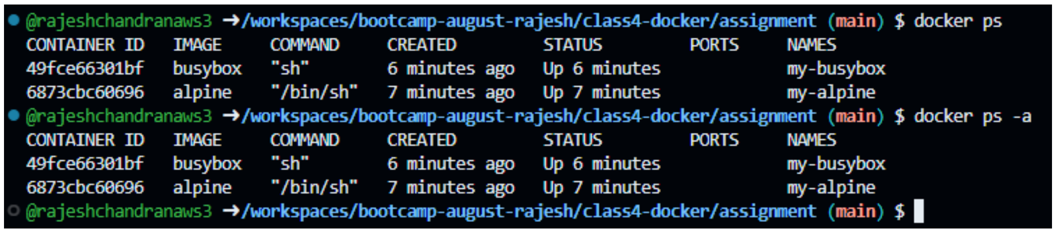
Docker Assignment

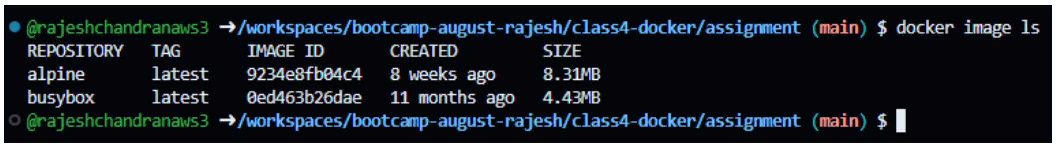
1. Docker Basics

1.1. Running your first containers

docker run -d -t --name my-alpine alpine
docker run -d -t --name my-busybox busybox
docker ps
docker ps -a
docker image ls



```
● @rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
  CONTAINER ID   IMAGE     COMMAND     CREATED         STATUS         PORTS     NAMES
  49fce66301bf   busybox   "sh"        6 minutes ago   Up 6 minutes             my-busybox
  6873cbc60696   alpine    "/bin/sh"   7 minutes ago   Up 7 minutes             my-alpine
● @rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps -a
  CONTAINER ID   IMAGE     COMMAND     CREATED         STATUS         PORTS     NAMES
  49fce66301bf   busybox   "sh"        6 minutes ago   Up 6 minutes             my-busybox
  6873cbc60696   alpine    "/bin/sh"   7 minutes ago   Up 7 minutes             my-alpine
○ @rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ 
```

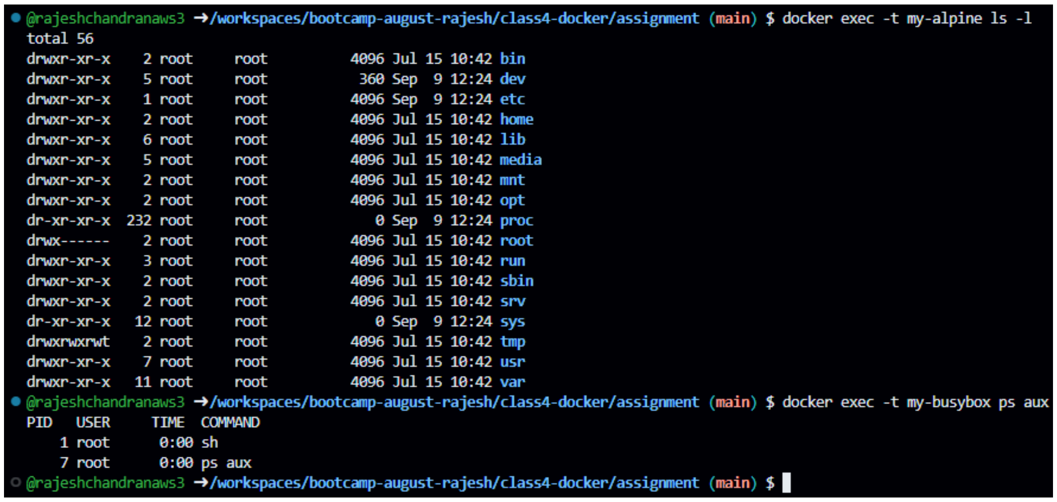


```
● @rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker image ls
  REPOSITORY   TAG       IMAGE ID       CREATED         SIZE
  alpine       latest    9234e8fb04c4   8 weeks ago     8.31MB
  busybox      latest    0ed463b26dae   11 months ago   4.43MB
○ @rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ 
```

Q: What's the difference between docker ps and docker ps -a?
docker ps shows only running containers.
docker ps -a shows all running as well as exited containers.
Q: Why are Alpine and BusyBox images so small?
They contain only the required binaries and libraries, without extra tools or packages.

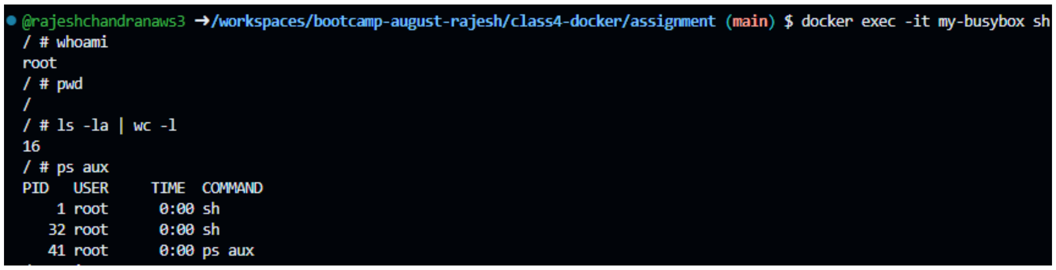
1.2. Container Interaction

docker exec -t my-alpine ls /
docker exec -t my-busybox ps aux

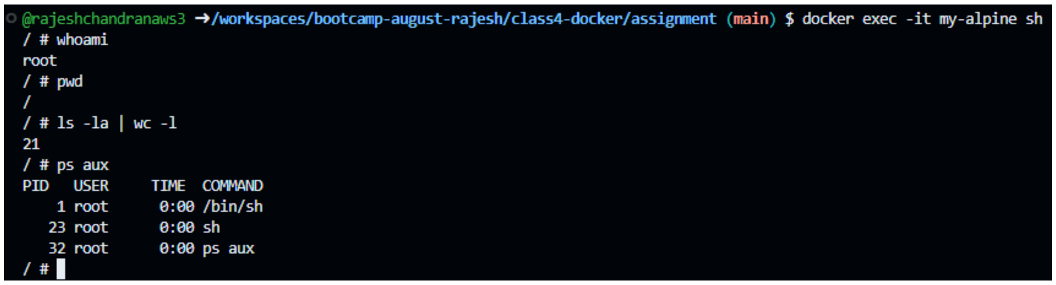


```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -t my-alpine ls -l
total 56
drwxr-xr-x    2 root     root          4096 Jul 15 10:42 bin
drwxr-xr-x    5 root     root           360 Sep  9 12:24 dev
drwxr-xr-x    1 root     root          4096 Sep  9 12:24 etc
drwxr-xr-x    2 root     root          4096 Jul 15 10:42 home
drwxr-xr-x    6 root     root          4096 Jul 15 10:42 lib
drwxr-xr-x    5 root     root          4096 Jul 15 10:42 media
drwxr-xr-x    2 root     root          4096 Jul 15 10:42 mnt
drwxr-xr-x    2 root     root          4096 Jul 15 10:42 opt
dr-xr-xr-x  232 root     root             0 Sep  9 12:24 proc
drwx------    2 root     root          4096 Jul 15 10:42 root
drwxr-xr-x    3 root     root          4096 Jul 15 10:42 run
drwxr-xr-x    2 root     root          4096 Jul 15 10:42 sbin
drwxr-xr-x    2 root     root          4096 Jul 15 10:42 srv
dr-xr-xr-x   12 root     root             0 Sep  9 12:24 sys
drwxrwxrwt    2 root     root          4096 Jul 15 10:42 tmp
drwxr-xr-x    7 root     root          4096 Jul 15 10:42 usr
drwxr-xr-x   11 root     root          4096 Jul 15 10:42 var
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -t my-busybox ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 sh
    7 root      0:00 ps aux
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

docker exec -it my-alpine sh
# Inside container: run `whoami`, `pwd`, `ls -la`
# Type `exit` to leave

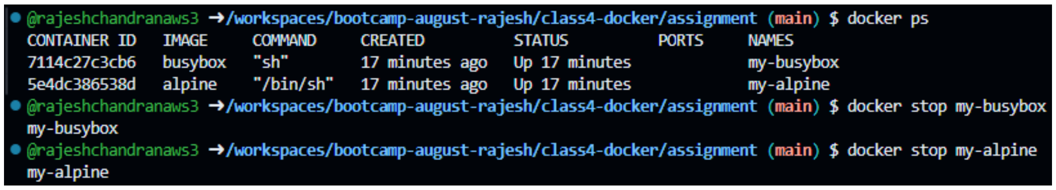


