

## 1. Problem 1:

- Create bridge network with subnet 192.168.0.0/24.
- Run 2 containers and attach containers to this network.
- Create another bridge network with subnet 10.5.0.0/24.
- Run any container and attach it to the new network.
- Make sure that the containers at different network can't ping each other

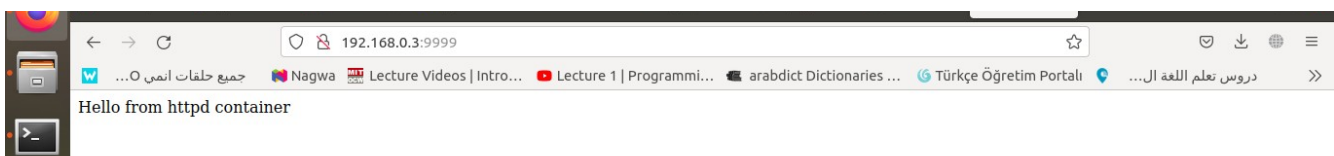
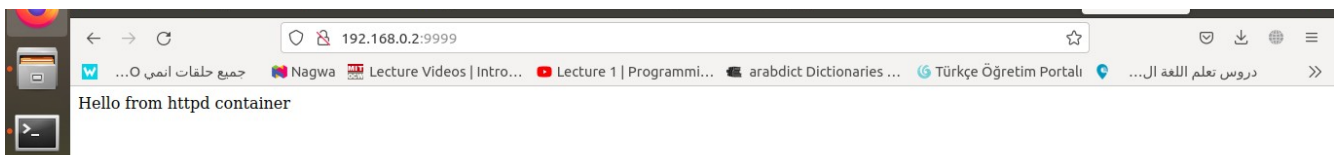
```
$ sudo docker network create --driver=bridge --subnet=192.168.0.0/24 mynet192
```

```
$ sudo docker run --name container1-mynet192 httpd:v1.0
```

```
$ sudo docker network connect mynet192 container1-mynet192
```

```
$ sudo docker run --name container2-mynet192 httpd:v1.0
```

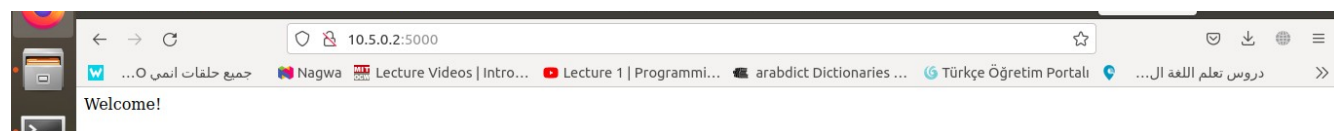
```
$ sudo docker network connect mynet192 container2-mynet192
```



```
$ sudo docker network create --driver=bridge --subnet=10.5.0.0/24 mynet10
```

```
$ sudo docker run --name container1-mynet10 flask:v1.0
```

```
$ sudo docker network connect mynet10 container1-mynet10
```



## 2. Problem 2:

- Create a docker compose to up mysql container, node express app depend on mysqldb.
- Add volume for mysqldb
- Read enviroment variables from .env file

#=====Dockerfile=====

FROM node:16

WORKDIR myapp

COPY package.json .

RUN npm install

RUN apt update && apt install -y mysql\\*

COPY . .

EXPOSE 3000

CMD [ "node", "app.js" ]

#=====docker-compose.yml=====

```
docker-compose.yml x
home > dina > Documents > cloud computing > Sprints - DevOps >
1  version: '3.8'
2
3  services:
4    app:
5      build: .
6      ports:
7        - 3000:3000
8      depends_on:
9        - mysqldb
10   mysqldb:
11     image: mysql
12     restart: always
13     environment:
14       DB_USERNAME: ${DB_USERNAME}
15       DB_PASSWORD: ${DB_PASSWORD}
16       DB_NAME: ${DB_NAME}
17     env_file: ".env"
18     volumes:
19       - db-data:/var/lib/mysql
20   volumes:
21     db-data:
22
```



### 3. Problem 3:

- Create static html file
- Write Dockerfile to build image based on httpd to host the html file and specify the following
- Copy the html file.
- Copy a new configuration file to listen on port 9999 instead of 80
- Open the port 9999 in the container
- Add environment variable CONTAINER with value docker .
- Add startup command to echo the variable