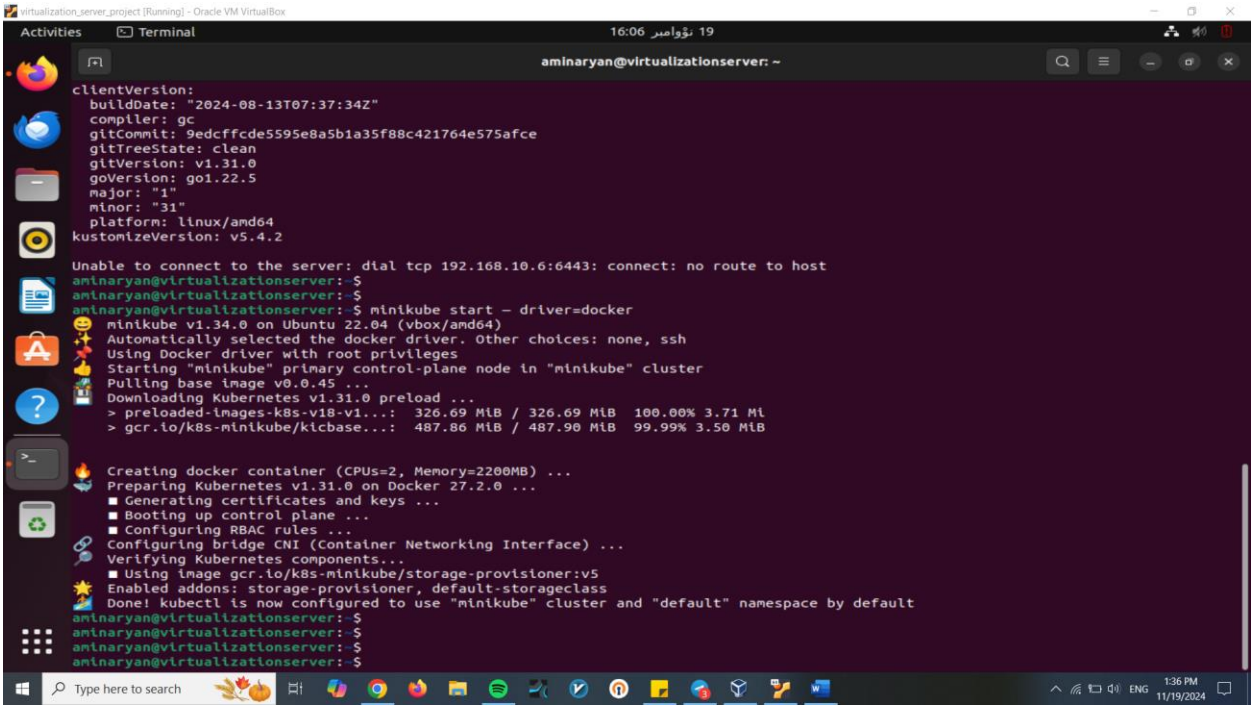
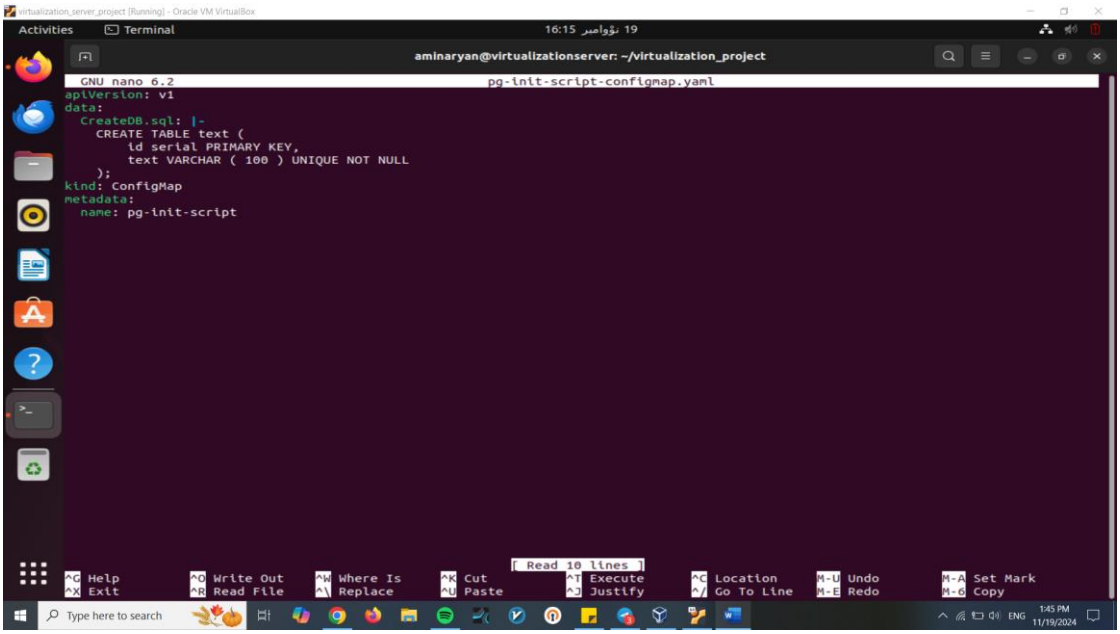
Project 1

First we install minikube:

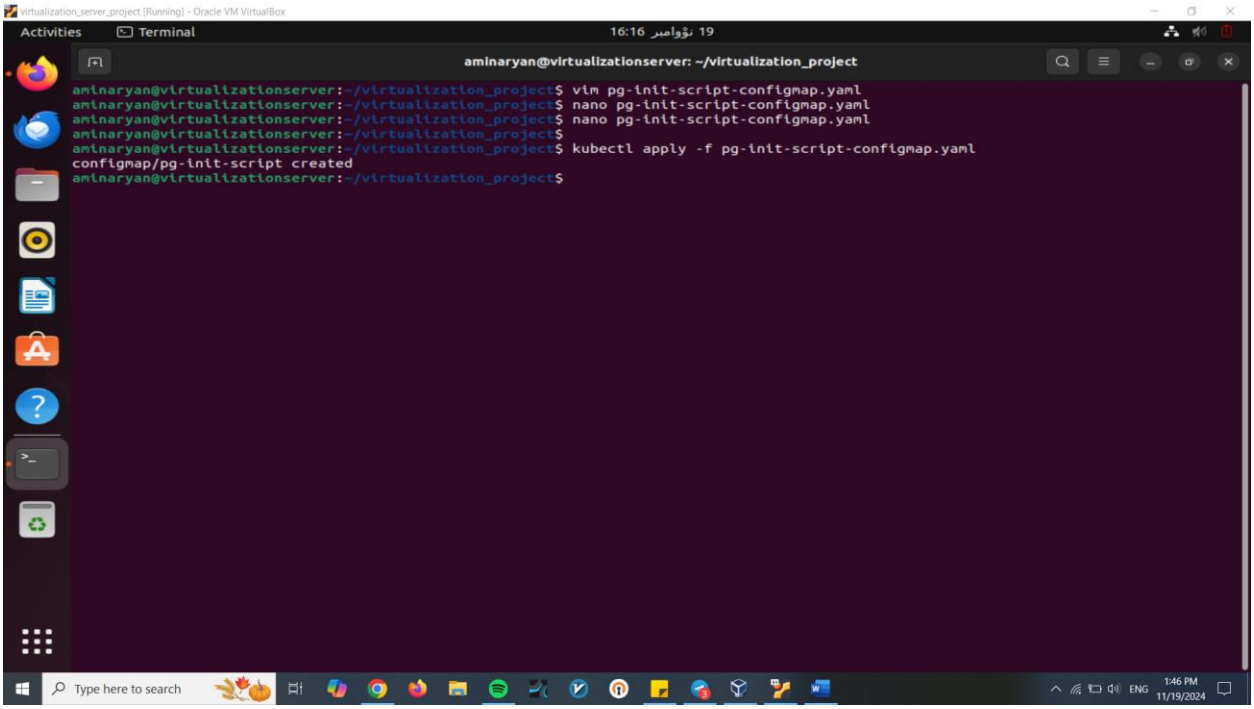


Database

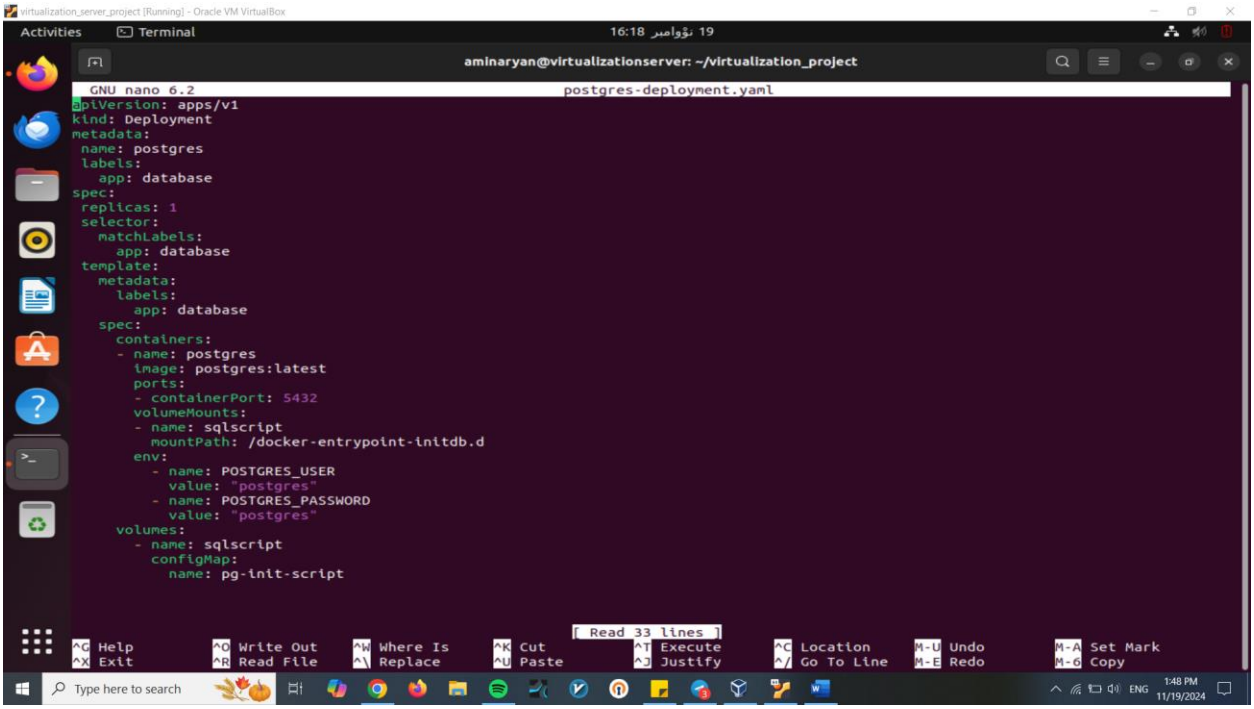
Now we try to deploy postgres to minikube. First we write a configmap for the table:



Now we apply it:



Now we write a postgres deployment:

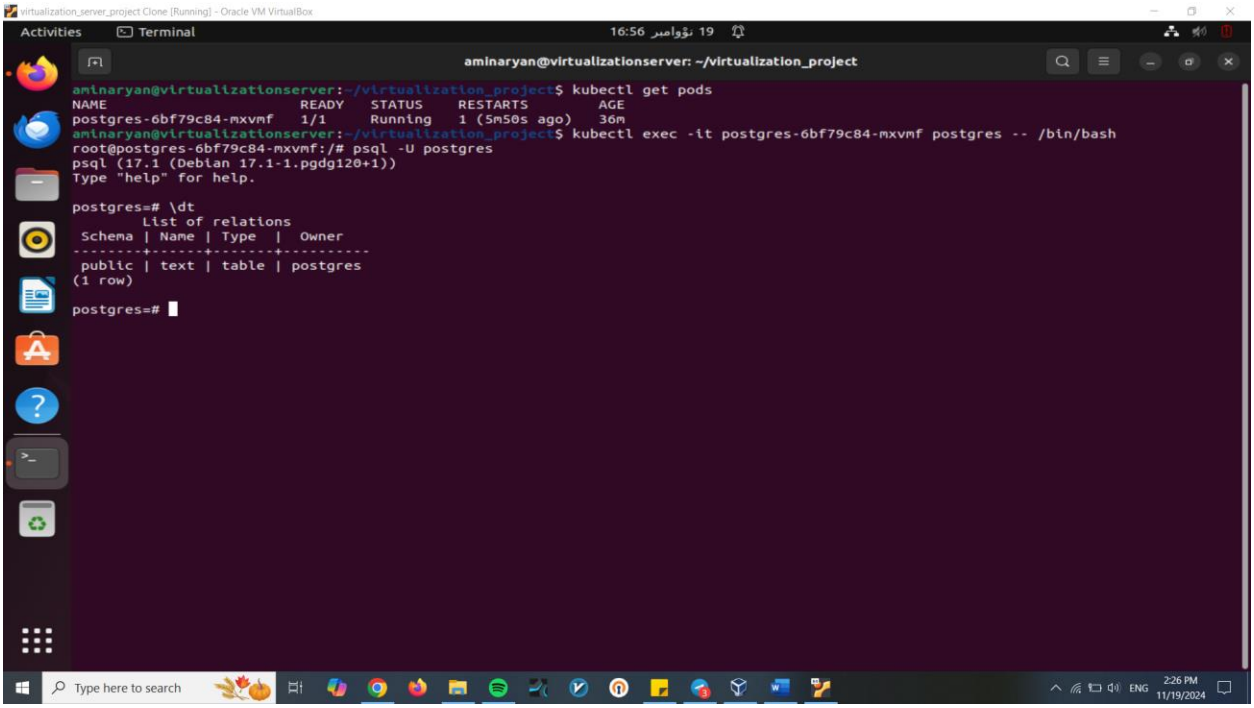


and now we apply it:

```
aminaryan@virtualizationserver:~/virtualization_project$ vim pg-init-script-configmap.yaml
aminaryan@virtualizationserver:~/virtualization_project$ nano pg-init-script-configmap.yaml
aminaryan@virtualizationserver:~/virtualization_project$ nano pg-init-script-configmap.yaml
aminaryan@virtualizationserver:~/virtualization_project$
aminaryan@virtualizationserver:~/virtualization_project$ kubectl apply -f pg-init-script-configmap.yaml
configmap/pg-init-script created
aminaryan@virtualizationserver:~/virtualization_project$ kubectl get configmap pg-init-script -o yaml
apiVersion: v1
data:
  CreateDB.sql: |-
    CREATE TABLE text (
        id serial PRIMARY KEY,
        text VARCHAR ( 100 ) UNIQUE NOT NULL
    );
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"CreateDB.sql":"CREATE TABLE text (\n    id serial PRIMARY KEY,\n    text VARCHAR ( 100 ) UNIQUE NO
T NULL\n);"},"kind":"ConfigMap","metadata":{"annotations":{},"name":"pg-init-script","namespace":"default"}}
  creationTimestamp: "2024-11-19T12:46:28Z"
  name: pg-init-script
  namespace: default
  resourceVersion: "876"
  uid: e37145bd-1599-4b5b-a500-ca6d7820a9dc
aminaryan@virtualizationserver:~/virtualization_project$ nano postgres-deployment.yaml
aminaryan@virtualizationserver:~/virtualization_project$
aminaryan@virtualizationserver:~/virtualization_project$ nano postgres-deployment.yaml
aminaryan@virtualizationserver:~/virtualization_project$
aminaryan@virtualizationserver:~/virtualization_project$ kubectl apply -f postgres-deployment.yaml
deployment.apps/postgres created
aminaryan@virtualizationserver:~/virtualization_project$ kubectl get pods
NAME                        READY   STATUS              RESTARTS   AGE
postgres-6bf79c84-mxvmf     0/1     ContainerCreating   0          11s
aminaryan@virtualizationserver:~/virtualization_project$
```