```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -it my-busybox sh
/ # whoami
root
/ # pwd
/
/ # ls -la | wc -l
16
/ # ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 sh
   32 root      0:00 sh
   41 root      0:00 ps aux
```

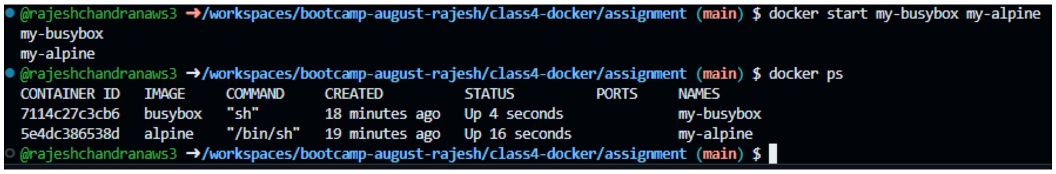


```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -it my-alpine sh
/ # whoami
root
/ # pwd
/
/ # ls -la | wc -l
21
/ # ps aux
PID   USER     TIME  COMMAND
    1 root      0:00 /bin/sh
   23 root      0:00 sh
   32 root      0:00 ps aux
/ #
```

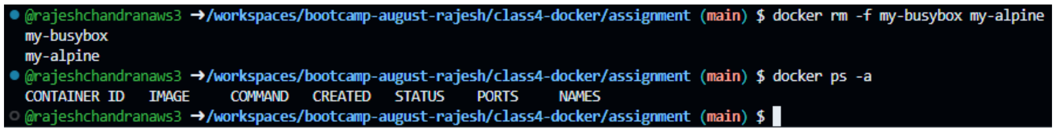
docker stop my-alpine
docker start my-alpine
docker rm -f my-busybox



```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED          STATUS          PORTS     NAMES
7114c27c3cb6   busybox   "sh"        17 minutes ago   Up 17 minutes             my-busybox
5e4dc386538d   alpine    "/bin/sh"   17 minutes ago   Up 17 minutes             my-alpine
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker stop my-busybox
my-busybox
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker stop my-alpine
my-alpine
```



```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker start my-busybox my-alpine
my-busybox
my-alpine
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE     COMMAND     CREATED          STATUS          PORTS     NAMES
7114c27c3cb6   busybox   "sh"        18 minutes ago   Up 4 seconds              my-busybox
5e4dc386538d   alpine    "/bin/sh"   19 minutes ago   Up 16 seconds             my-alpine
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

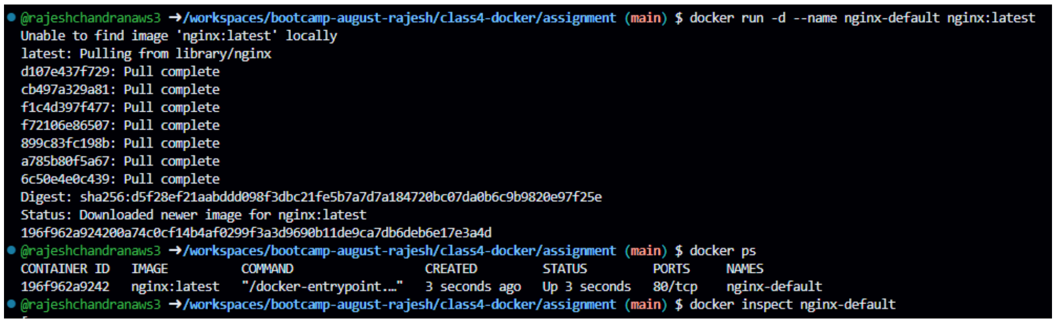


```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker rm -f my-busybox my-alpine
my-busybox
my-alpine
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

2. Docker Networking

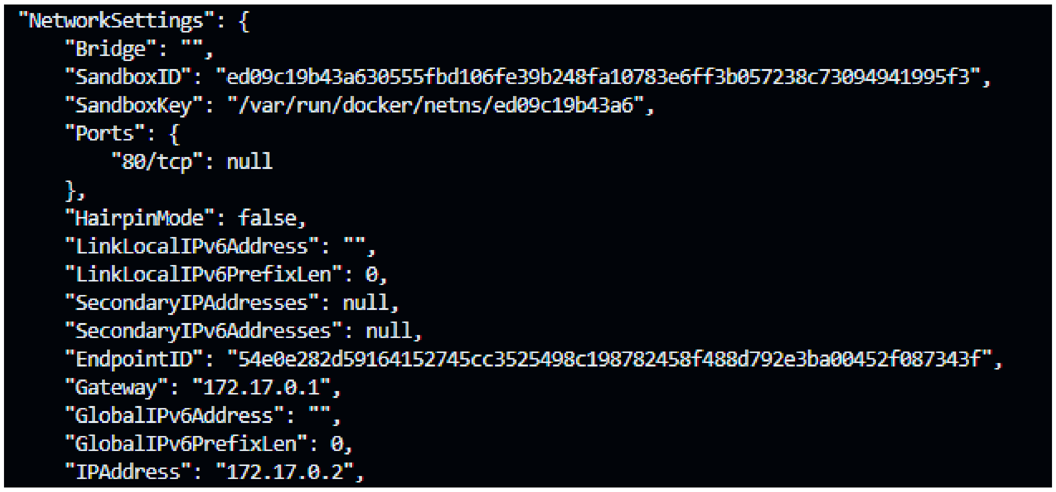
2.1. Default Bridge Network

docker run -d --name nginx-default nginx:latest



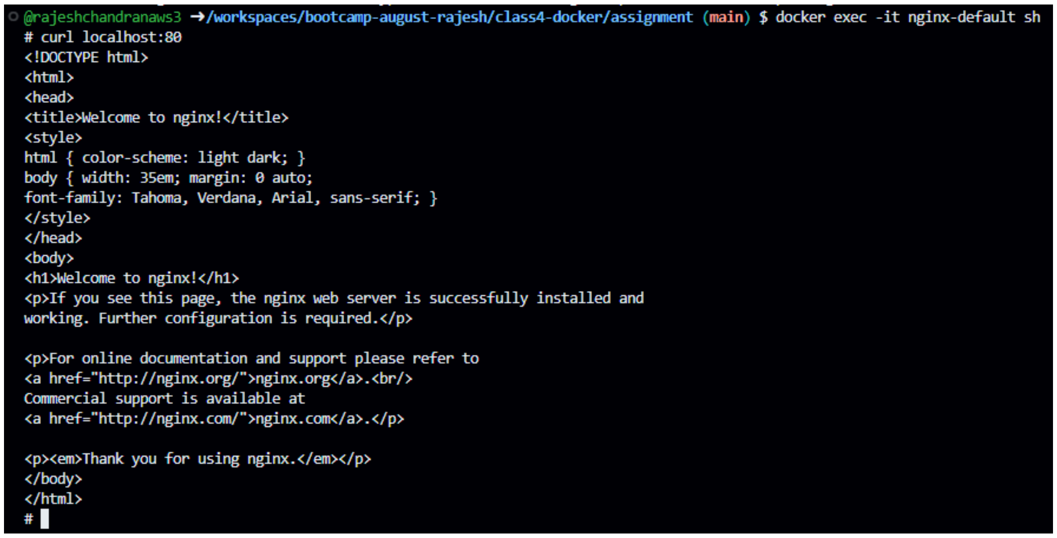
```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker run -d --name nginx-default nginx:latest
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
d107e437f729: Pull complete
cb497a329a81: Pull complete
f1c4d397f477: Pull complete
f72106e86507: Pull complete
899c83fc198b: Pull complete
a785b80f5a67: Pull complete
6c50e4e0c439: Pull complete
Digest: sha256:d5f28ef21aabddd098f3dbc21fe5b7a7d7a184720bc07da0b6c9b9820e97f25e
Status: Downloaded newer image for nginx:latest
196f962a924200a74c0cf14b4af0299f3a3d9690b11de9ca7db6deb6e17e3a4d
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS     NAMES
196f962a9242   nginx:latest   "/docker-entrypoint.…"   3 seconds ago   Up 3 seconds   80/tcp    nginx-default
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker inspect nginx-default
```

docker inspect nginx-default
# Look for NetworkSettings section



