🧩 Docker Lab Exercises

🔹 Exercise 1: Getting Started with Containers

Objective: Learn how to run and inspect containers.

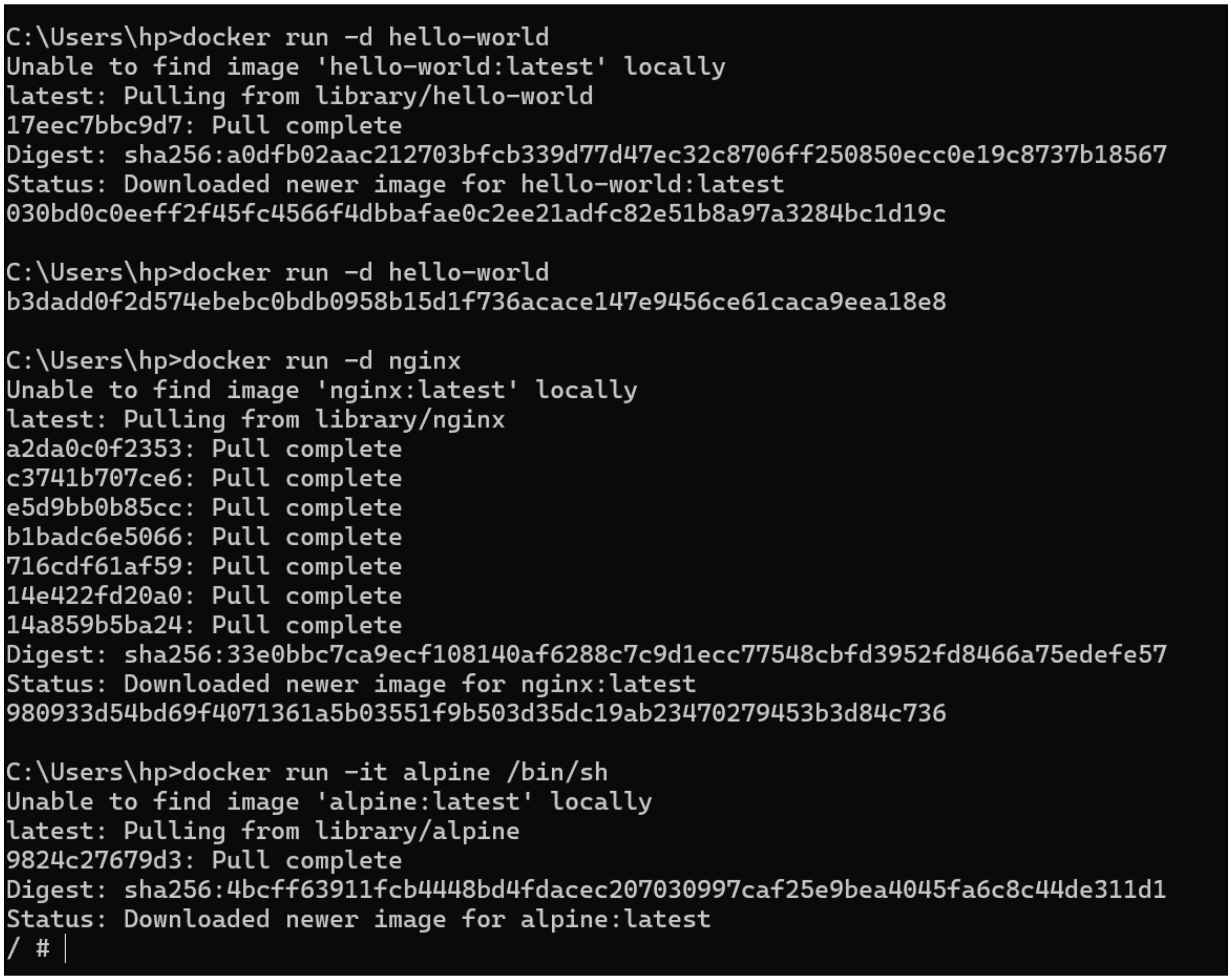
- Run `hello-world`, `nginx`, and `alpine` containers.
- Use `docker ps` and `docker ps -a` to inspect states.
- Explore `docker run` flags: `--rm`, `-it`, `-d`, `-p`.
- Use `docker exec` and `docker logs`.

