuntu3.4 [442 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 openvs
witch-common amd64 3.3.0-1ubuntu3.2 [1040 kB]
```

2. **Creating OVS bridge :**

In a network slicing scenario, we need a way to separate and route traffic logically inside the same machine.
Verifying if the bridge is created or not by the command “sudo ovs-vsctl show”

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ovs-vsctl ad
d-br br-free5gc
^C2025-04-06T08:45:09Z|00002|fatal_signal|WARN|terminating with signal
 2 (Interrupt)

garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ovs-vsctl sh
ow
c776e342-bd11-4f70-bddf-a54f59c9ae54
    Bridge br-free5gc
        Port br-free5gc
            Interface br-free5gc
                type: internal
    ovs_version: "3.3.0"
```

3. **Creating virtual interfaces for slices**
   Slice A : for regular users
   Slice B : for IoT

   Creating virtual ethernet pairs (veth) , vethA on one side and vethA-peer on the other side , are like 2 ends of wire sending data .
   Doing the same with vethB and vethB-peer

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip link add
vethA type veth peer name vethA-peer
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip link add
vethB type veth peer name vethB-peer
```

4. **Bringing up all the interfaces with the bridges :** commands don’t give any output when they run successfully. They just quietly bring the interfaces up.

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ovs-vsctl ad
d-port br-free5gc vethA
sudo ovs-vsctl add-port br-free5gc vethB
sudo ip link set vethA up
sudo ip link set vethA-peer up
sudo ip link set vethB up
sudo ip link set vethB-peer up
```

5. **Verifying and checking the status of the bridges connected successfully or not**

```
63: vethfc2e12a@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue master br-ec724e057294 state UP mode DEFAULT group default
    link/ether 06:86:00:b2:72:1b brd ff:ff:ff:ff:ff:ff link-netnsid 3
67: vethA-peer@vethA: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop
 state DOWN mode DEFAULT group default qlen 1000
    link/ether 92:3c:cf:3d:f2:a1 brd ff:ff:ff:ff:ff:ff
68: vethA@vethA-peer: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop
 state DOWN mode DEFAULT group default qlen 1000
    link/ether b2:92:84:2a:74:9b brd ff:ff:ff:ff:ff:ff
69: vethB-peer@vethB: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop
 state DOWN mode DEFAULT group default qlen 1000
    link/ether 3e:0c:05:b1:6b:6a brd ff:ff:ff:ff:ff:ff
70: vethB@vethB-peer: <BROADCAST,MULTICAST,M-DOWN> mtu 1500 qdisc noop
 state DOWN mode DEFAULT group default qlen 1000
    link/ether 4e:69:07:69:cc:32 brd ff:ff:ff:ff:ff:ff
```

6. **Testing the connectivity :** assigning IP to test the pink between peers , this confirms packets are travelling over the virtual ethernet pair across OVS bridge

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip addr add
10.0.0.1/24 dev vethA-peer
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip addr add
10.0.0.2/24 dev vethB-peer
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ ping 10.0.0.2 -c
4
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.089 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.061 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3142ms
rtt min/avg/max/mdev = 0.051/0.064/0.089/0.014 ms
```

NOW WE WILL SIMULATE REALISTIC 5G BEHAVIOUR LIKE NETWORK SLICING AND QOS WITH THE HELP OF VLANs OVER OVS

Docker-based 5G core is up and running, adding VLAN-based slicing via OVS allows you to simulate how real telecom networks separate different services

1. **Creating new veth pairs for VLAN separation**

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip link add
slice1-veth type veth peer name slice1-peer
[sudo] password for garimaz128:
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip link add
slice2-veth type veth peer name slice2-peer
```

2. **Attaching OVS bridge to VLAN tags and assigning ip to all of them :**

Lets supposes , slice-1 belongs to VLAN 10 AND slice-2 belongs to VLAN 20

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip link add
slice1-veth type veth peer name slice1-peer
[sudo] password for garimaz128:
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ip link add
slice2-veth type veth peer name slice2-peer
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ sudo ovs-vsctl ad
d-port br-free5gc slice1-veth tag=10
sudo ovs-vsctl add-port br-free5gc slice2-veth tag=20
sudo ip link set slice1-veth up
sudo ip link set slice1-peer up
sudo ip link set slice2-veth up
sudo ip link set slice2-peer up
sudo ip addr add 192.168.10.1/24 dev slice1-peer
sudo ip addr add 192.168.20.1/24 dev slice2-peer
^C2025-04-06T09:20:47Z|00002|fatal_signal|WARN|terminating with signal
 2 (Interrupt)
```

3. **Ping test performance for slice 1 and slice 2 both :**

```
garimaz128@GarimazPC:~/Project5G-ansible-deployment$ ping 192.168.10.1
  # From inside a container/namespace on VLAN 10
PING 192.168.10.1 (192.168.10.1) 56(84) bytes of data.
