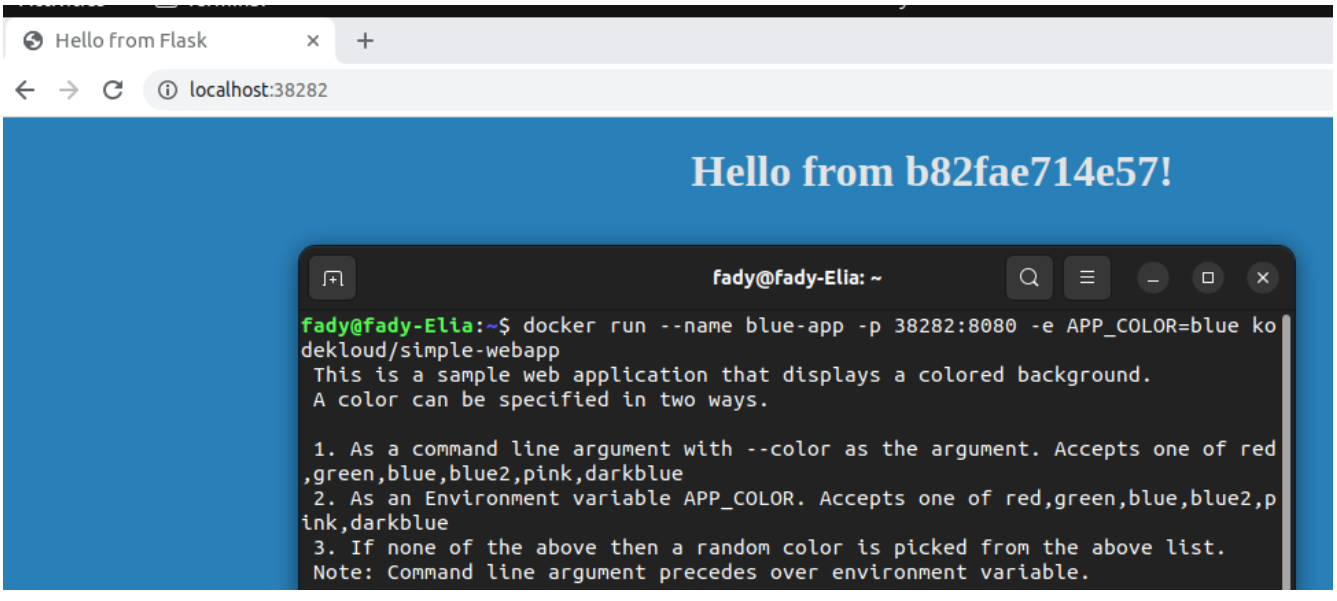
1- Run an instance of nginx:alpine with a name nginx and map port 8080 on the container to 38282 on the host.

```
fady@fady-Elia:~$ docker run -d --name nginx  -p 38282:8080 nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
f56be85fc22e: Already exists
97c80f11709c: Already exists
afb503c1f124: Already exists
f8c948b732dd: Already exists
d021bba29710: Already exists
cadcca1af197: Already exists
4aacde79cec4: Already exists
Digest: sha256:2e776a66a3556f001aba13431b26e448fe8acba277bf93d2ab1a785571a46d90
Status: Downloaded newer image for nginx:alpine
docker: Error response from daemon: Conflict. The container name "/nginx" is al
ready in use by container "3c5e5caa16a065721174d2201ab4300f88118276f0022afe6cf1
a6c2ec60f002". You have to remove (or rename) that container to be able to reus
e that name.
See 'docker run --help'.
fady@fady-Elia:~$
```

2- create ubuntu image and check the size of it

```
fady@fady-Elia:~$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
dbf6a9befcde: Already exists
Digest: sha256:dfd64a3b4296d8c9b62aa3309984f8620b98d87e47492599ee20739e8eb54fbf
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
fady@fady-Elia:~$ docker images ubuntu
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    3b418d7b466a   4 weeks ago   77.8MB
fady@fady-Elia:~$
```

3- Run a container named blue-app using image kodekloud/simplewebapp and set the environment variable APP_COLOR to blue. Make the application available on port 38282 on the host. The application listens on port 8080.