```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "ed09c19b43a630555fbd106fe39b248fa10783e6ff3b057238c73094941995f3",
    "SandboxKey": "/var/run/docker/netns/ed09c19b43a6",
    "Ports": {
        "80/tcp": null
    },
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "54e0e282d59164152745cc3525498c198782458f488d792e3ba00452f087343f",
    "Gateway": "172.17.0.1",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "IPAddress": "172.17.0.2",
```

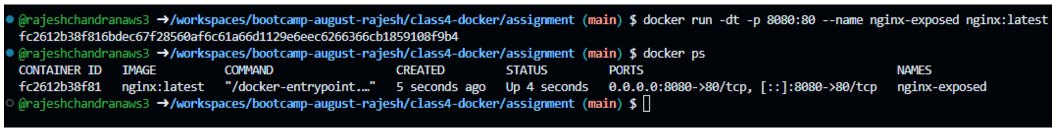
docker exec -it nginx-default curl localhost:80
# Try accessing from host (this should fail):
curl localhost:80



```
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -it nginx-default sh
# curl localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
#
```

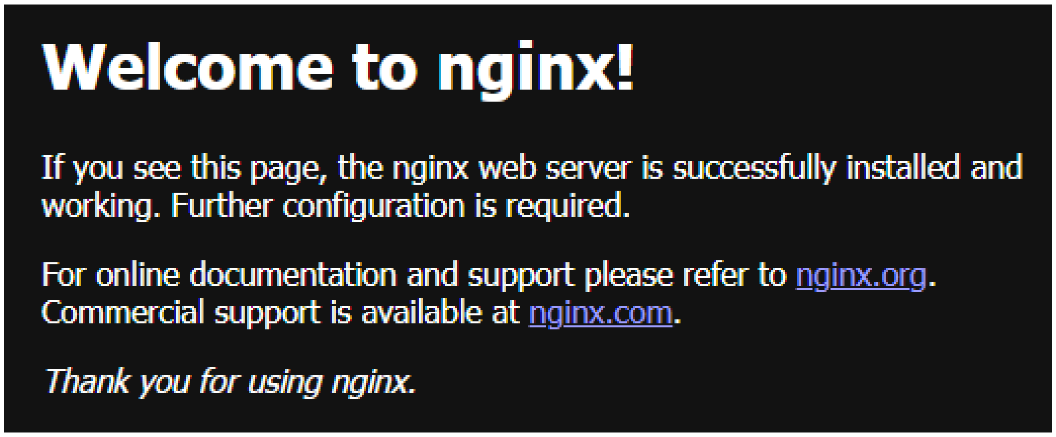
2.2. Port Forwarding

docker run -d -p 8080:80 --name nginx-exposed nginx:latest



```
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker run -dt -p 8080:80 --name nginx-exposed nginx:latest
fc2612b38f816bdec67f28560af6c61a66d1129e6eec6266366cb1859108f9b4
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                                   NAMES
fc2612b38f81   nginx:latest   "/docker-entrypoint.…"   5 seconds ago   Up 4 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   nginx-exposed
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

https://glowing-zebra-5gpprpg745w7f7gg6-8080.app.github.dev/



Welcome to nginx!

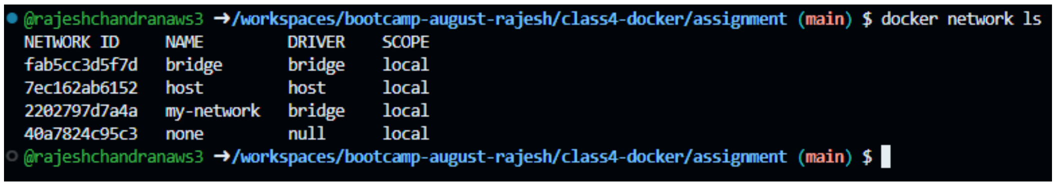
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

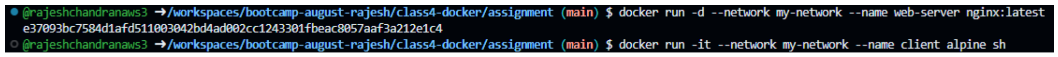
2.3. Custom Bridge Network

docker network create my-network
docker network ls



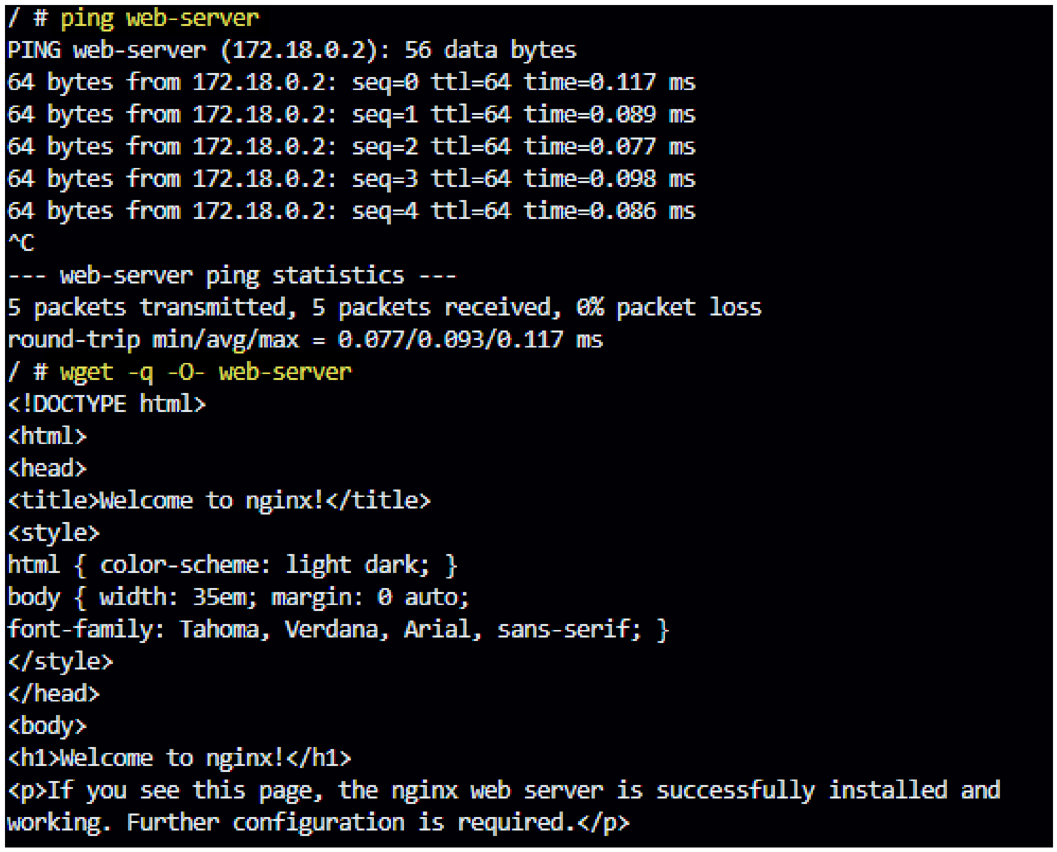
```
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker network ls
NETWORK ID     NAME         DRIVER    SCOPE
fab5cc3d5f7d   bridge       bridge    local
7ec162ab6152   host         host      local
2202797d7a4a   my-network   bridge    local
40a7824c95c3   none         null      local
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

docker run -d --network my-network --name web-server nginx:latest
docker run -it --network my-network --name client alpine sh