^C
--- 192.168.10.1 ping statistics ---
25 packets transmitted, 0 received, 100% packet loss, time 24889ms

garimaz128@GarimazPC:~/Project5G-ansible-deployment$ ping 192.168.20.1
PING 192.168.20.1 (192.168.20.1) 56(84) bytes of data.
^C
--- 192.168.20.1 ping statistics ---
14 packets transmitted, 0 received, 100% packet loss, time 13483ms
```

PART 3 : Free5GC Monitoring Dashboard

As the third stage of our project, We implemented a real-time monitoring and visualization system for the Docker-based Free5GC deployment using Prometheus and Grafana. This enabled us to observe the health and resource usage of each 5G core network function, aiding both debugging and optimization.

- Deploying **Free5GC** with Docker (Only AMF , SMF and UPF)
- Installing and configuring **Prometheus (for data scraping)**
- Exposing Free5GC metrics via **Prometheus exporters**
- Visualizing metrics in **Grafana (for visualisation) and cAdvisor (for exporting container metrics)**

STEP 1 : We updated the packages and installed basic tools

sudo apt install -y git curl docker.io docker-compose

And started docker daemon as the base environment.

STEP 2 : We created a project folder and inside it , we made a docker-compose yaml file and a prometheus config file. When done , we launched it.

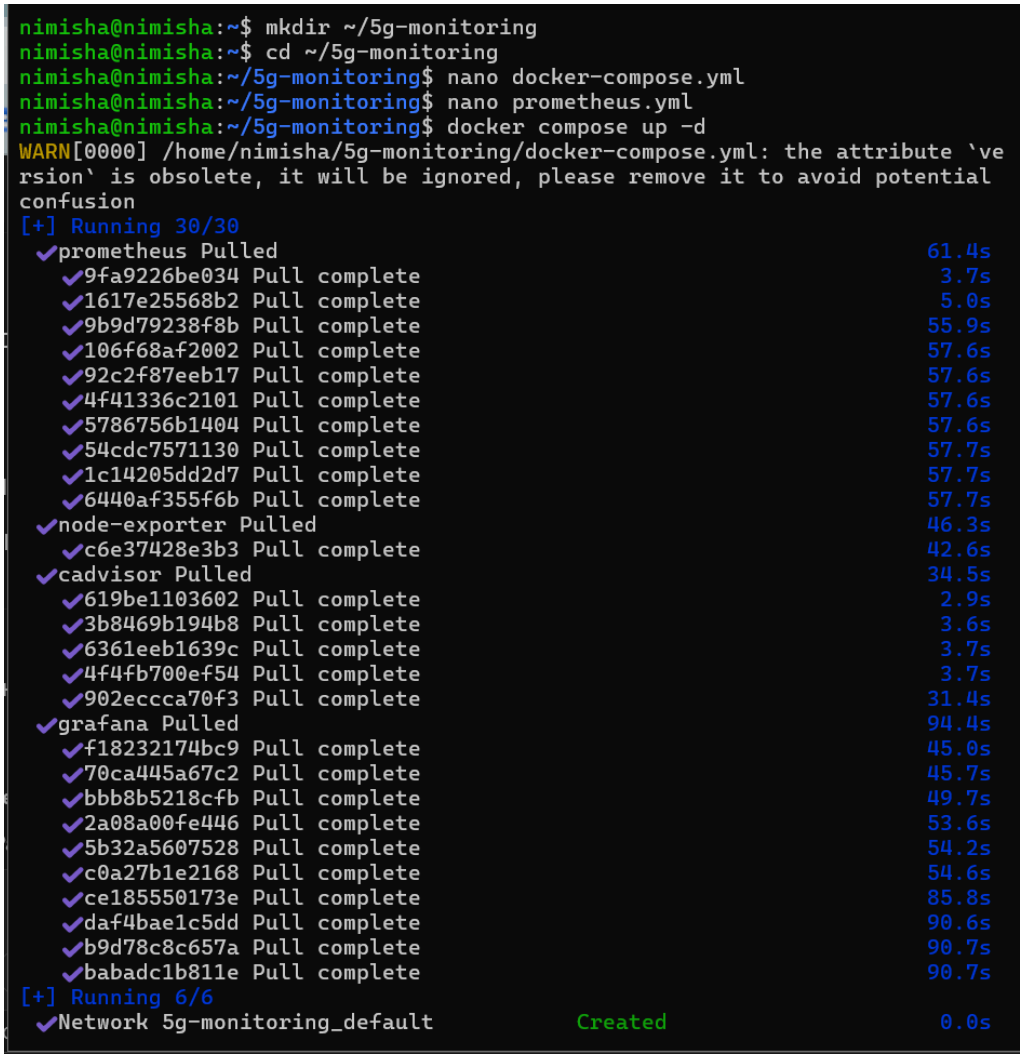
These are the website url :

Prometheus: http://localhost:9090

Grafana: http://localhost:3000

Node Exporter: http://localhost:9100

cAdvisor: http://localhost:8080



STEP 3 : To Monitor Free5GC components , we deploy the containers with labels when running them. This way, Prometheus can collect the necessary metrics for each container.

In these examples, we are labeling each Free5GC component with the respective service label (e.g., free5gc-upf, free5gc-amf, free5gc-smf).

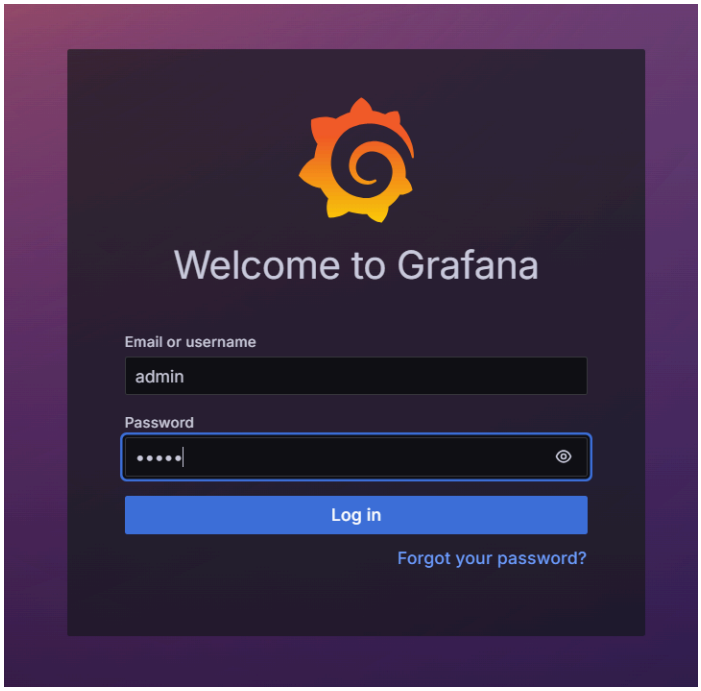
```
 ✔Container cadvisor                    Sta...                    3.1s
 ✔Container prometheus                  S...                      3.1s
 ✔Container grafana                     Star...                   2.8s
 ✔Container node-exporter               Started                   3.1s