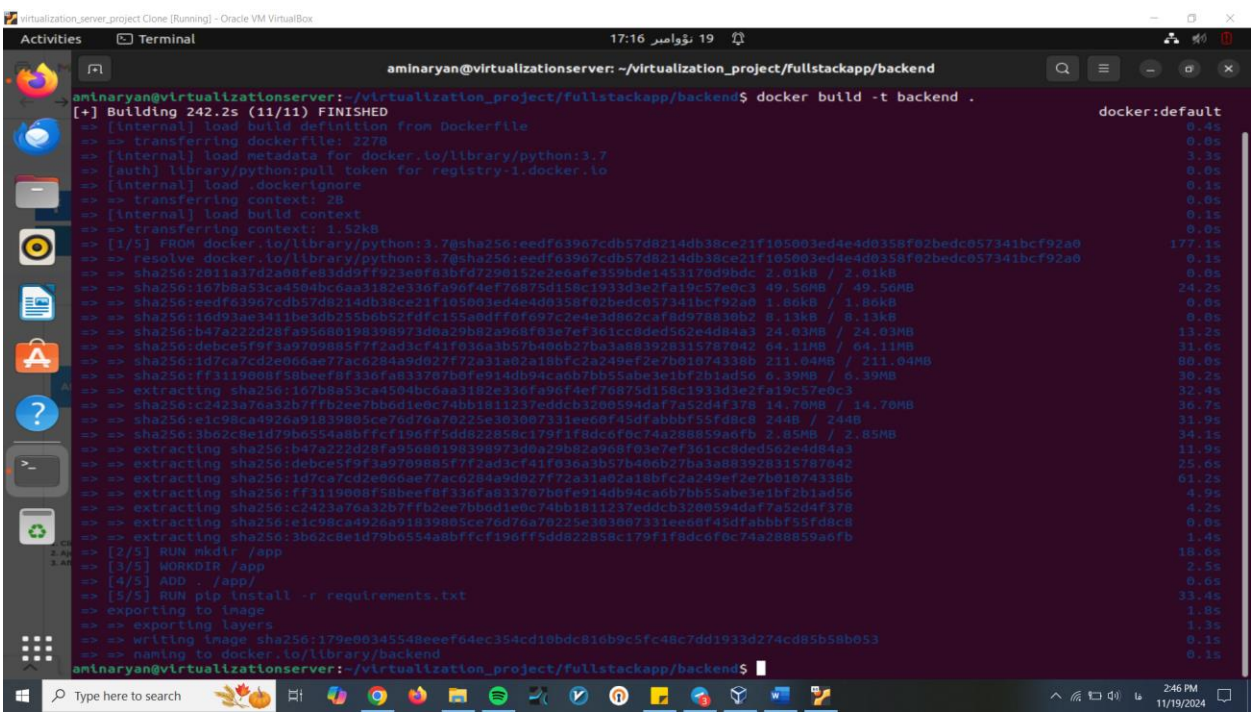
Now because we do not want users to communicate with the database directly, and want the database to communicate with the backend, we apply this service:

```
  GNU nano 6.2                                pg-service.yaml
apiVersion: v1
kind: Service
metadata:
 name: pg-service
 labels:
   app: database
spec:
 type: ClusterIP
 ports:
 - port: 5432
 selector:
   app: database
```

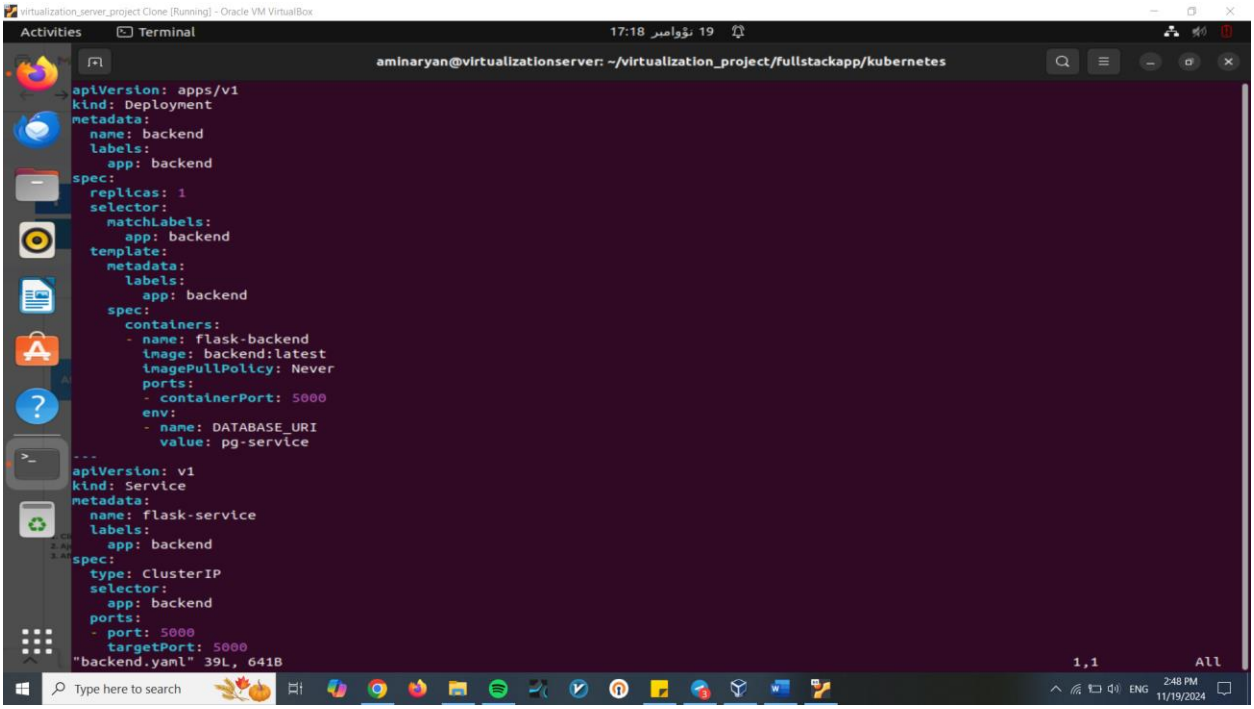


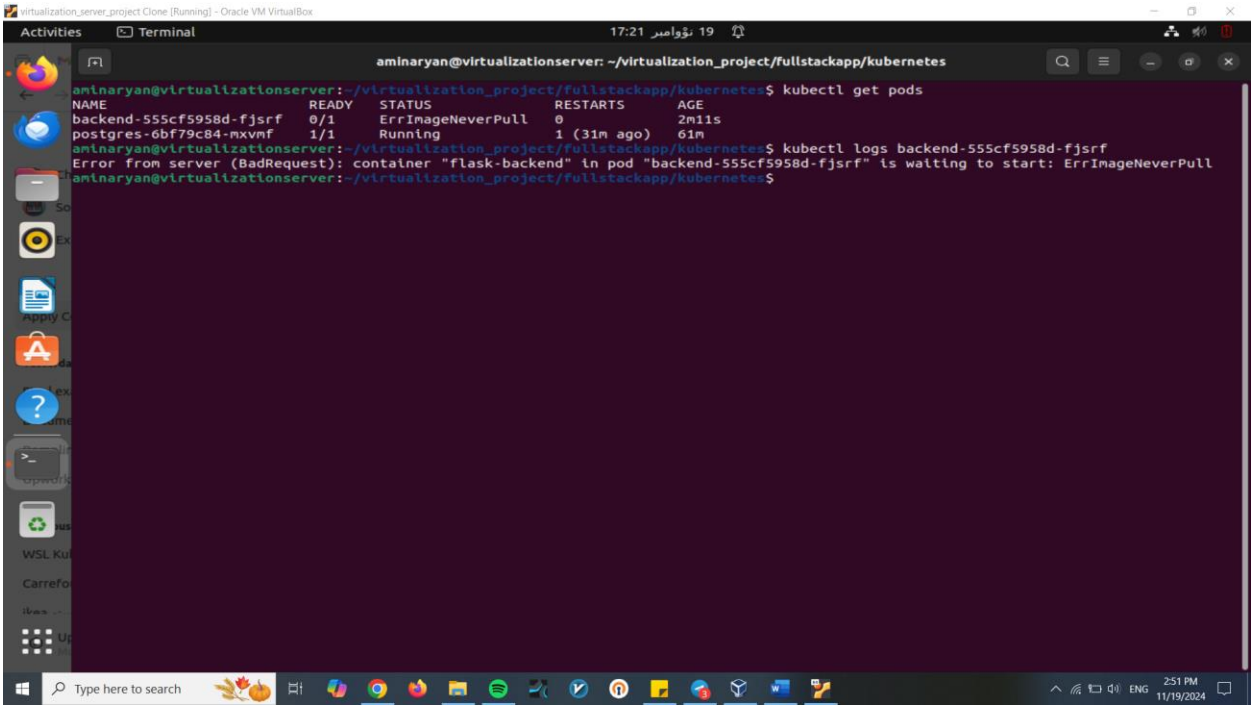
Backend

Now we try to build the docker image for the backend:



And now we apply the backend deployment and service:





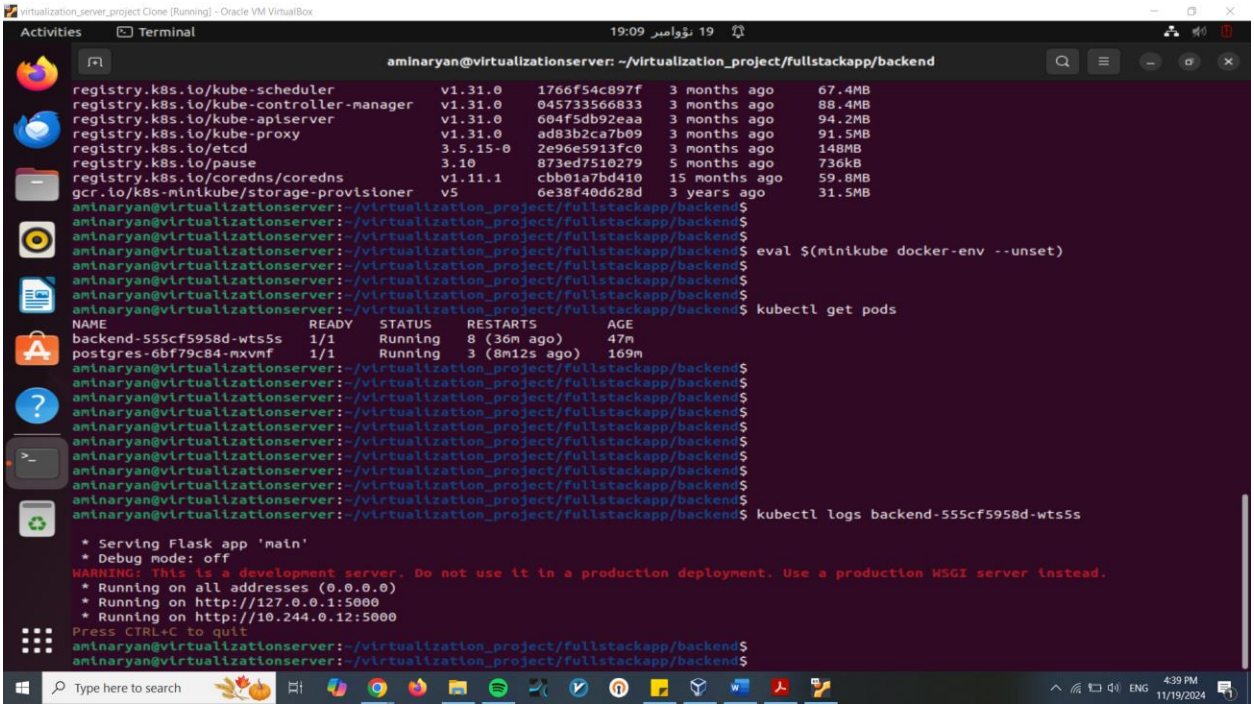
We see that we have an error. It is because we have to load the built image into minikube:

```
aminaryan@virtualizationserver: ~/virtualization_project/fullstackapp/backend
[+] Building 2.4s (11/11) FINISHED                                          docker:default
 => [internal] load build definition from Dockerfile                                  0.2s
 => => transferring dockerfile: 227B                                                  0.0s
 => [internal] load metadata for docker.io/library/python:3.7                         1.6s
 => [auth] library/python:pull token for registry-1.docker.io                         0.0s
 => [internal] load .dockerignore                                                     0.1s
 => => transferring context: 2B                                                       0.0s
 => [internal] load build context                                                     0.1s
 => => transferring context: 94B                                                      0.0s
 => [1/5] FROM docker.io/library/python:3.7@sha256:eedf63967cdb57d8214db38ce21f105003ed4e4d0358f02bedc057341bcf92a0  0.0s
 => CACHED [2/5] RUN mkdir /app                                                       0.0s
 => CACHED [3/5] WORKDIR /app                                                         0.0s
 => CACHED [4/5] ADD . /app/                                                          0.0s
 => CACHED [5/5] RUN pip install -r requirements.txt                                  0.0s
 => exporting to image                                                                0.0s
 => => exporting layers                                                               0.0s
 => => writing image sha256:179e00345548eeef64ec354cd10bdc816b9c5fc48c7dd1933d274cd85b58b053  0.0s
 => => naming to docker.io/library/backend:latest                                     0.0s
aminaryan@virtualizationserver:~/virtualization_project/fullstackapp/backend$
aminaryan@virtualizationserver:~/virtualization_project/fullstackapp/backend$
aminaryan@virtualizationserver:~/virtualization_project/fullstackapp/backend$ minikube image load backend:latest
```

Now we try again:

```
aminaryan@virtualizationserver: ~/virtualization_project/fullstackapp/kubernetes
aminaryan@virtualizationserver:~/virtualization_project/fullstackapp/kubernetes$ kubectl get pods
NAME                        READY   STATUS    RESTARTS      AGE
backend-555cf5958d-fjsrf    0/1     Error     5 (91s ago)   40m
postgres-6bf79c84-mxvmf     1/1     Running   2 (12m ago)   99m
aminaryan@virtualizationserver:~/virtualization_project/fullstackapp/kubernetes$ kubectl logs backend-555cf5958d-fjsrf
Traceback (most recent call last):
  File "/app/main.py", line 13, in <module>
    db.init_app(app)
  File "/usr/local/lib/python3.7/site-packages/flask_sqlalchemy/extension.py", line 256, in init_app
    "A 'SQLAlchemy' instance has already been registered on this Flask app."
RuntimeError: A 'SQLAlchemy' instance has already been registered on this Flask app. Import and use that instance instead.
aminaryan@virtualizationserver:~/virtualization_project/fullstackapp/kubernetes$
```

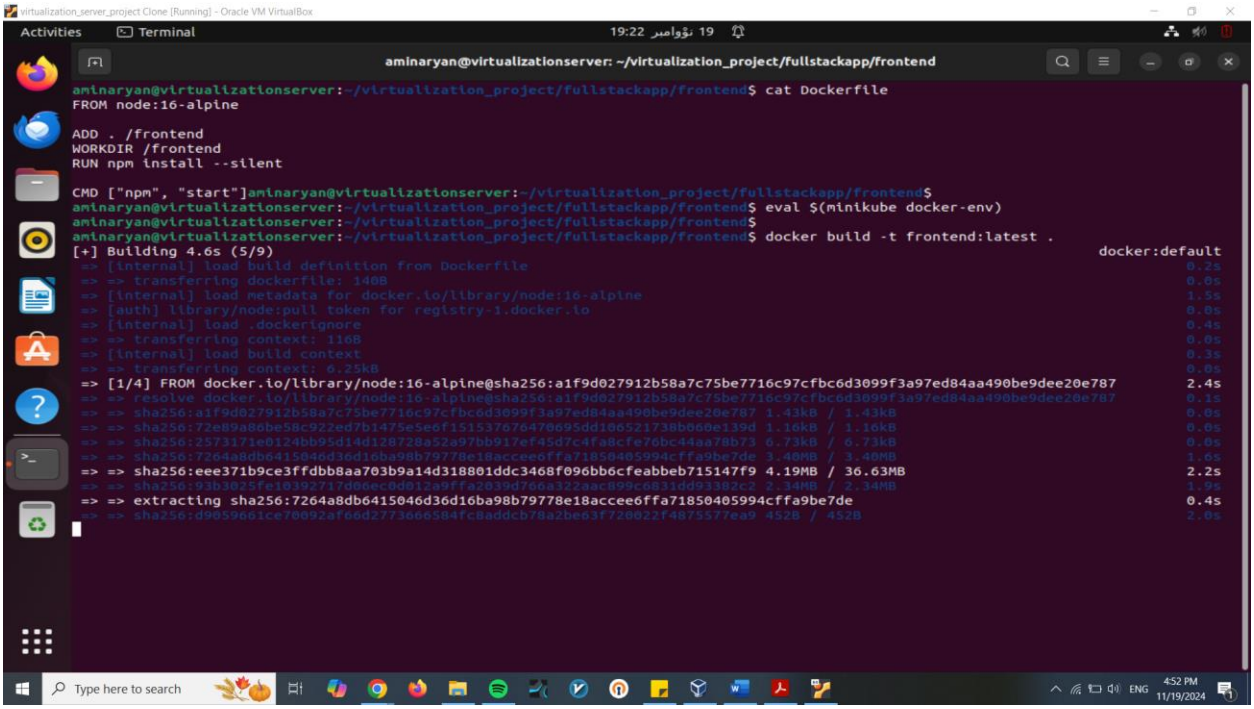
As you can see the image error has been fixed. Now we have a Runtime error. We will fix the code (remove line 13 because it is not needed) and try again:



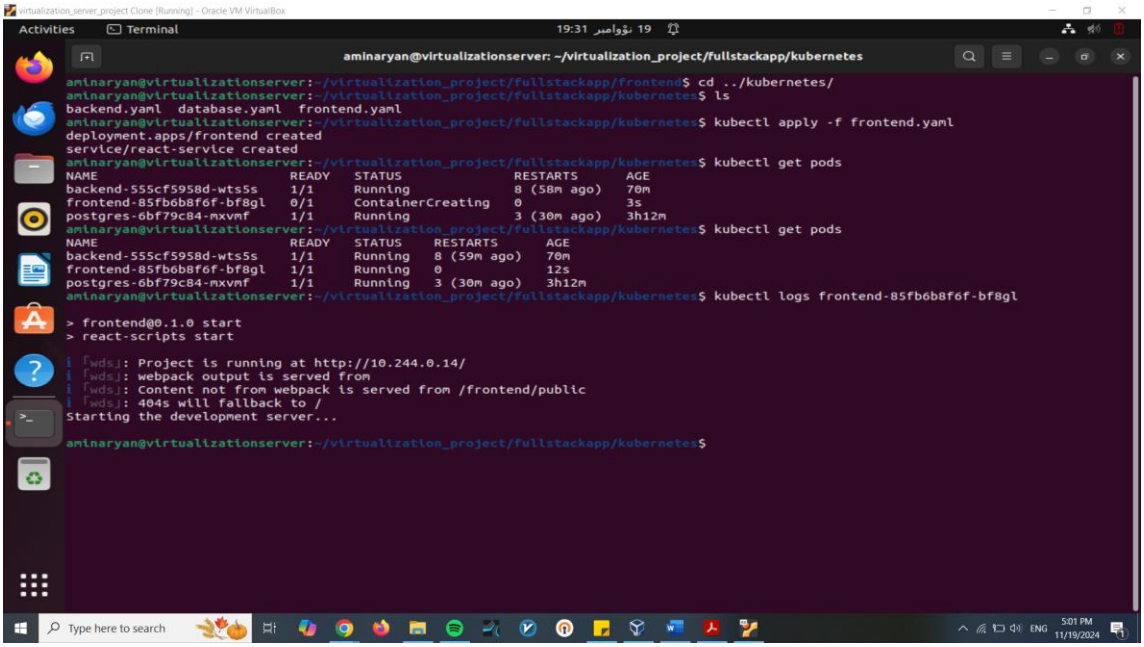
As you can see, the backend has been deployed successfully.