Checkpoint:
What happens if a container doesn't run in detached mode? What if ports aren't mapped?

If a container doesn’t run in detached mode, it will be run in the same terminal where we are writing commands. So we will enter the container directly in our terminal. While using detached mode enables us to run a container in background while working on other things in our terminal.

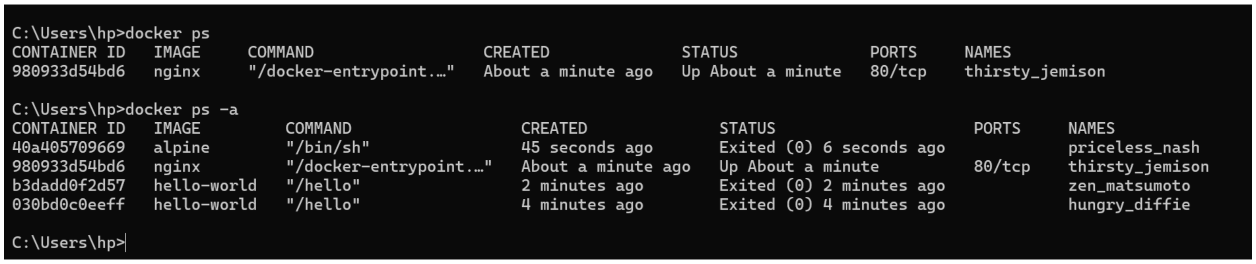
The container's internal ports exist only inside the container's isolated network namespace. These container ports are not accessible from outside the container or the host machine. Therefore, we might not be able to access the ports inside the container from outside the host machine pr container.

- docker run -d hello-world
- docker run -d nginx
- docker run -it alpine /bin/sh



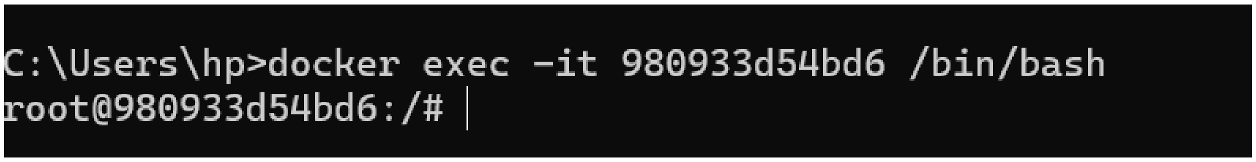
```
C:\Users\hp>docker run -d hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:a0dfb02aac212703bfcb339d77d47ec32c8706ff250850ecc0e19c8737b18567
Status: Downloaded newer image for hello-world:latest
030bd0c0eeff2f45fc4566f4dbbafae0c2ee21adfc82e51b8a97a3284bc1d19c

C:\Users\hp>docker run -d hello-world
b3dadd0f2d574ebebc0bdb0958b15d1f736acace147e9456ce61caca9eea18e8

C:\Users\hp>docker run -d nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
a2da0c0f2353: Pull complete
c3741b707ce6: Pull complete
e5d9bb0b85cc: Pull complete
b1badc6e5066: Pull complete
716cdf61af59: Pull complete
14e422fd20a0: Pull complete
14a859b5ba24: Pull complete
Digest: sha256:33e0bbc7ca9ecf108140af6288c7c9d1ecc77548cbfd3952fd8466a75edefe57
Status: Downloaded newer image for nginx:latest
980933d54bd69f4071361a5b03551f9b503d35dc19ab23470279453b3d84c736

C:\Users\hp>docker run -it alpine /bin/sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
9824c27679d3: Pull complete
Digest: sha256:4bcff63911fcb4448bd4fdacec207030997caf25e9bea4045fa6c8c44de311d1
Status: Downloaded newer image for alpine:latest
/ #
```