nimisha@nimisha:~/5g-monitoring$ docker ps
CONTAINER ID   IMAGE                     COMMAND                  CREATED
      STATUS                  PORTS                NAMES
248fcb8c7b4c   gcr.io/cadvisor/cadvisor   "/usr/bin/cadvisor -…"   4 minutes
 ago   Up 4 minutes (healthy)   0.0.0.0:8080->8080/tcp   cadvisor
f1c5d8d02ce8   prom/node-exporter         "/bin/node_exporter"     4 minutes
 ago   Up 4 minutes            0.0.0.0:9100->9100/tcp   node-exporter
b1fc0c36f2d7   prom/prometheus            "/bin/prometheus --c…"   4 minutes
 ago   Up 4 minutes            0.0.0.0:9090->9090/tcp   prometheus
bd562416bac7   grafana/grafana            "/run.sh"                4 minutes
 ago   Up 4 minutes            0.0.0.0:3000->3000/tcp   grafana
nimisha@nimisha:~/5g-monitoring$ docker run -d --name free5gc-upf --label "f
ree5gc=upf" -p 5000:5000 nginx
7e014bbdc7d70a97c4f39f5c96e031573b03022ff48215790fbc79577232864a
nimisha@nimisha:~/5g-monitoring$ docker run -d --name free5gc-amf --label "f
ree5gc=amf" -p 5001:5001 nginx
83198b4c9b9f7e990420902957d6102695c1301aa26ca1ab2546475d793f08bc