Frontend

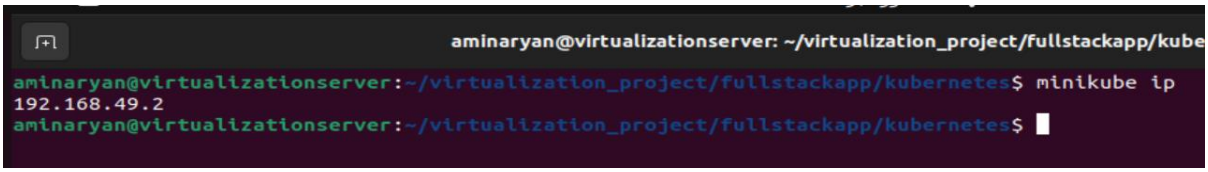
Now we try to deploy the frontend. First we build an image from the frontend project:

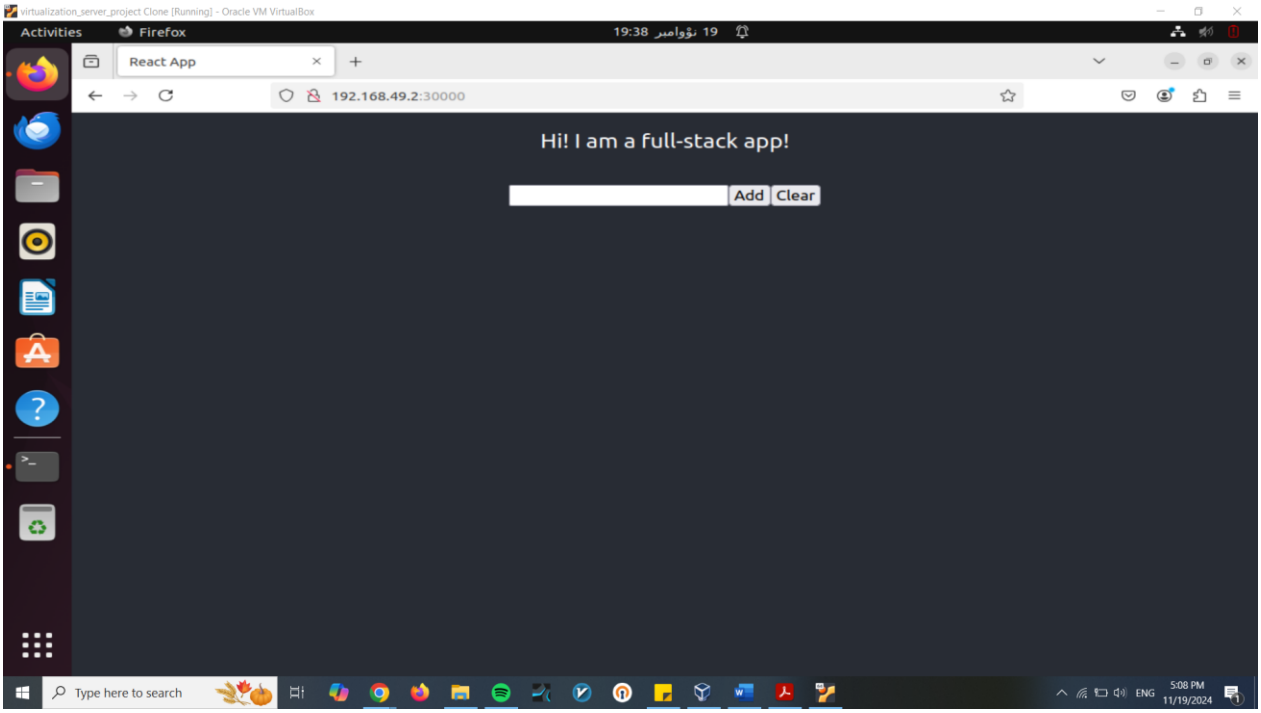


Now that we have our image in minikube, we deploy the frontend into Kuber:



We successfully deployed the project. Now we are able to open the project on browser using minikube ip:





Project 2

We go step by step with the readme in github project. First we pull images from new docker repositories:

```
docker pull mysql:5.7
docker pull oaisoftwarealliance/oai-amf:latest
docker pull oaisoftwarealliance/oai-nrf:latest
docker pull oaisoftwarealliance/oai-smf:latest
docker pull oaisoftwarealliance/oai-spgwu-tiny:latest

docker pull oaisoftwarealliance/oai-gnb:develop
docker pull oaisoftwarealliance/oai-nr-ue:develop
```

```
root@DESKTOP-DPA9QIC: ~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator
90986bb8de6e: Pull complete
ae71319cb779: Pull complete
ffc89e9dfd88: Pull complete
43d05e938198: Pull complete
064b2d298fba: Pull complete
df9a4d85569b: Pull complete
Digest: sha256:4bc6bc963e6d8443453676cae56536f4b8156d78bae03c0145cbe47c2aad73bb
Status: Downloaded newer image for mysql:5.7
docker.io/library/mysql:5.7
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker pull oaisoftwarealliance/oai-amf:latest
latest: Pulling from oaisoftwarealliance/oai-amf
d67238ebdaf8: Pull complete
Digest: sha256:a61bfdf1c4f2b03478a568814e0ae6bd7223fa39a07e7c9ce6e6cd9c49e50cd1
Status: Downloaded newer image for oaisoftwarealliance/oai-amf:latest
docker.io/oaisoftwarealliance/oai-amf:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker pull oaisoftwarealliance/oai-nrf:latest
latest: Pulling from oaisoftwarealliance/oai-nrf
8066dc26d515: Pull complete
Digest: sha256:ee6961e04c3e86d25fc7060702c005e4f465568d7df622004aa985fecf57518e
Status: Downloaded newer image for oaisoftwarealliance/oai-nrf:latest
docker.io/oaisoftwarealliance/oai-nrf:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker pull oaisoftwarealliance/oai-smf:latest
latest: Pulling from oaisoftwarealliance/oai-smf
e72f4564fe47: Pull complete
Digest: sha256:734bc99866aa7c9d600ece3617a9d7e577395eb9754088ee1c2226935e8dc1ff
Status: Downloaded newer image for oaisoftwarealliance/oai-smf:latest
docker.io/oaisoftwarealliance/oai-smf:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker pull oaisoftwarealliance/oai-spgwu-tiny:latest
latest: Pulling from oaisoftwarealliance/oai-spgwu-tiny
04c26aef1f80: Pull complete
Digest: sha256:f56d80a4976e2cd083e2a5968751aaa9c35b250c141bc70c3b7749b8e9cd4f8e
Status: Downloaded newer image for oaisoftwarealliance/oai-spgwu-tiny:latest
docker.io/oaisoftwarealliance/oai-spgwu-tiny:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker pull oaisoftwarealliance/oai-gnb:develop
develop: Pulling from oaisoftwarealliance/oai-gnb
7646c8da3324: Downloading [================>                                  ]  9.919MB/29.53MB
9fc0a6f7134b: Downloading [====>                                              ]  7.006MB/83.91MB
1b7f4066350f: Download complete
75adaefd1a1d: Download complete
1a94f97f2609: Waiting
441e164ecdb3: Waiting
eea8bdaf7d7b: Waiting
830069b6db79: Waiting
dcc8bd26bed2: Waiting
a596ab0d3f28: Waiting
63ec0f6b8480: Waiting
b7f96f197743: Waiting
4f4fb700ef54: Waiting
eb2f8229d375: Waiting
```