- docker ps
- docker ps -a



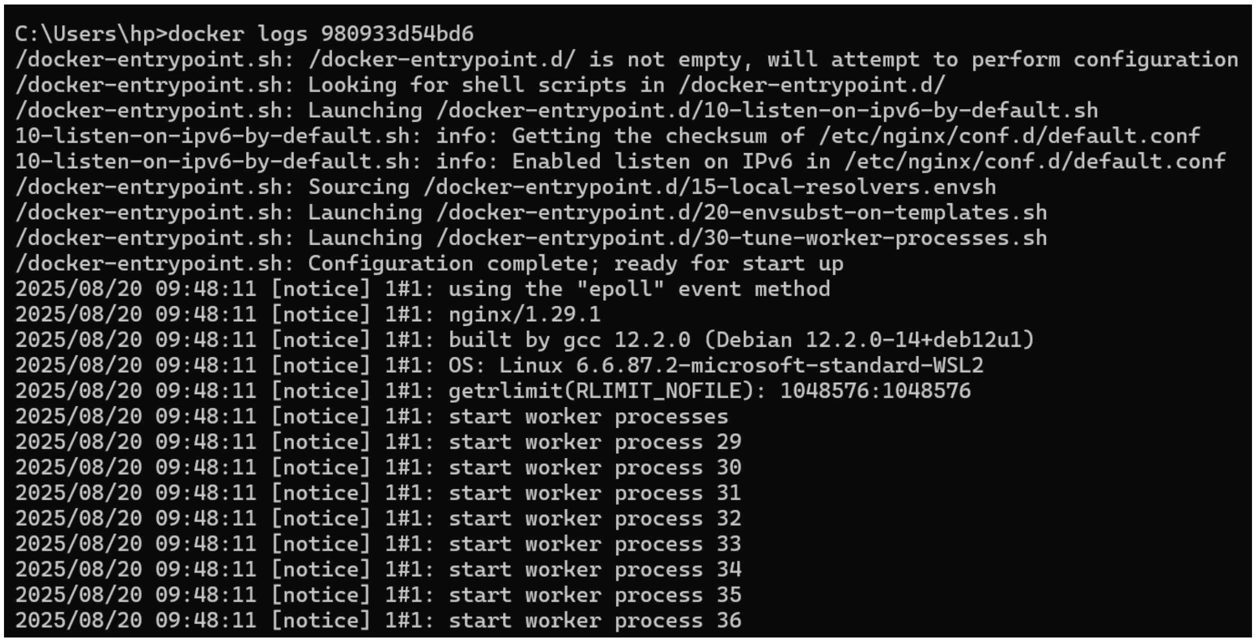
```
C:\Users\hp>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS     NAMES
980933d54bd6   nginx     "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    thirsty_jemison

C:\Users\hp>docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED              STATUS                      PORTS     NAMES
40a405709669   alpine        "/bin/sh"                45 seconds ago       Exited (0) 6 seconds ago              priceless_nash
980933d54bd6   nginx         "/docker-entrypoint.…"   About a minute ago   Up About a minute           80/tcp    thirsty_jemison
b3dadd0f2d57   hello-world   "/hello"                 2 minutes ago        Exited (0) 2 minutes ago              zen_matsumoto
030bd0c0eeff   hello-world   "/hello"                 4 minutes ago        Exited (0) 4 minutes ago              hungry_diffie

C:\Users\hp>
```

- docker exec -it 980933d54bd6 /bin/bash



```
C:\Users\hp>docker exec -it 980933d54bd6 /bin/bash
root@980933d54bd6:/#
```