```
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker run -d --network my-network --name web-server nginx:latest
e37093bc7584d1afd511003042bd4ad002cc1243301fbeac8057aaf3a212e1c4
@rajeshchandranaws3 →/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker run -it --network my-network --name client alpine sh
```

# Inside the Alpine container:
ping web-server
wget -qO- http://web-server



```
/ # ping web-server
PING web-server (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.117 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.089 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.077 ms
64 bytes from 172.18.0.2: seq=3 ttl=64 time=0.098 ms
64 bytes from 172.18.0.2: seq=4 ttl=64 time=0.086 ms
^C
--- web-server ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.077/0.093/0.117 ms
/ # wget -q -O- web-server
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```

Q: Why can containers ping each other by name in custom networks but not in the default bridge?

Default bridge → No name resolution (must use container IP).
Custom bridge → Built-in DNS allows containers to ping each other by name.

Q: What happens when you try to access the web server from your host machine in the custom network?

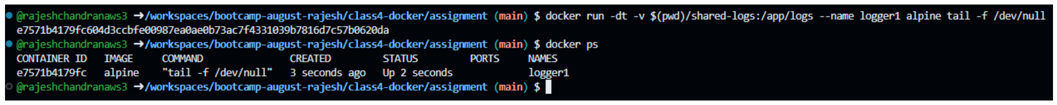
We cannot reach container because port is not published. Inorder to reach the container we have to publish the port using -p.

3. Docker Volumes

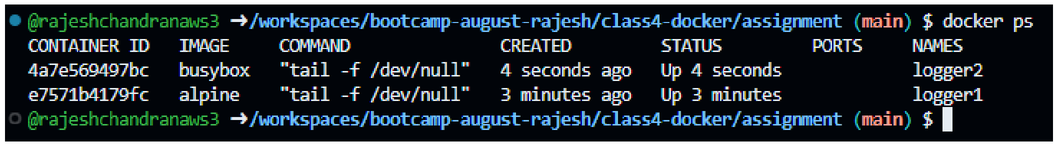
3.1. Bind Mounts

mkdir shared-logs

docker run -d -v $(pwd)/shared-logs:/app/logs --name logger1 alpine tail -f /dev/null
docker run -d -v $(pwd)/shared-logs:/app/logs --name logger2 busybox tail -f /dev/null

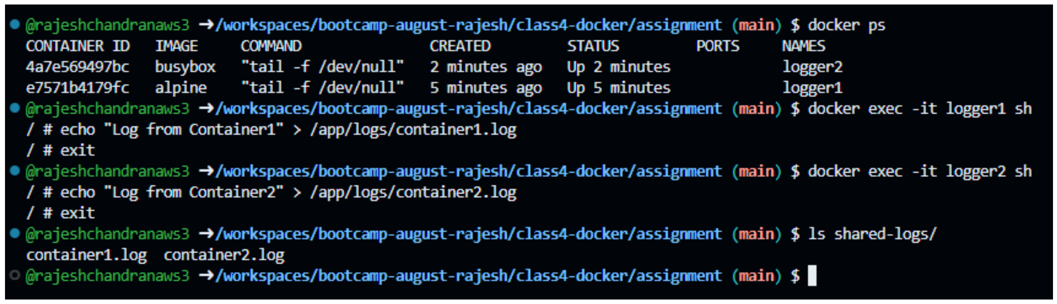


```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker run -dt -v $(pwd)/shared-logs:/app/logs --name logger1 alpine tail -f /dev/null
e7571b4179fc604d3ccbfe00987ea0ae0b73ac7f4331039b7816d7c57b0620da
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE     COMMAND               CREATED         STATUS         PORTS     NAMES
e7571b4179fc   alpine    "tail -f /dev/null"   3 seconds ago   Up 2 seconds             logger1
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```



```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE     COMMAND               CREATED         STATUS         PORTS     NAMES
4a7e569497bc   busybox   "tail -f /dev/null"   4 seconds ago   Up 4 seconds             logger2
e7571b4179fc   alpine    "tail -f /dev/null"   3 minutes ago   Up 3 minutes             logger1
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

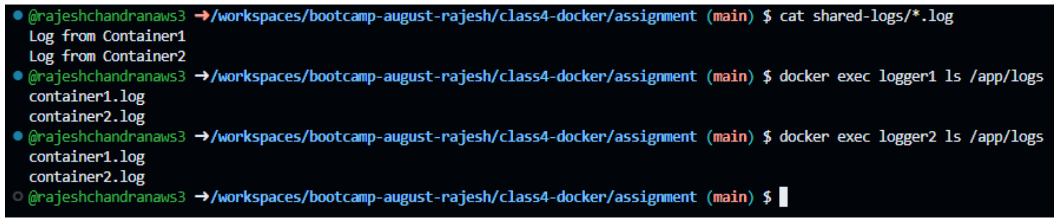
docker exec logger1 sh -c "echo 'Log from container 1' > /app/logs/container1.log"
docker exec logger2 sh -c "echo 'Log from container 2' > /app/logs/container2.log"



```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE     COMMAND               CREATED         STATUS         PORTS     NAMES
4a7e569497bc   busybox   "tail -f /dev/null"   2 minutes ago   Up 2 minutes             logger2
e7571b4179fc   alpine    "tail -f /dev/null"   5 minutes ago   Up 5 minutes             logger1
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -it logger1 sh
/ # echo "Log from Container1" > /app/logs/container1.log
/ # exit
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -it logger2 sh
/ # echo "Log from Container2" > /app/logs/container2.log
/ # exit
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ ls shared-logs/
container1.log  container2.log
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

# Check from host:
ls shared-logs/
cat shared-logs/*.log

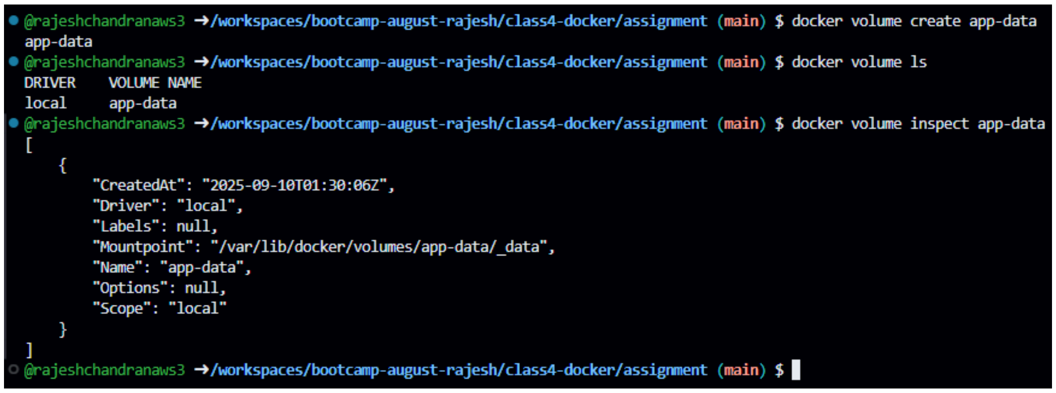
# Check from containers:
docker exec logger1 ls /app/logs/
docker exec logger2 ls /app/logs/



