Task01-Docker

OverView:-

1. Write a Dockerfile for any simple backend app
2. Build it
3. Create a custom bridge network
4. Create a container instance from your docker image and it should use the new bridge network

Steps:-

Step 1: Prepare the working space

1.1. Create a Directory called Zerosploit

```
mkdir ~/zerosploit
```

1.2. Create the python file

```
vim app.py
```

1.3. Write a simple backend python program

```
import time
```

#This chunk of code will take the first and last name of the user and print it

```
first_name = input("Enter your first name: ")

last_name = input("Enter your last name: ")

print(f"Hello, {first_name} {last_name}!")
```

#This chunk of code will keep the container running because it will stop if no process is running

```
print("Sleeping for 10 seconds...")

time.sleep(10)
```

Step 2: write the Dockerfile

2.1. Create the Dockerfile

```
vim Dockerfile
```

2.2. Write the following inside the Dockerfile

#using this lightweight base image is better so the container size won't be big

```
FROM python:3.10-slim
```

#Moving to a directory inside the container during run time

```
WORKDIR /app
```

#Copy the python program from the host to the working directory inside the container (/app)

```
COPY app.py .
```

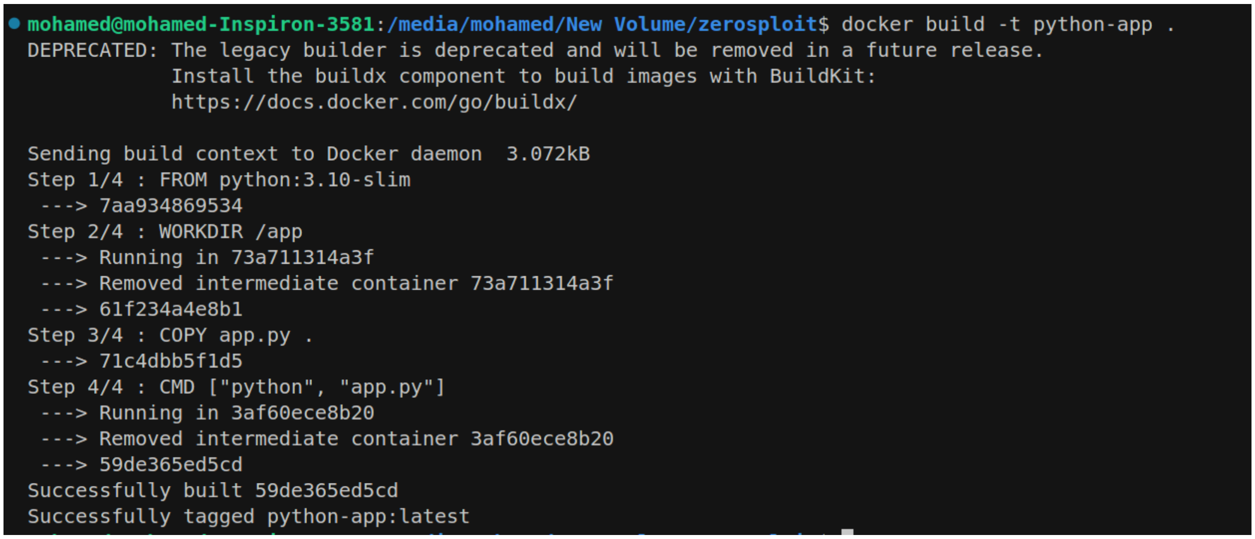
#Executing the following command after the build is done (~$python app.py)

```
CMD ["python", "app.py"]
```