And then we retag them to work with docker-compose:

```
root@DESKTOP-DPA9QIC: ~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker image tag oaisoftwarealliance/oai-amf:latest oai-amf:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker image tag oaisoftwarealliance/oai-nrf:latest oai-nrf:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker image tag oaisoftwarealliance/oai-smf:latest oai-smf:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker image tag oaisoftwarealliance/oai-spgwu-tiny:latest oai-spgwu-tiny:latest
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker image tag oaisoftwarealliance/oai-gnb:develop oai-gnb:develop
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker image tag oaisoftwarealliance/oai-nr-ue:develop oai-nr-ue:develop
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator#
```

After that, we use docker-compose to deploy the project:

```
root@DESKTOP-DPA9QIC: ~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker-compose up -d mysql oai-nrf oai-amf oai-smf oai-spgwu oai-ext-dn
[+] Running 1/4
 ⠿ oai-ext-dn 1 layers [⠿]      0B/0B      Pulled                                                                                                    2.1s
   ⠿ 7c457f213c76 Already exists                                                                                                                   0.0s
[+] Running 8/8
 ⠿ Network rfsim5g-oai-traffic_net-net  Created                                                                                                     0.7s
 ⠿ Network rfsim5g-oai-public-net       Created                                                                                                     0.7s
 ⠿ Container rfsim5g-mysql              Started                                                                                                     1.8s
 ⠿ Container rfsim5g-oai-nrf            Started                                                                                                     1.9s
 ⠿ Container rfsim5g-oai-spgwu          Started                                                                                                     4.1s
 ⠿ Container rfsim5g-oai-amf            Started                                                                                                     4.0s
 ⠿ Container rfsim5g-oai-smf            Started                                                                                                     4.0s
 ⠿ Container rfsim5g-oai-ext-dn         Started                                                                                                     4.9s
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator# docker-compose ps -a
NAME                 IMAGE                  COMMAND                  SERVICE             CREATED             STATUS                          PORTS
rfsim5g-mysql        mysql:5.7              "docker-entrypoint.s…"   mysql               25 seconds ago      Up 24 seconds (health: starting)   3306/tcp, 33060/tcp
rfsim5g-oai-amf      oai-amf:latest         "/openair-amf/bin/oa…"   oai-amf             25 seconds ago      Up 22 seconds (health: starting)   80/tcp, 9090/tcp, 38412/sctp
rfsim5g-oai-ext-dn   ubuntu:bionic          "/bin/bash -c ' apt …"   oai-ext-dn          24 seconds ago      Up 20 seconds (health: starting)
rfsim5g-oai-nrf      oai-nrf:latest         "/openair-nrf/bin/oa…"   oai-nrf             25 seconds ago      Up 24 seconds (health: starting)   80/tcp, 9090/tcp
rfsim5g-oai-smf      oai-smf:latest         "/bin/bash -c '/open…"   oai-smf             25 seconds ago      Up 22 seconds (health: starting)   80/tcp, 8080/tcp, 8805/udp
rfsim5g-oai-spgwu    oai-spgwu-tiny:latest  "python3 /openair-sp…"   oai-spgwu           25 seconds ago      Up 22 seconds (health: starting)   2152/udp, 8805/udp
root@DESKTOP-DPA9QIC:~/task2/openairinterface5g/ci-scripts/yaml_files/5g_rfsimulator#
```

```
rfsim5g-public: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.71.129  netmask 255.255.255.192  broadcast 192.168.71.191
        inet6 fe80::42:65ff:fe32:2edc  prefixlen 64  scopeid 0x20<link>
        ether 02:42:65:32:2e:dc  txqueuelen 0  (Ethernet)
        RX packets 81  bytes 9190 (9.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 101  bytes 13730 (13.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

rfsim5g-traffic: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.72.129  netmask 255.255.255.192  broadcast 192.168.72.191
        inet6 fe80::42:48ff:feea:2190  prefixlen 64  scopeid 0x20<link>
        ether 02:42:48:ea:21:90  txqueuelen 0  (Ethernet)
        RX packets 6487  bytes 379441 (379.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12638  bytes 30444662 (30.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

As you can see, the project is being run with docker-compose.

In regarding to installing the docker-compose, I already had docker-compose on my system but this is how it can be installed:

```
1. sudo curl -L
   "https://github.com/docker/compose/releases/download/1.29.2/docker-
   compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

2. sudo chmod +x /usr/local/bin/docker-compose
```

Project 3

First we install microk8s on ubuntu:

```
root@new-server: ~

root@new-server:~#
root@new-server:~#
root@new-server:~# sudo snap install microk8s --classic
2024-11-25T12:28:57Z INFO Waiting for automatic snapd restart...
microk8s (1.31/stable) v1.31.2 from Canonical installed
root@new-server:~#
root@new-server:~#
root@new-server:~# sudo microk8s status
microk8s is not running. Use microk8s inspect for a deeper inspection.
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~# sudo microk8s status
microk8s is not running. Use microk8s inspect for a deeper inspection.
root@new-server:~# Run ps -ef | grep kubelet
Command 'Run' not found, did you mean:
  command 'bun' from snap bun-js (1.1.36)
  command 'zun' from deb python3-zunclient (4.4.0-0ubuntu1)
See 'snap info <snapname>' for additional versions.
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~# microk8s status
microk8s is running
high-availability: no
  datastore master nodes: 127.0.0.1:19001
  datastore standby nodes: none
addons:
  enabled:
    dns                  # (core) CoreDNS
    ha-cluster           # (core) Configure high availability on the current node
    helm                 # (core) Helm - the package manager for Kubernetes
    helm3                # (core) Helm 3 - the package manager for Kubernetes
  disabled:
    cert-manager         # (core) Cloud native certificate management
    cis-hardening        # (core) Apply CIS K8s hardening
    community            # (core) The community addons repository
    dashboard            # (core) The Kubernetes dashboard
```

Then I decided to add an alias to simplify the use of microk8s:

```
root@new-server:~# alias kubectl='sudo microk8s kubectl'
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~#
root@new-server:~# kubectl get nodes
NAME          STATUS   ROLES    AGE   VERSION
new-server    Ready    <none>   2m    v1.31.2
root@new-server:~# kubectl get nodes
NAME          STATUS   ROLES    AGE    VERSION
new-server    Ready    <none>   2m2s   v1.31.2
root@new-server:~#
```

Now we have microk8s running and ready. We try to enable DNS and Storage for it:

root@new-server: ~/virtualization_project/task3

```
root@new-server:~/virtualization_project/task3# sudo microk8s enable dns storage
Infer repository core for addon dns
Infer repository core for addon storage
WARNING: Do not enable or disable multiple addons in one command.
         This form of chained operations on addons will be DEPRECATED in the future.
         Please, enable one addon at a time: 'microk8s enable <addon>'
Addon core/dns is already enabled
DEPRECATION WARNING: 'storage' is deprecated and will soon be removed. Please use 'hostpath-storage' instead.

Infer repository core for addon hostpath-storage
Enabling default storage class.
WARNING: Hostpath storage is not suitable for production environments.
         A hostpath volume can grow beyond the size limit set in the volume claim manifest.

deployment.apps/hostpath-provisioner created
storageclass.storage.k8s.io/microk8s-hostpath created
serviceaccount/microk8s-hostpath created
clusterrole.rbac.authorization.k8s.io/microk8s-hostpath created
clusterrolebinding.rbac.authorization.k8s.io/microk8s-hostpath created
Storage will be available soon.
root@new-server:~/virtualization_project/task3#
```

It's time to write the needed deployments and services for this project. Here, I tried to read the docker-compose file for each service and write a compatible deployment and service yaml file for microk8s. We start by database:

```yaml
mysql:
    container_name: "rfsim5g-mysql"
    image: mysql:5.7
    volumes:
        - ./oai_db.sql:/docker-entrypoint-initdb.d/oai_db.sql
        - ./mysql-healthcheck.sh:/tmp/mysql-healthcheck.sh
    environment:
        - TZ=Europe/Paris
        - MYSQL_DATABASE=oai_db
        - MYSQL_USER=test
        - MYSQL_PASSWORD=test
        - MYSQL_ROOT_PASSWORD=linux
    healthcheck:
        test: /bin/bash -c "/tmp/mysql-healthcheck.sh"
        interval: 10s
        timeout: 5s
        retries: 5
    networks:
        public_net:
            ipv4_address: 192.168.71.131
```

First of all, we have a file for our database. Therefore, we should create a configmap to map the file we have to microk8s. we create a configmap called "db-scripts":

root@new-server: ~/virtualization_project/task3

```
root@new-server:~/virtualization_project/task3# kubectl create configmap db-scripts --from-file=oai_db.sql
configmap/db-scripts created
root@new-server:~/virtualization_project/task3# kubectl get configmaps
NAME               DATA   AGE
db-scripts         1      6s
kube-root-ca.crt   1      35h
root@new-server:~/virtualization_project/task3#
```

Now we write the db-deployment for our mysql database:

root@new-server: ~/virtualization_project/task3

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
        env:
        - name: MYSQL_DATABASE
          value: "oai_db"
        - name: MYSQL_USER
          value: "test"
        - name: MYSQL_PASSWORD
          value: "test"
        - name: MYSQL_ROOT_PASSWORD
          value: "linux"
        volumeMounts:
        - name: db-scripts
          mountPath: /docker-entrypoint-initdb.d/oai_db.sql
          subPath: oai_db.sql
      volumes:
      - name: db-scripts
        configMap:
          name: db-scripts
```

We set env for every deployment based on environment in docker-compose. Also, we add volumes and configmap to our deployment for it to be able to access the database file.