- docker logs 980933d54bd6



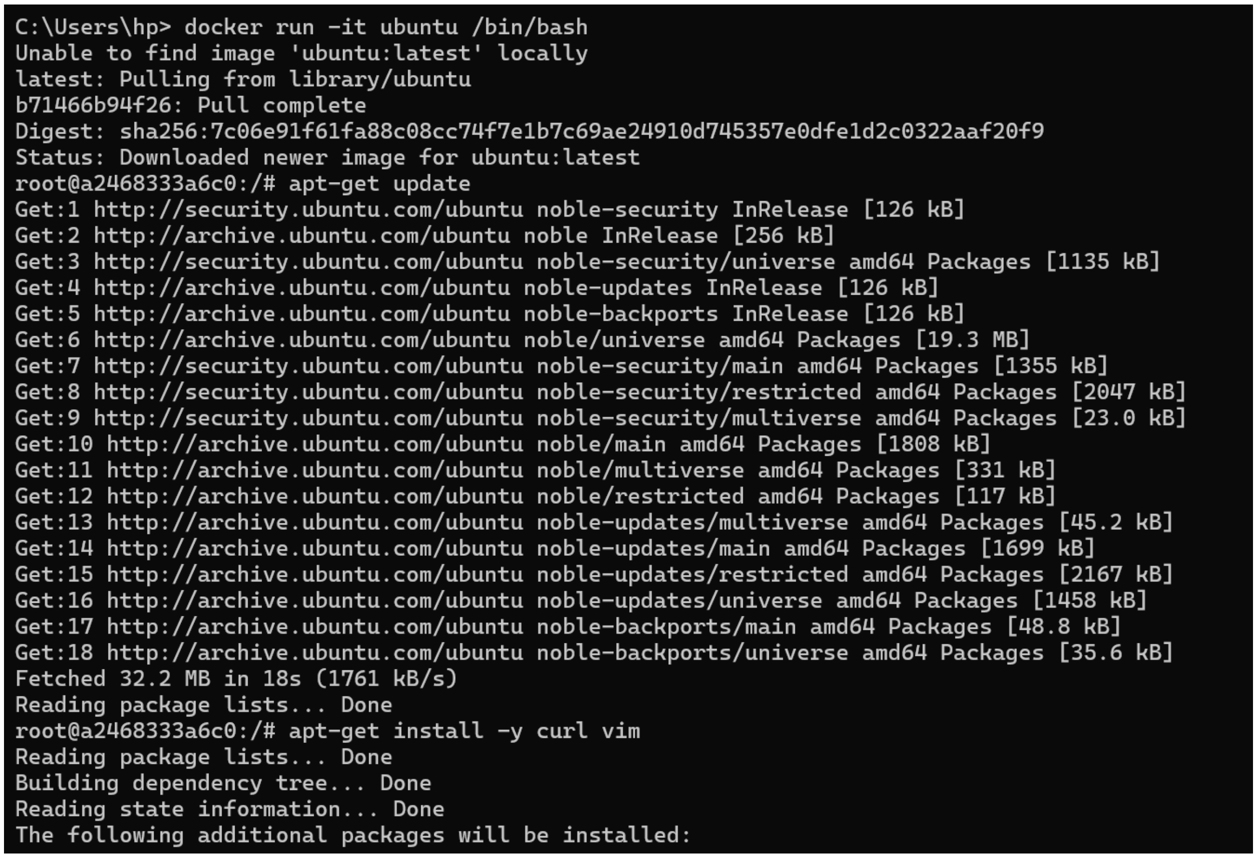
```
C:\Users\hp>docker logs 980933d54bd6
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/08/20 09:48:11 [notice] 1#1: using the "epoll" event method
2025/08/20 09:48:11 [notice] 1#1: nginx/1.29.1
2025/08/20 09:48:11 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14+deb12u1)
2025/08/20 09:48:11 [notice] 1#1: OS: Linux 6.6.87.2-microsoft-standard-WSL2
2025/08/20 09:48:11 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/08/20 09:48:11 [notice] 1#1: start worker processes
2025/08/20 09:48:11 [notice] 1#1: start worker process 29
2025/08/20 09:48:11 [notice] 1#1: start worker process 30
2025/08/20 09:48:11 [notice] 1#1: start worker process 31
2025/08/20 09:48:11 [notice] 1#1: start worker process 32
2025/08/20 09:48:11 [notice] 1#1: start worker process 33
2025/08/20 09:48:11 [notice] 1#1: start worker process 34
2025/08/20 09:48:11 [notice] 1#1: start worker process 35
2025/08/20 09:48:11 [notice] 1#1: start worker process 36
```

🔹 Exercise 2: Working with Container State

Objective: Modify containers and commit custom images.