nimisha@nimisha:~/5g-monitoring$ docker run -d --name free5gc-smf --label "f
ree5gc=smf" -p 5002:5002 nginx
3cea80b7e0b5dfd54ac58d66c925f62a6ad5ada6fdf339e7bda836376b6043f9
nimisha@nimisha:~/5g-monitoring$
nimisha@nimisha:~/5g-monitoring$ docker run -d \
  --name=cadvisor \
  --volume=/var/run/docker.sock:/var/run/docker.sock \
  --volume=/sys:/sys \
  --volume=/var/lib/docker/:/var/lib/docker/ \
  -p 8080:8080 \
  google/cadvisor:latest
Unable to find image 'google/cadvisor:latest' locally
latest: Pulling from google/cadvisor
ff3a5c916c92: Pull complete
44a45bb65cdf: Pull complete
0bbe1a2fe2a6: Pull complete
Digest: sha256:815386ebbe9a3490f38785ab11bda34ec8dacf4634af77b8912832d4f85dc
a04
```

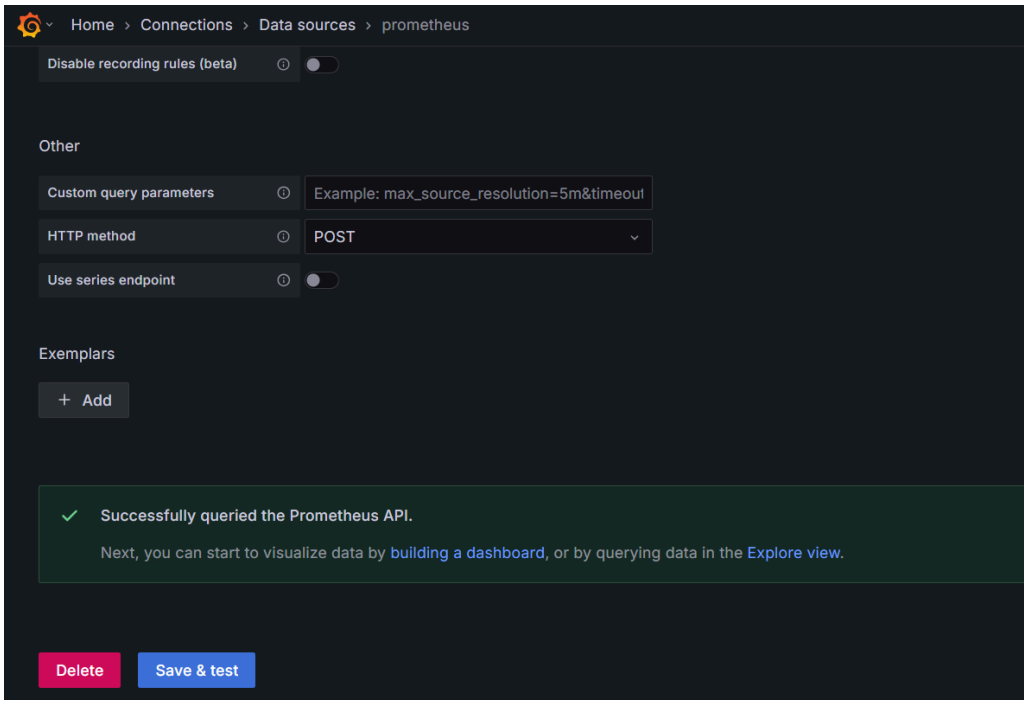
STEP 4 : Now we run the cadvisor which provides insights to container resource usage. In your Prometheus configuration file (`prometheus.yml`), add a job for cAdvisor. This tells Prometheus to scrape metrics from cAdvisor (running on port 8080). And restart it.

```
docker: Error response from daemon: Conflict. The container name "/cadvisor"