We also write a db-service for our other deployments to access the database:

root@new-server: ~/virt

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
    app: mysql
  ports:
  - protocol: TCP
    port: 3306
    targetPort: 3306
```

Now we apply them to microk8s:

```
root@new-server:~/virtualization_project/task3# vim db-deployment.yaml
root@new-server:~/virtualization_project/task3# vim db-service.yaml
root@new-server:~/virtualization_project/task3#
root@new-server:~/virtualization_project/task3# kubectl apply -f db-deployment.yaml
deployment.apps/mysql created
root@new-server:~/virtualization_project/task3# kubectl apply -f db-service.yaml
service/mysql created
root@new-server:~/virtualization_project/task3# kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
kubernetes   ClusterIP   10.152.183.1     <none>        443/TCP    35h
mysql        ClusterIP   10.152.183.223   <none>        3306/TCP   7s
root@new-server:~/virtualization_project/task3# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
mysql-c78f5d65b-vv6cb    1/1     Running   0          20s
root@new-server:~/virtualization_project/task3#
```

Now we move on to the next deployment which is “oai-nrf”:

```yaml
oai-nrf:
    container_name: "rfsim5g-oai-nrf"
    image: oai-nrf:latest
    environment:
        - NRF_INTERFACE_NAME_FOR_SBI=eth0
        - NRF_INTERFACE_PORT_FOR_SBI=80
        - NRF_INTERFACE_HTTP2_PORT_FOR_SBI=9090
        - NRF_API_VERSION=v1
        - INSTANCE=0
        - PID_DIRECTORY=/var/run
    networks:
        public_net:
            ipv4_address: 192.168.71.130
    volumes:
        - ./nrf-healthcheck.sh:/openair-nrf/bin/nrf-healthcheck.sh
    healthcheck:
        test: /bin/bash -c "/openair-nrf/bin/nrf-healthcheck.sh"
        interval: 10s
        timeout: 5s
        retries: 5
```

We write a deployment for it:

root@new-server: ~/virtualization_project/task3

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oai-nrf
  labels:
    app: oai-nrf
spec:
  replicas: 1
  selector:
    matchLabels:
      app: oai-nrf
  template:
    metadata:
      labels:
        app: oai-nrf
    spec:
      containers:
        - name: oai-nrf
          image: oaisoftwarealliance/oai-nrf:latest
          env:
            - name: NRF_INTERFACE_NAME_FOR_SBI
              value: eth0
            - name: NRF_INTERFACE_PORT_FOR_SBI
              value: "80"
            - name: NRF_INTERFACE_HTTP2_PORT_FOR_SBI
              value: "9090"
            - name: NRF_API_VERSION
              value: v1
            - name: INSTANCE
              value: "0"
            - name: PID_DIRECTORY
              value: /var/run
          ports:
            - containerPort: 8080
              name: http
```

Like before, the env is just a copy of environment from docker-compose. After that, we should use the new image mentioned in the project description and also we have to expose port 8080

for it because this images runs a listener on this port. Also because of that, we have to write a service for it:

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: oai-nrf
  labels:
    app: oai-nrf
spec:
  selector:
    app: oai-nrf
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
      name: http
  type: ClusterIP
```

Now we do the same for oai-amf. The thing in oai-amf is that it needs to connect to mysql and oai-nrf. In docker-compose, because these two services has some static IP, oai-amf needed the IP of these services but in microk8s, they both have a service instead of static IP. Therefore, we have to pass the service name instead of an static IP to oai-amf in its environment variables:

```yaml
    - name: NRF_IPV4_ADDRESS
      value: "oai-nrf"
    - name: NRF_PORT
      value: "80"
    - name: NRF_API_VERSION
      value: "v1"
    - name: NRF_FQDN
      value: "oai-nrf"
    - name: MYSQL_SERVER
      value: "mysql"
```

root@new-server: ~/virtualization_project/task3

```
root@new-server:~/virtualization_project/task3# kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mysql-c78f5d65b-vv6cb       1/1     Running   0          16h
oai-amf-645cfb678b-gqkcs    1/1     Running   0          3m38s
oai-nrf-6db6c99cc4-zgskw    1/1     Running   0          3h53m
root@new-server:~/virtualization_project/task3#
```

Mohammadamin Aryan – Virtualization Project

oai-amf:

```
root@new-server: ~/virtualization_project/task3
[2024-11-27 17:54:50.396] [amf_sbi] [info] Could not get JSON content from the response
[2024-11-27 17:54:50.397] [amf_sbi] [debug] NF Update, response from NRF, JSON data:
null
[2024-11-27 17:54:50.397] [amf_app] [debug] Received SBI_UPDATE_NF_INSTANCE_RESPONSE
[2024-11-27 17:54:50.397] [amf_app] [debug] Handle NF Update response
[2024-11-27 17:54:50.397] [amf_app] [debug] Set a timer to the next Heart-beat (10)
[2024-11-27 17:55:00.398] [amf_app] [debug] Send ITTI msg to SBI task to trigger NRF Heartbeat
[2024-11-27 17:55:00.400] [amf_sbi] [info] Receive Update NF Instance Request, handling ...
[2024-11-27 17:55:00.400] [amf_sbi] [debug] Send NF Update to NRF
[2024-11-27 17:55:00.401] [amf_sbi] [debug] Send NF Update to NRF, Msg body [{"op":"replace","path":"/nfStatus","value":"REGISTERED"}]
[2024-11-27 17:55:00.402] [amf_sbi] [debug] Send NF Update to NRF, NRF URI http://oai-nrf:8080/nnrf-nfm/v1/nf-instances/0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 17:55:00.402] [amf_sbi] [info] Send HTTP message to http://oai-nrf:8080/nnrf-nfm/v1/nf-instances/0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 17:55:00.403] [amf_sbi] [info] HTTP message Body: [{"op":"replace","path":"/nfStatus","value":"REGISTERED"}]
[2024-11-27 17:55:00.403] [amf_sbi] [debug] Send a simple HTTP request
[2024-11-27 17:55:00.422] [amf_sbi] [info] Get response with HTTP code (204)
[2024-11-27 17:55:00.423] [amf_sbi] [info] Could not get JSON content from the response
[2024-11-27 17:55:00.423] [amf_sbi] [debug] NF Update, response from NRF, JSON data:
null
[2024-11-27 17:55:00.423] [amf_app] [debug] Received SBI_UPDATE_NF_INSTANCE_RESPONSE
[2024-11-27 17:55:00.423] [amf_app] [debug] Handle NF Update response
[2024-11-27 17:55:00.423] [amf_app] [debug] Set a timer to the next Heart-beat (10)
[2024-11-27 17:55:09.558] [amf_app] [info]
  |-----------------------------------------------------------------------------------------------------------------------------------|
  |-----------------------------------------------------------------------gNBs' Information-----------------------------------------|
  |  Index |              Status              |              Global Id              |              gNB Name              |                PLMN                |
  |    -   |                -                 |                  -                  |                  -                 |                  -                 |
  |-----------------------------------------------------------------------------------------------------------------------------------|

  |-----------------------------------------------------------------------------------------------------------------------------------|
  |-----------------------------------------------------------------------UEs' Information-------------------------------------------|
  |  Index |     5GMM State     |        IMSI        |        GUTI        |   RAN UE NGAP ID   |   AMF UE NGAP ID   |        PLMN        |       Cell Id      |
  |    -   |                    |         -          |         -          |                    |                    |         -          |                    |
  |-----------------------------------------------------------------------------------------------------------------------------------|