4- Deploy a mysql database using the mysql image and name it mysqldb Set the database password to use db_pass123 then inspect it to check the value

```
fady@fady-Elia:~$ docker run --name mysqldb -e MYSQL_ROOT_PASSWORD=db_pass123 -
d mysql
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
90e2fb2facff: Pull complete
ba60eb20fd5f: Pull complete
4f509402d469: Pull complete
496c2cfa6815: Pull complete
8ec1dfa9522c: Pull complete
6dec7ba896f8: Pull complete
dc9ff75362b0: Pull complete
73e4682f9014: Pull complete
9ffdeecd6fb6: Pull complete
a4346ccfb53f: Pull complete
434c13bc32de: Pull complete
Digest: sha256:d6164ff4855b9b3f2c7748c6ec564ccff841f79a7023db0f9293143481a44b6e
Status: Downloaded newer image for mysql:latest
7da52248f2b1918f3723b671dae2ca48c85811ec8d940f6821ad1ce1782ba409
fady@fady-Elia:~$ docker inspect --format='{{index .Config.Env 0}}' mysqldb
MYSQL_ROOT_PASSWORD=db_pass123
fady@fady-Elia:~$
```

5- pull the code from https://github.com/sabreensalama/dockerizenode-app-task and create a docker file for this flask app

```
fady@fady-Elia:~$ git clone git@github.com:sabreensalama/dockerize-node-app-tas
k.git
Cloning into 'dockerize-node-app-task'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 12 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (1/1), done.
fady@fady-Elia:~$
```

Dockerfile > ...

```
1   # Use the official Python base image with the desired version
2   FROM python:3.9
3
4   # Set the working directory inside the container
5   WORKDIR /app
6
7   # Copy the requirements.txt file to the container
8   COPY requirements.txt .
9
10  # Install the required Python packages
11  RUN pip install --no-cache-dir -r requirements.txt
12
13  # Copy the entire project code to the container
14  COPY . .
15
16  # Set the command to run the Flask app
17  CMD ["python", "app.py"]
```

```
fady@fady-Elia:~/simple-flask-app$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
flask-app                 latest    1a0a47ba61ac   23 minutes ago   917MB
nginx                     alpine    fe7edaf8a8dc   3 days ago       41.4MB
mysql                     latest    05db07cd74c0   3 days ago       565MB
python                    3.9       3a6891e6dad7   5 days ago       906MB
ubuntu                    latest    3b418d7b466a   4 weeks ago      77.8MB
kodekloud/simple-webapp   latest    c6e3cd9aae36   4 years ago      84.8MB
fady@fady-Elia:~/simple-flask-app$ docker build -t flask-app .
Sending build context to Docker daemon  65.02kB
Step 1/7 : FROM python:3.8-slim-buster
3.8-slim-buster: Pulling from library/python
99bf4787315b: Pull complete

a8a848364b53: Pull complete

ca9f63e352d8: Pull complete

b7c88b22ab23: Pull complete

d5dd36a4520b: Pull complete

Digest: sha256:eb48d017c5e117d9fbcbe991b4dbc61339734e01578d8d350b38fe2033a67
```

6- Create a volume called mysql_data, Run a mysql container again, but this time map a volume to the container so that the data stored by the container is stored at /opt/data on the host. Use the same name : mysql-db and same password: db_pass123 as before. Mysql stores data at /var/lib/mysql inside the container.

```
fady@fady-Elia:~$ docker volume create mysql_data
mysql_data
fady@fady-Elia:~$ docker run --name mysql-db -e MYSQL_ROOT_PASSWORD=db_pass123
-v mysql_data:/var/lib/mysql -d mysql
8edde22ec4bfd3f97bb65fdf2d41f18b40103e4a5b970e8192bf8a22e0328d19
fady@fady-Elia:~$ docker inspect mysql-db
[
    {
        "Id": "8edde22ec4bfd3f97bb65fdf2d41f18b40103e4a5b970e8192bf8a22e0328d19
",
        "Created": "2023-05-28T10:25:45.537627604Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "mysqld"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 44121,
            "ExitCode": 0,
```