```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ cat shared-logs/*.log
Log from Container1
Log from Container2
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec logger1 ls /app/logs
container1.log
container2.log
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec logger2 ls /app/logs
container1.log
container2.log
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

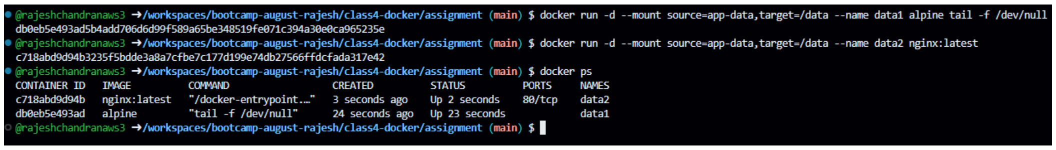
3.2. Docker Volumes

docker volume create app-data
docker volume ls
docker volume inspect app-data



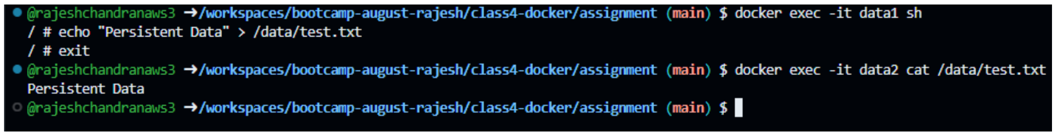
```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker volume create app-data
app-data
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker volume ls
DRIVER    VOLUME NAME
local     app-data
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker volume inspect app-data
[
    {
        "CreatedAt": "2025-09-10T01:30:06Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/app-data/_data",
        "Name": "app-data",
        "Options": null,
        "Scope": "local"
    }
]
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

docker run -d --mount source=app-data,target=/data --name data1 alpine tail -f /dev/null
docker run -d --mount source=app-data,target=/data --name data2 nginx:latest



```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker run -d --mount source=app-data,target=/data --name data1 alpine tail -f /dev/null
db0eb5e493ad5b4add706d6d99f589a65be348519fe071c394a30e0ca965235e
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker run -d --mount source=app-data,target=/data --name data2 nginx:latest
c718abd9d94b3235f5bdde3a8a7cfbe7c177d199e74db27566ffdcfada317e42
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS     NAMES
c718abd9d94b   nginx:latest   "/docker-entrypoint.…"   3 seconds ago    Up 2 seconds    80/tcp    data2
db0eb5e493ad   alpine         "tail -f /dev/null"      24 seconds ago   Up 23 seconds             data1
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

docker exec data1 sh -c "echo 'Persistent data' > /data/test.txt"
docker exec data2 cat /data/test.txt



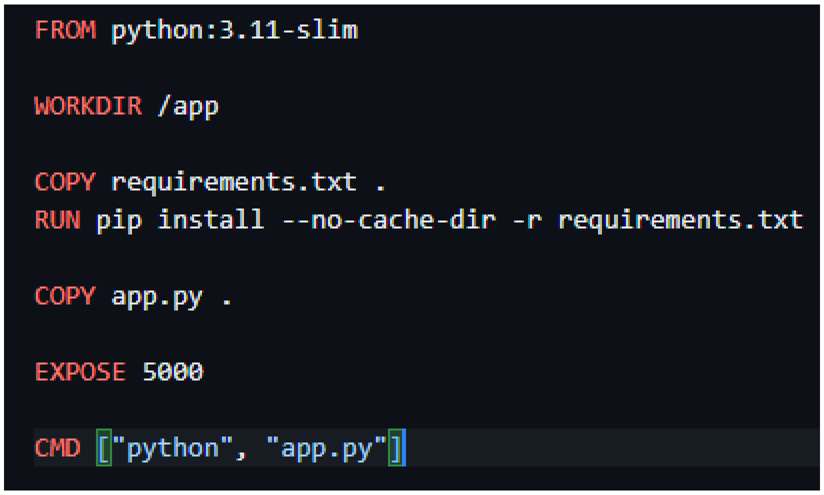
```
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -it data1 sh
/ # echo "Persistent Data" > /data/test.txt
/ # exit
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $ docker exec -it data2 cat /data/test.txt
Persistent Data
@rajeshchandranaws3 ➜/workspaces/bootcamp-august-rajesh/class4-docker/assignment (main) $
```

4. Building Docker Images

4.1. Create a Flask Application

mkdir flask-docker-app
cd flask-docker-app

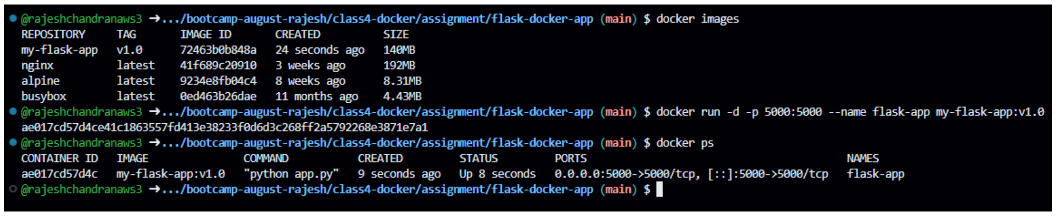
Create app.py and requirements.txt
Create Dockerfile



```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

EXPOSE 5000

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

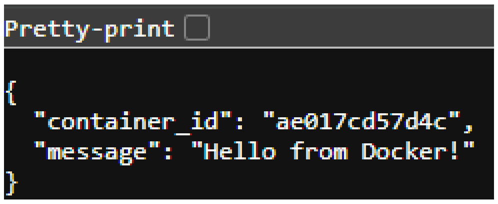
4.2. Build and Test Image

docker build -t my-flask-app:v1.0 .
docker run -d -p 5000:5000 --name flask-app my-flask-app:v1.0



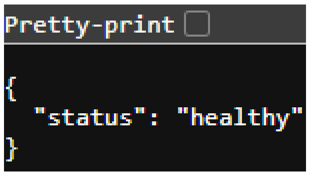
```
● @rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker images
  REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
  my-flask-app   v1.0      72463b0b848a   24 seconds ago   140MB
  nginx          latest    41f689c20910   3 weeks ago      192MB
  alpine         latest    9234e8fb04c4   8 weeks ago      8.31MB
  busybox        latest    0ed463b26dae   11 months ago    4.43MB
● @rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker run -d -p 5000:5000 --name flask-app my-flask-app:v1.0
  ae017cd57d4ce41c1863557fd413e38233f0d6d3c268ff2a5792268e3871e7a1
● @rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker ps
  CONTAINER ID   IMAGE               COMMAND           CREATED         STATUS         PORTS                                       NAMES
  ae017cd57d4c   my-flask-app:v1.0   "python app.py"   9 seconds ago   Up 8 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   flask-app
○ @rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $
```

https://glowing-zebra-5gpprpg745w7f7gg6-5000.app.github.dev/