[2024-11-27 17:55:10.424] [amf_app] [debug] Send ITTI msg to SBI task to trigger NRF Heartbeat
[2024-11-27 17:55:10.425] [amf_sbi] [info] Receive Update NF Instance Request, handling ...
[2024-11-27 17:55:10.426] [amf_sbi] [debug] Send NF Update to NRF
[2024-11-27 17:55:10.426] [amf_sbi] [debug] Send NF Update to NRF, Msg body [{"op":"replace","path":"/nfStatus","value":"REGISTERED"}]
[2024-11-27 17:55:10.427] [amf_sbi] [debug] Send NF Update to NRF, NRF URI http://oai-nrf:8080/nnrf-nfm/v1/nf-instances/0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 17:55:10.427] [amf_sbi] [info] Send HTTP message to http://oai-nrf:8080/nnrf-nfm/v1/nf-instances/0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 17:55:10.428] [amf_sbi] [info] HTTP message Body: [{"op":"replace","path":"/nfStatus","value":"REGISTERED"}]
[2024-11-27 17:55:10.428] [amf_sbi] [debug] Send a simple HTTP request
[2024-11-27 17:55:10.444] [amf_sbi] [info] Get response with HTTP code (204)
[2024-11-27 17:55:10.445] [amf_sbi] [info] Could not get JSON content from the response
[2024-11-27 17:55:10.445] [amf_sbi] [debug] NF Update, response from NRF, JSON data:
null
[2024-11-27 17:55:10.446] [amf_app] [debug] Received SBI_UPDATE_NF_INSTANCE_RESPONSE
[2024-11-27 17:55:10.446] [amf_app] [debug] Handle NF Update response
[2024-11-27 17:55:10.446] [amf_app] [debug] Set a timer to the next Heart-beat (10)
root@new-server:~/virtualization_project/task3#
```

oai-nrf:

```
root@new-server: ~/virtualization_project/task3
[2024-11-27 16:54:40.381] [nrf_sbi] [info] Got a request to update an NF instance, Instance ID: 0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 16:54:40.382] [nrf_app] [info] Handle Update NF Instance request, NF type AMF (HTTP version 2)
[2024-11-27 16:54:40.383] [nrf_app] [info] NF Heart-Beat procedure!
[2024-11-27 16:54:40.383] [nrf_app] [debug] Replace member nfStatus with new value REGISTERED
[2024-11-27 16:54:40.383] [nrf_app] [info] Updated the NF profile (profile ID 0bcee430-2277-4c8b-92ef-69423da0368d)
[2024-11-27 16:54:40.384] [nrf_app] [debug] NF update for Heartbeat, current time 1732726480384
[2024-11-27 16:54:40.384] [nrf_app] [debug] Set NF status to REGISTERED
[2024-11-27 16:54:50.394] [nrf_sbi] [info] Got a request to update an NF instance, Instance ID: 0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 16:54:50.394] [nrf_app] [info] Handle Update NF Instance request, NF type AMF (HTTP version 2)
[2024-11-27 16:54:50.395] [nrf_app] [info] NF Heart-Beat procedure!
[2024-11-27 16:54:50.395] [nrf_app] [debug] Replace member nfStatus with new value REGISTERED
[2024-11-27 16:54:50.395] [nrf_app] [info] Updated the NF profile (profile ID 0bcee430-2277-4c8b-92ef-69423da0368d)
[2024-11-27 16:54:50.395] [nrf_app] [debug] NF update for Heartbeat, current time 1732726490395
[2024-11-27 16:54:50.395] [nrf_app] [debug] Set NF status to REGISTERED
[2024-11-27 16:55:00.419] [nrf_sbi] [info] Got a request to update an NF instance, Instance ID: 0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 16:55:00.419] [nrf_app] [info] Handle Update NF Instance request, NF type AMF (HTTP version 2)
[2024-11-27 16:55:00.420] [nrf_app] [info] NF Heart-Beat procedure!
[2024-11-27 16:55:00.420] [nrf_app] [debug] Replace member nfStatus with new value REGISTERED
[2024-11-27 16:55:00.420] [nrf_app] [info] Updated the NF profile (profile ID 0bcee430-2277-4c8b-92ef-69423da0368d)
[2024-11-27 16:55:00.420] [nrf_app] [debug] NF update for Heartbeat, current time 1732726500420
[2024-11-27 16:55:00.420] [nrf_app] [debug] Set NF status to REGISTERED
[2024-11-27 16:55:10.440] [nrf_sbi] [info] Got a request to update an NF instance, Instance ID: 0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 16:55:10.441] [nrf_app] [info] Handle Update NF Instance request, NF type AMF (HTTP version 2)
[2024-11-27 16:55:10.441] [nrf_app] [info] NF Heart-Beat procedure!
[2024-11-27 16:55:10.441] [nrf_app] [debug] Replace member nfStatus with new value REGISTERED
[2024-11-27 16:55:10.441] [nrf_app] [info] Updated the NF profile (profile ID 0bcee430-2277-4c8b-92ef-69423da0368d)
[2024-11-27 16:55:10.442] [nrf_app] [debug] NF update for Heartbeat, current time 1732726510442
[2024-11-27 16:55:10.442] [nrf_app] [debug] Set NF status to REGISTERED
[2024-11-27 16:55:20.466] [nrf_sbi] [info] Got a request to update an NF instance, Instance ID: 0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 16:55:20.466] [nrf_app] [info] Handle Update NF Instance request, NF type AMF (HTTP version 2)
[2024-11-27 16:55:20.467] [nrf_app] [info] NF Heart-Beat procedure!
[2024-11-27 16:55:20.467] [nrf_app] [debug] Replace member nfStatus with new value REGISTERED
[2024-11-27 16:55:20.467] [nrf_app] [info] Updated the NF profile (profile ID 0bcee430-2277-4c8b-92ef-69423da0368d)
[2024-11-27 16:55:20.467] [nrf_app] [debug] NF update for Heartbeat, current time 1732726520467
[2024-11-27 16:55:20.467] [nrf_app] [debug] Set NF status to REGISTERED
[2024-11-27 16:55:30.492] [nrf_sbi] [info] Got a request to update an NF instance, Instance ID: 0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 16:55:30.498] [nrf_app] [info] Handle Update NF Instance request, NF type AMF (HTTP version 2)
[2024-11-27 16:55:30.498] [nrf_app] [info] NF Heart-Beat procedure!
[2024-11-27 16:55:30.498] [nrf_app] [debug] Replace member nfStatus with new value REGISTERED
[2024-11-27 16:55:30.498] [nrf_app] [info] Updated the NF profile (profile ID 0bcee430-2277-4c8b-92ef-69423da0368d)
[2024-11-27 16:55:30.498] [nrf_app] [debug] NF update for Heartbeat, current time 1732726530498
[2024-11-27 16:55:30.498] [nrf_app] [debug] Set NF status to REGISTERED
[2024-11-27 16:55:40.524] [nrf_sbi] [info] Got a request to update an NF instance, Instance ID: 0bcee430-2277-4c8b-92ef-69423da0368d
[2024-11-27 16:55:40.525] [nrf_app] [info] Handle Update NF Instance request, NF type AMF (HTTP version 2)
[2024-11-27 16:55:40.525] [nrf_app] [info] NF Heart-Beat procedure!
[2024-11-27 16:55:40.525] [nrf_app] [debug] Replace member nfStatus with new value REGISTERED
[2024-11-27 16:55:40.526] [nrf_app] [info] Updated the NF profile (profile ID 0bcee430-2277-4c8b-92ef-69423da0368d)
[2024-11-27 16:55:40.526] [nrf_app] [debug] NF update for Heartbeat, current time 1732726540526
[2024-11-27 16:55:40.526] [nrf_app] [debug] Set NF status to REGISTERED
root@new-server:~/virtualization_project/task3#
```

Next we try to apply oai-smf to microk8s. First of all, because there is a conf file for this service, we have to add it to microk8s using configmap:

```
root@new-server: ~/virtualization_project/task3
root@new-server:~/virtualization_project/task3# kubectl create configmap oai-smf-config --from-file=oai-smf.conf
configmap/oai-smf-config created
root@new-server:~/virtualization_project/task3#
```

Now we write a deployment and a service for oai-smf. Like oai-nrf, we write all the envs for it and replace the static IPs with service names of oai-nrf and oai-spgwu (not deployed yet) and we expose the port 80 for this service. Also we write a volume part for it to read from the conf file:

```
      volumeMounts:
        - name: smf-config-volume
          mountPath: /openair-smf/bin/oai-smf.conf
          subPath: oai-smf.conf
  volumes:
    - name: smf-config-volume
      configMap:
        name: oai-smf-config
```

```
root@new-server: ~/virtualization_project/task3
root@new-server:~/virtualization_project/task3# kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
mysql-c78f5d65b-vv6cb       1/1     Running   0          18h
oai-amf-645cfb678b-gqkcs    1/1     Running   0          106m
oai-nrf-6db6c99cc4-zgskw    1/1     Running   0          5h35m
oai-smf-d8946bf58-wwjxv     1/1     Running   0          86m
root@new-server:~/virtualization_project/task3# kubectl get services
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)           AGE
kubernetes   ClusterIP   10.152.183.1     <none>        443/TCP           2d6h
mysql        ClusterIP   10.152.183.223   <none>        3306/TCP          18h
oai-nrf      ClusterIP   10.152.183.117   <none>        8080/TCP          18h
oai-smf      ClusterIP   10.152.183.176   <none>        80/TCP,9090/TCP   88m
root@new-server:~/virtualization_project/task3#
```

Now we try to deploy oai-spgwu. Like before, we write a deployment and a service to expose port 80 and we write all their environment variables and replace IPs with suited service names. You can see the deployment and service in the github project.