Step 3: Build the new Image

3.1. Build the new image from the Dockerfile

```
docker build -t python-app .
```

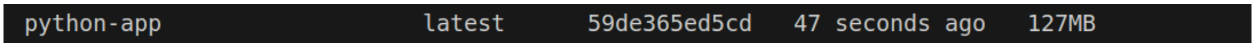


```
mohamed@mohamed-Inspiron-3581:/media/mohamed/New Volume/zerosploit$ docker build -t python-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.072kB
Step 1/4 : FROM python:3.10-slim
 ---> 7aa934869534
Step 2/4 : WORKDIR /app
 ---> Running in 73a711314a3f
 ---> Removed intermediate container 73a711314a3f
 ---> 61f234a4e8b1
Step 3/4 : COPY app.py .
 ---> 71c4dbb5f1d5
Step 4/4 : CMD ["python", "app.py"]
 ---> Running in 3af60ece8b20
 ---> Removed intermediate container 3af60ece8b20
 ---> 59de365ed5cd
Successfully built 59de365ed5cd
Successfully tagged python-app:latest
```

3.1. Check for the newly created image

```
docker images
```



```
python-app                     latest       59de365ed5cd   47 seconds ago   127MB
```

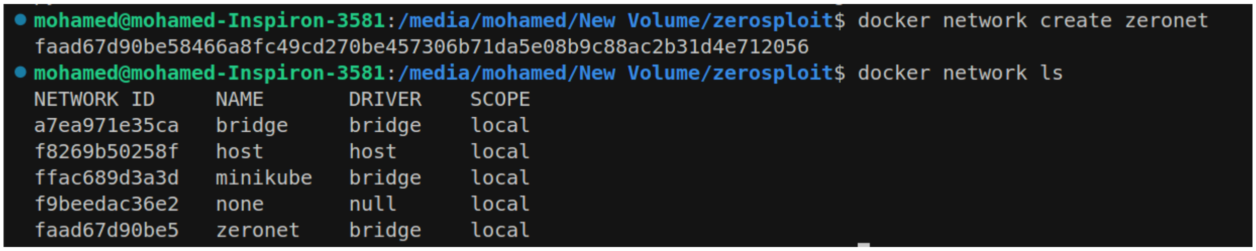
Step 4: Create a new Network Called zeronet

4.1. Create the new network

```
docker network create zeronet
```

4.2. Show available networks

```
docker network ls
```



```
● mohamed@mohamed-Inspiron-3581:/media/mohamed/New Volume/zerosploit$ docker network create zeronet
  faad67d90be58466a8fc49cd270be457306b71da5e08b9c88ac2b31d4e712056
● mohamed@mohamed-Inspiron-3581:/media/mohamed/New Volume/zerosploit$ docker network ls
  NETWORK ID     NAME       DRIVER    SCOPE
  a7ea971e35ca   bridge     bridge    local
  f8269b50258f   host       host      local
  ffac689d3a3d   minikube   bridge    local
  f9beedac36e2   none       null      local
  faad67d90be5   zeronet    bridge    local
```

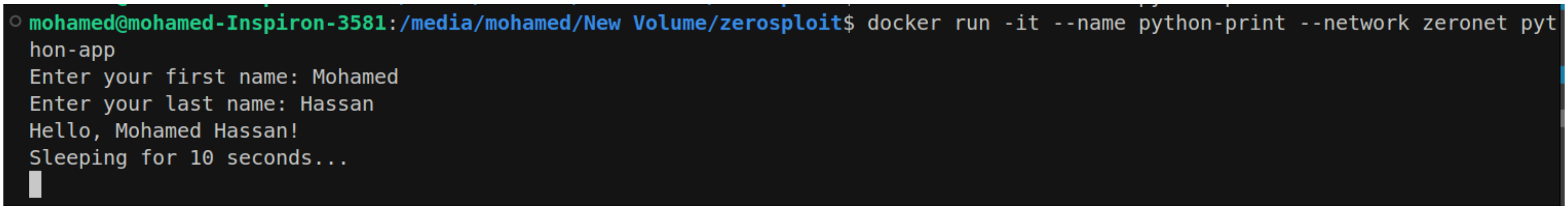
- The default created network type will be bridged network
- All containers created in this network are in the same subnet (LAN)
- All containers inside this network can communicate with each other only by name, as the bridge network also offers (DNS) service