```
{
  "container_id": "ae017cd57d4c",
  "message": "Hello from Docker!"
}
```

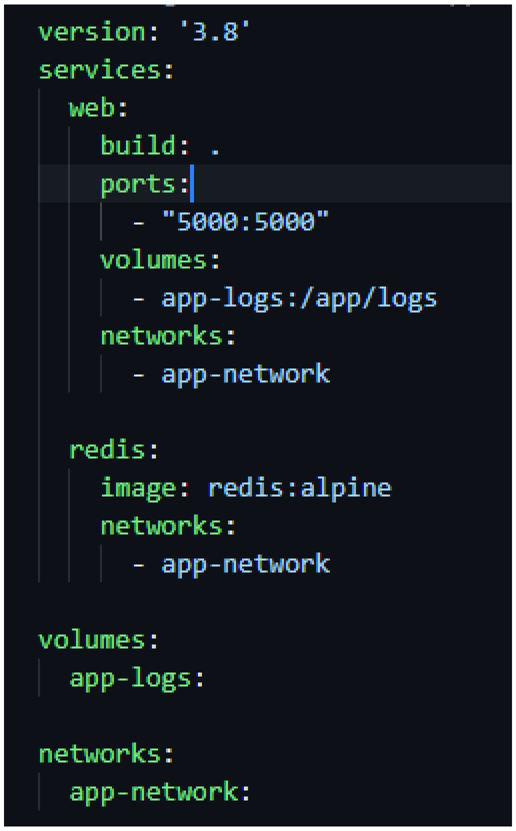
https://glowing-zebra-5gpprpg745w7f7gg6-5000.app.github.dev/health



```
{
  "status": "healthy"
}
```

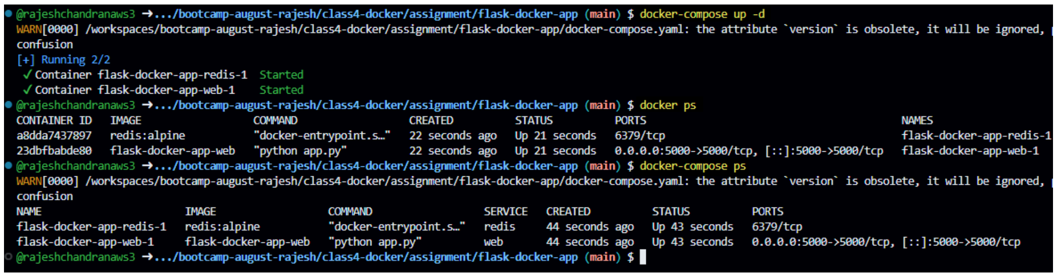
4.3. Multi-container Application

Create docker-compose.yml



```yaml
version: '3.8'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - app-logs:/app/logs
    networks:
      - app-network

  redis:
    image: redis:alpine
    networks:
      - app-network

volumes:
  app-logs:

networks:
  app-network:
```

docker-compose up -d
docker-compose ps

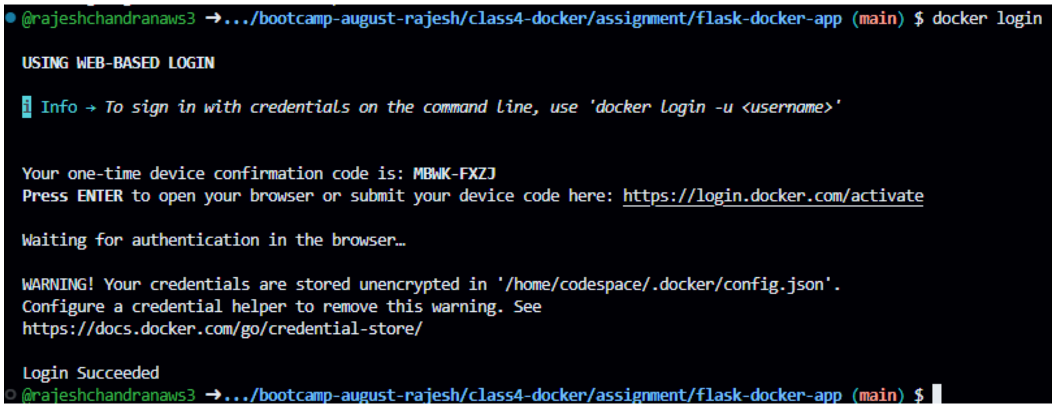


```
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker-compose up -d
WARN[0000] /workspaces/bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app/docker-compose.yaml: the attribute `version` is obsolete, it will be ignored,
confusion
[+] Running 2/2
 ✓ Container flask-docker-app-redis-1  Started
 ✓ Container flask-docker-app-web-1    Started
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                                       NAMES
a8dda7437897   redis:alpine           "docker-entrypoint.s…"   22 seconds ago   Up 21 seconds   6379/tcp                                    flask-docker-app-redis-1
23dbfbabde80   flask-docker-app-web   "python app.py"          22 seconds ago   Up 21 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   flask-docker-app-web-1
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker-compose ps
WARN[0000] /workspaces/bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app/docker-compose.yaml: the attribute `version` is obsolete, it will be ignored,
confusion
NAME                       IMAGE                  COMMAND                  SERVICE   CREATED          STATUS          PORTS
flask-docker-app-redis-1   redis:alpine           "docker-entrypoint.s…"   redis     44 seconds ago   Up 43 seconds   6379/tcp
flask-docker-app-web-1     flask-docker-app-web   "python app.py"          web       44 seconds ago   Up 43 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $
```

5. Image Registry

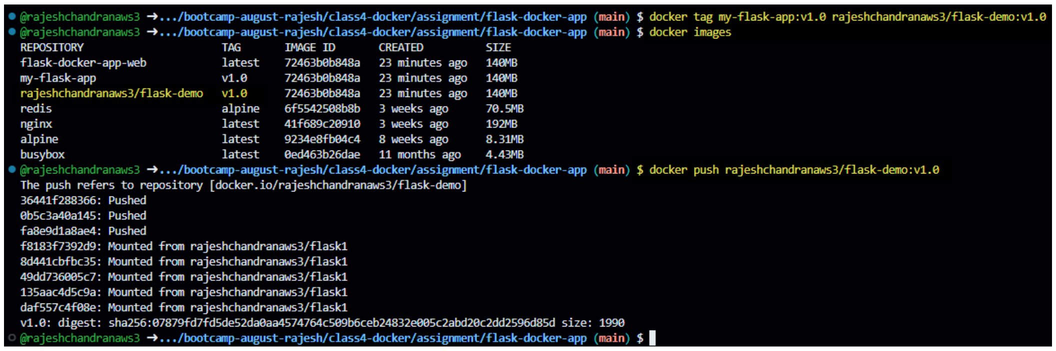
5.1. Push to Docker Hub

docker login

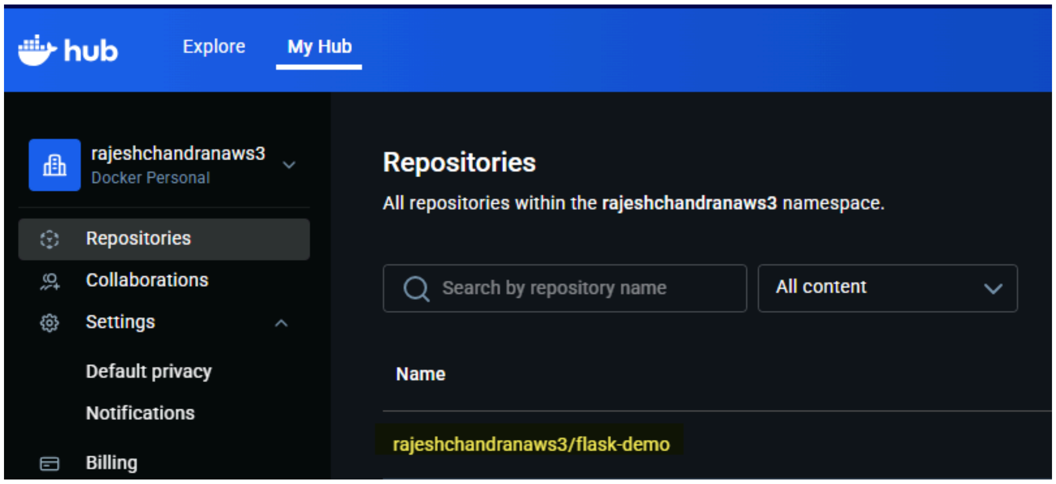