 is already in use by container "248fcb8c7b4c78a912faa251e04998fc5261ef18cb3
b96edf0d1f4c544d63603". You have to remove (or rename) that container to be
able to reuse that name.
See 'docker run --help'.
nimisha@nimisha:~/5g-monitoring$ docker stop cadvisor
cadvisor
nimisha@nimisha:~/5g-monitoring$
nimisha@nimisha:~/5g-monitoring$ docker rm cadvisor
cadvisor
nimisha@nimisha:~/5g-monitoring$ docker run -d \
  --name=cadvisor \
  --volume=/var/run/docker.sock:/var/run/docker.sock \
  --volume=/sys:/sys \
  --volume=/var/lib/docker/:/var/lib/docker/ \
  -p 8080:8080 \
  google/cadvisor:latest
1f3762196a34fa097fa583954d098f8d34f1e28e6826bebb805f597a40628d9d
nimisha@nimisha:~/5g-monitoring$
nimisha@nimisha:~/5g-monitoring$ nano prometheus.yml
nimisha@nimisha:~/5g-monitoring$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1372 qdisc mq state UP group
default qlen 1000
    link/ether 00:15:5d:b4:c0:fb brd ff:ff:ff:ff:ff:ff
    inet 172.24.175.73/20 brd 172.24.175.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:feb4:c0fb/64 scope link