Step 5: Run an instance from the “python-app” image

5.1. Run the new instance (container)

```
docker run -it --name python-print --network zeronet \
python-app
```

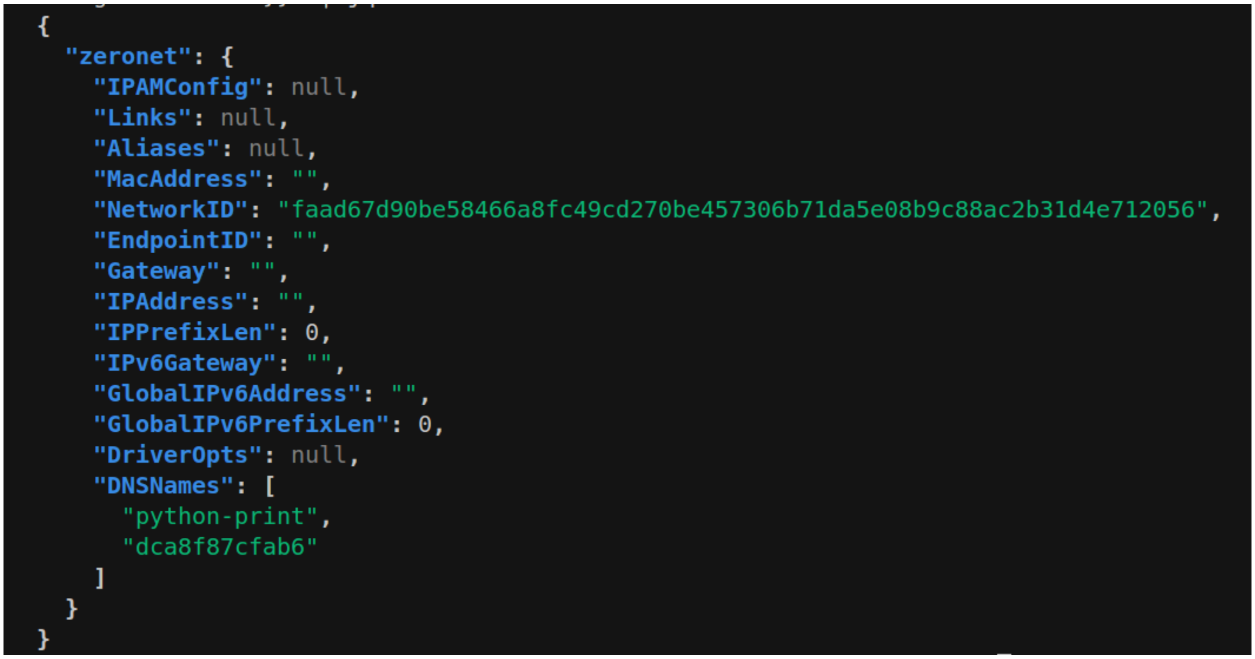
- (-it) used for interactive terminal, as the python program requires an input.
- (- -network) used to choose the network that the container will be attached to



```
mohamed@mohamed-Inspiron-3581:/media/mohamed/New Volume/zerosploit$ docker run -it --name python-print --network zeronet pyt
hon-app
Enter your first name: Mohamed
Enter your last name: Hassan
Hello, Mohamed Hassan!
Sleeping for 10 seconds...
```

5.2. Check for the network settings for the container

```
docker inspect python-print \
 --format '{{ json .NetworkSettings.Networks }}' | jq
```



```
{
  "zeronet": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": null,
    "MacAddress": "",
    "NetworkID": "faad67d90be58466a8fc49cd270be457306b71da5e08b9c88ac2b31d4e712056",
    "EndpointID": "",
    "Gateway": "",
    "IPAddress": "",
    "IPPrefixLen": 0,
    "IPv6Gateway": "",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "DriverOpts": null,
    "DNSNames": [
      "python-print",
      "dca8f87cfab6"
    ]
  }
}
```