```
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker login

USING WEB-BASED LOGIN

i Info → To sign in with credentials on the command line, use 'docker login -u <username>'


Your one-time device confirmation code is: MBWK-FXZJ
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser…

WARNING! Your credentials are stored unencrypted in '/home/codespace/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $
```

docker tag my-flask-app:v1.0 rajeshchandranaws3/flask-demo:v1.0
docker push rajeshchandranaws3/flask-demo:v1.0



```
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker tag my-flask-app:v1.0 rajeshchandranaws3/flask-demo:v1.0
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker images
REPOSITORY                     TAG       IMAGE ID       CREATED          SIZE
flask-docker-app-web           latest    72463b0b848a   23 minutes ago   140MB
my-flask-app                   v1.0      72463b0b848a   23 minutes ago   140MB
rajeshchandranaws3/flask-demo  v1.0      72463b0b848a   23 minutes ago   140MB
redis                          alpine    6f5542508b8b   3 weeks ago      70.5MB
nginx                          latest    41f689c20910   3 weeks ago      192MB
alpine                         latest    9234e8fb04c4   8 weeks ago      8.31MB
busybox                        latest    0ed463b26dae   11 months ago    4.43MB
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker push rajeshchandranaws3/flask-demo:v1.0
The push refers to repository [docker.io/rajeshchandranaws3/flask-demo]
36441f288366: Pushed
0b5c3a40a145: Pushed
fa8e9d1a8ae4: Pushed
f8183f7392d9: Mounted from rajeshchandranaws3/flask1
8d441cbfbc35: Mounted from rajeshchandranaws3/flask1
49dd736005c7: Mounted from rajeshchandranaws3/flask1
135aac4d5c9a: Mounted from rajeshchandranaws3/flask1
daf557c4f08e: Mounted from rajeshchandranaws3/flask1
v1.0: digest: sha256:07879fd7fd5de52da0aa4574764c509b6ceb24832e005c2abd20c2dd2596d85d size: 1990
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $
```

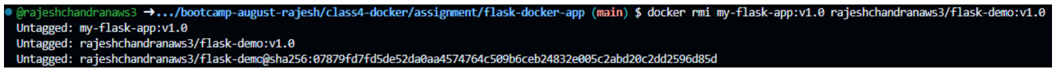


Repositories

All repositories within the **rajeshchandranaws3** namespace.

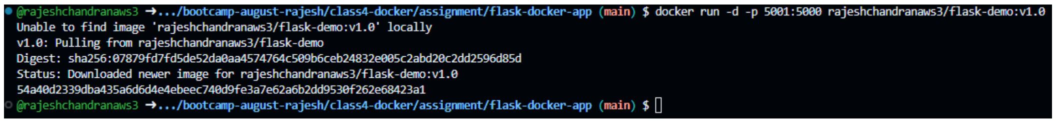
Name

rajeshchandranaws3/flask-demo



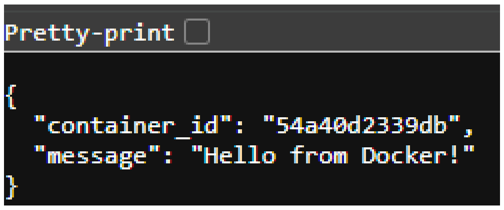
```
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker rmi my-flask-app:v1.0 rajeshchandranaws3/flask-demo:v1.0
Untagged: my-flask-app:v1.0
Untagged: rajeshchandranaws3/flask-demo:v1.0
Untagged: rajeshchandranaws3/flask-demo@sha256:07879fd7fd5de52da0aa4574764c509b6ceb24832e005c2abd20c2dd2596d85d
```

docker rmi my-flask-app:v1.0 rajeshchandranaws3/flask-demo:v1.0
docker run -d -p 5001:5000 rajeshchandranaws3/flask-demo:v1.0



```
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $ docker run -d -p 5001:5000 rajeshchandranaws3/flask-demo:v1.0
Unable to find image 'rajeshchandranaws3/flask-demo:v1.0' locally
v1.0: Pulling from rajeshchandranaws3/flask-demo
Digest: sha256:07879fd7fd5de52da0aa4574764c509b6ceb24832e005c2abd20c2dd2596d85d
Status: Downloaded newer image for rajeshchandranaws3/flask-demo:v1.0
54a40d2339dba435a6d6d4e4ebeec740d9fe3a7e62a6b2dd9530f262e68423a1
@rajeshchandranaws3 ➜.../bootcamp-august-rajesh/class4-docker/assignment/flask-docker-app (main) $
```

https://glowing-zebra-5gpprpg745w7f7gg6-5001.app.github.dev/

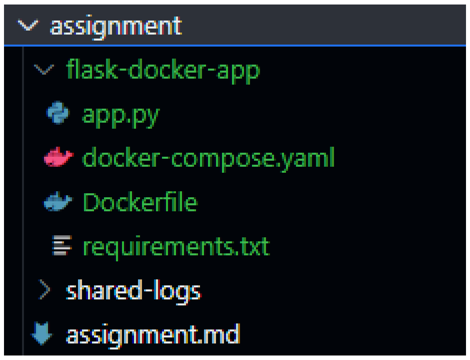