       valid_lft forever preferred_lft forever
nimisha@nimisha:~/5g-monitoring$ nano prometheus.yml
nimisha@nimisha:~/5g-monitoring$ docker restart prometheus
prometheus
nimisha@nimisha:~/5g-monitoring$
```

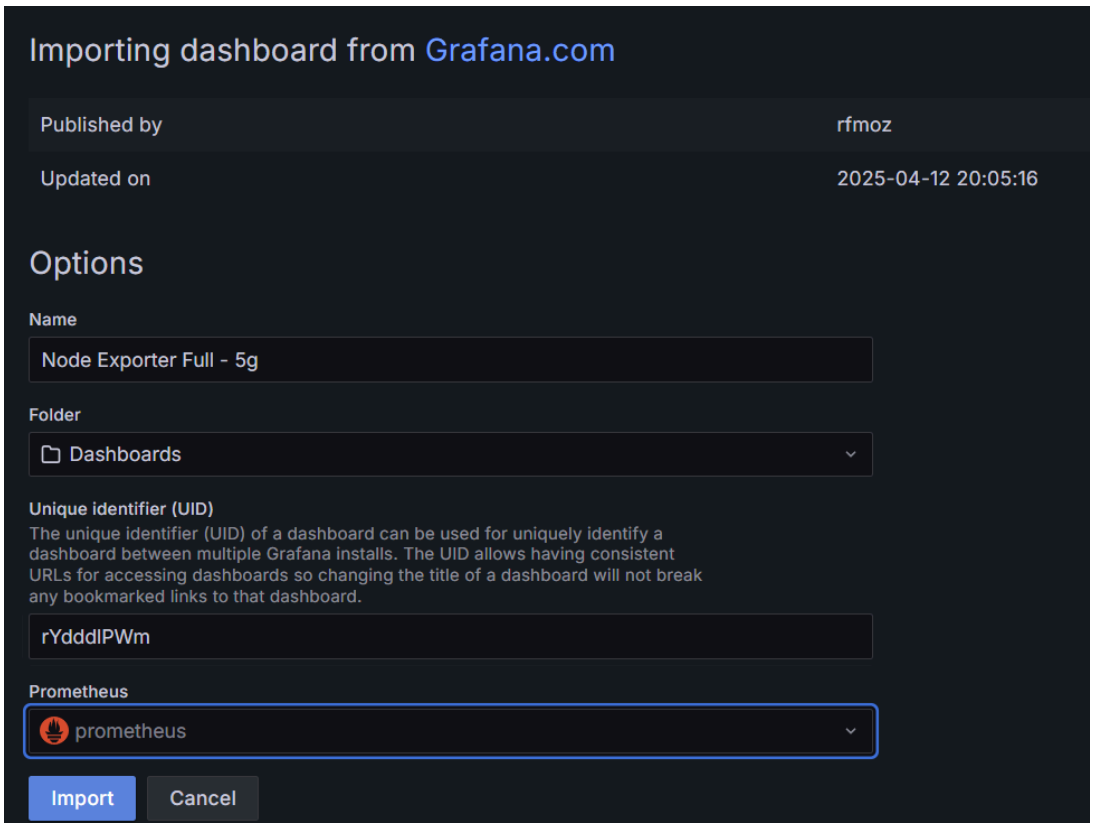
Since, prometheus was not working as a localhost, we changed it to my local host ip and restarted it.

STEP 5 : Then, We logged into Grafana with our login credentials and added prometheus as a data source. And save and test if it was working.

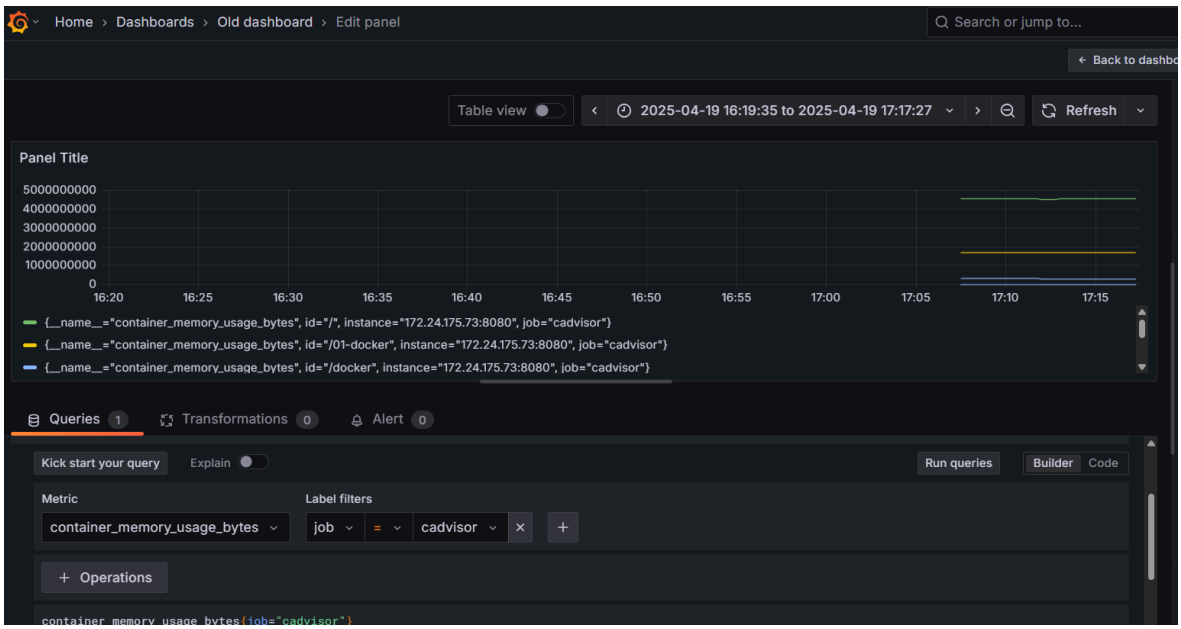




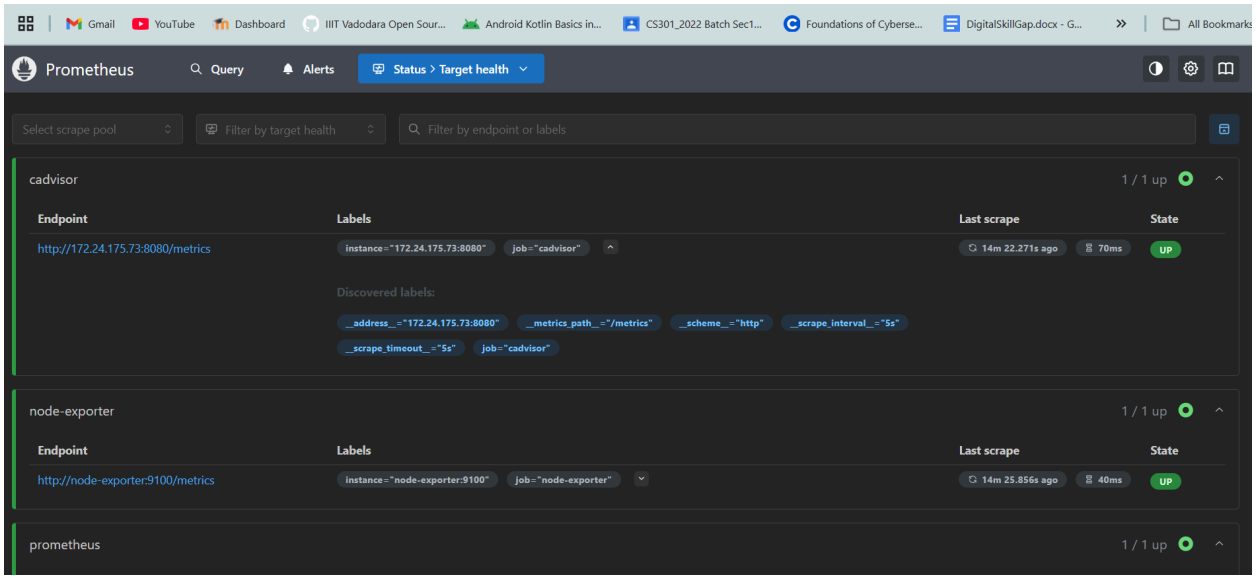
STEP 6 : Importing a dashboard in Grafana with ID 1860 which is actually very good for cAdvisor.



STEP 7 : Use grafana to add Prometheus as the data source and enter queries to check metrics for the containers that are up and running. In the below image, we can see the metric memory usage :



Note. – 1. We are checking if Prometheus is scraping correctly. And cAdvisor is up.



We can see the metrics here.

Note 2. This is the web UI of cAdvisor (up and running.)