```
{
  "container_id": "54a40d2339db",
  "message": "Hello from Docker!"
}
```

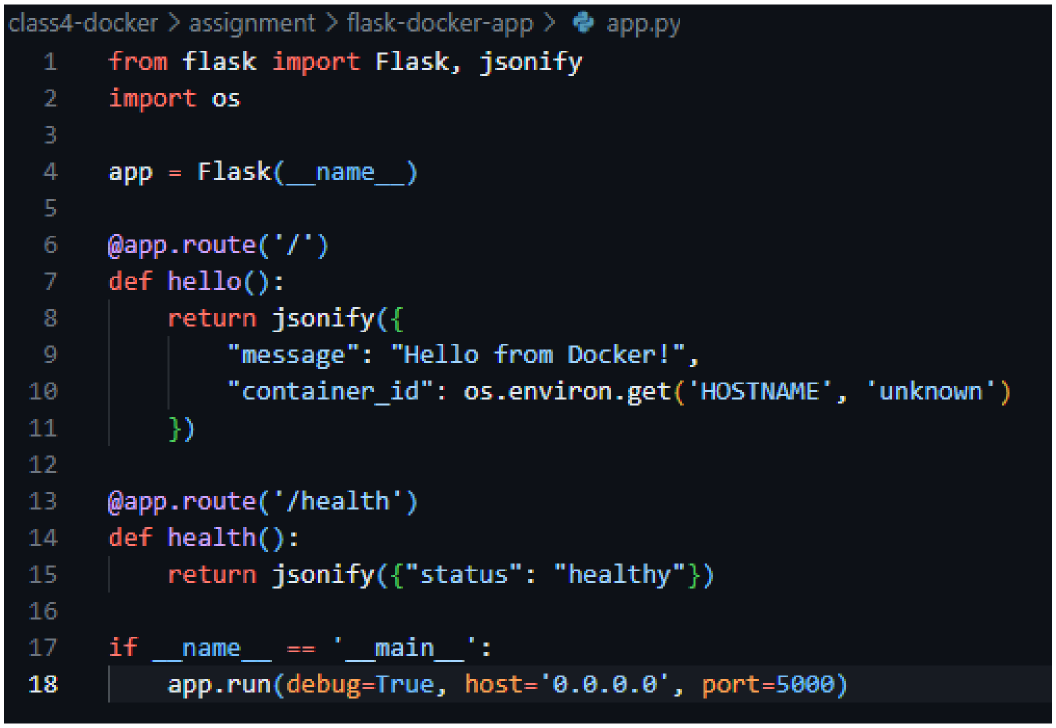
6. Code Repository

6.1. GitHub

- Flask application code
- Dockerfile
- docker-compose.yml
- README.md with setup instructions

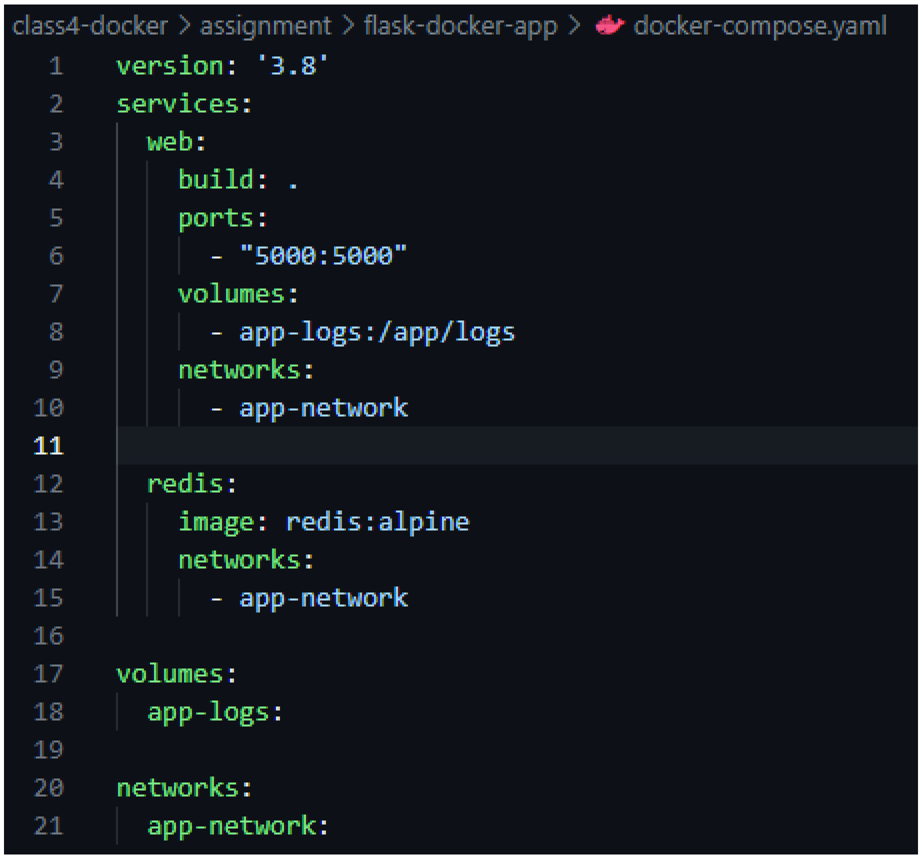


```
assignment
  flask-docker-app
    app.py
    docker-compose.yaml
    Dockerfile
    requirements.txt
  shared-logs
  assignment.md
```

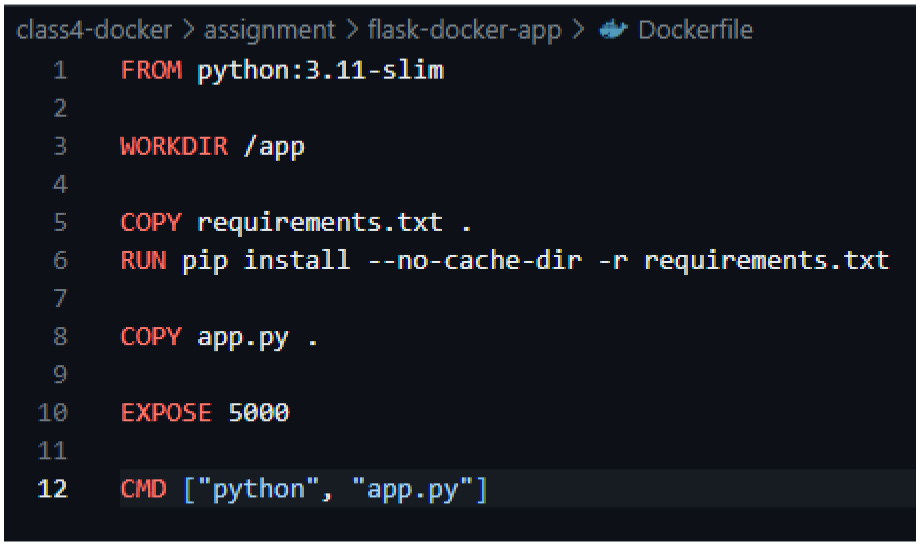


class4-docker > assignment > flask-docker-app > app.py

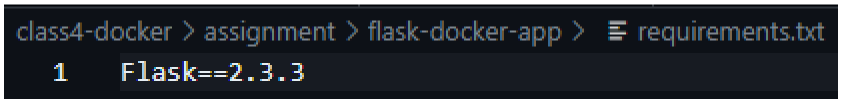
```python
from flask import Flask, jsonify
import os

app = Flask(__name__)

@app.route('/')
def hello():
    return jsonify({
        "message": "Hello from Docker!",
        "container_id": os.environ.get('HOSTNAME', 'unknown')
    })

@app.route('/health')
def health():
    return jsonify({"status": "healthy"})

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```


class4-docker > assignment > flask-docker-app > docker-compose.yaml

```yaml
version: '3.8'
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - app-logs:/app/logs
    networks:
      - app-network

  redis:
    image: redis:alpine
    networks:
      - app-network

volumes:
  app-logs:

networks:
  app-network:
```


class4-docker > assignment > flask-docker-app > Dockerfile

```dockerfile
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

EXPOSE 5000

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


class4-docker > assignment > flask-docker-app > requirements.txt

```
Flask==2.3.3
```

7. Reflection Questions

7.1. Answer Section

7.1.1. Container vs VM: Explain the key differences between Docker containers and virtual machines.

Ans:

- Containers are faster and lighter but less isolated; VMs are more secure but resource-intensive.
- Containers share the host OS kernel, VMs have their own complete OS
- Containers are more portable across environments, VMs are platform-specific

7.1.2. Networking: Why do containers in custom bridge networks have DNS resolution while default bridge network containers don't?

Ans:

- Custom bridge networks have an embedded DNS server, default bridge doesn't have dns.
- Custom networks automatically register container names as hostnames for DNS lookup
- Custom networks resolve container names to IP addresses automatically

7.1.3. Data Persistence: When would you choose bind mounts over Docker volumes and vice versa?

Ans:

- Bind mounts are used for development and direct file access; Docker volumes used for production and managed persistence.
- Bind mounts need specific hostfile path. Docker volumes means docker will manage the storage life cycle

7.1.4. Image Optimization: What strategies could you use to reduce Docker image size?

Ans:

- Multi-Stage build. That means use separate stages for build and runtime.
- Use distroless image. That means start with minimal base image instead of full OS Image.
- Minimize the docker layers.
- Only copy the required files. Do not copy the entire project

7.1.5. Security: What are three security best practices when building Docker images?

Ans:

- Run as non-root
- Scan for vulnerabilities
- Keep images minimal to reduce security risks.

7.1.6. Production Readiness: What additional considerations would you need for running containers in production?

Ans:
Production requires:

- Container orchestration
- Monitoring & Logging
- Security hardening
- High availability.