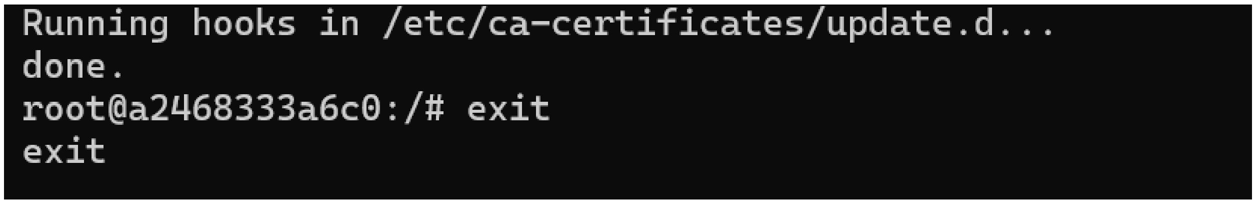
- Run an Ubuntu container, install `curl` and `vim`.
- Exit and commit the image as `ubuntu-tools`.
- Run a new container from the committed image.
- Tag the image and list it with `docker images`.

- docker run -it ubuntu /bin/bash
- apt-get update
- apt-get install -y curl vim



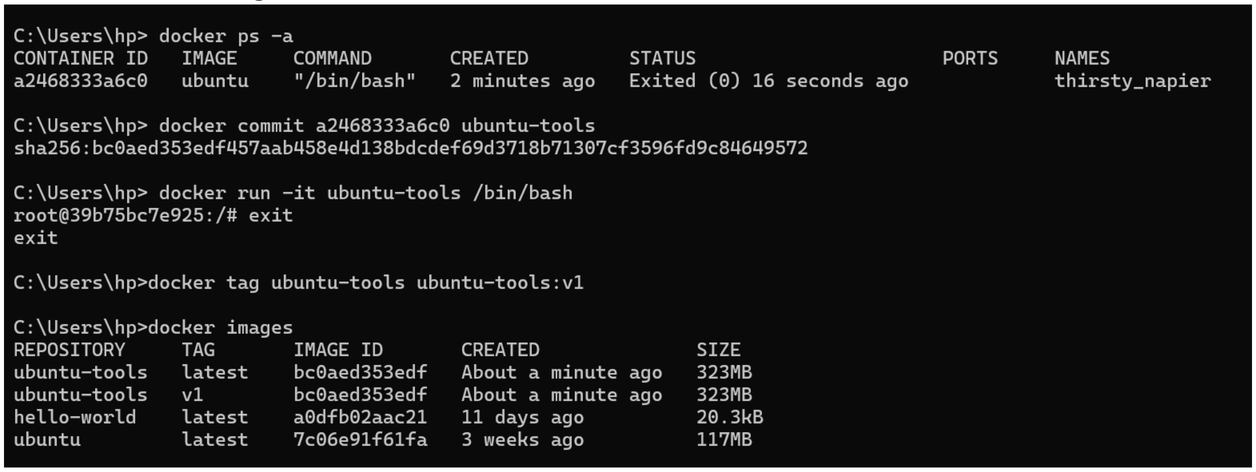
```
C:\Users\hp> docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
b71466b94f26: Pull complete
Digest: sha256:7c06e91f61fa88c08cc74f7e1b7c69ae24910d745357e0dfe1d2c0322aaf20f9
Status: Downloaded newer image for ubuntu:latest
root@a2468333a6c0:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1135 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1355 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2047 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [23.0 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [45.2 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1699 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2167 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1458 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [48.8 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [35.6 kB]
Fetched 32.2 MB in 18s (1761 kB/s)
Reading package lists... Done
root@a2468333a6c0:/# apt-get install -y curl vim
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

- exit



```
Running hooks in /etc/ca-certificates/update.d...
done.
root@a2468333a6c0:/# exit
exit
```

- docker ps -a
- docker commit a2468333a6c0 ubuntu-tools
- docker run -it ubuntu-tools /bin/bash
- docker tag ubuntu-tools ubuntu-tools:v1

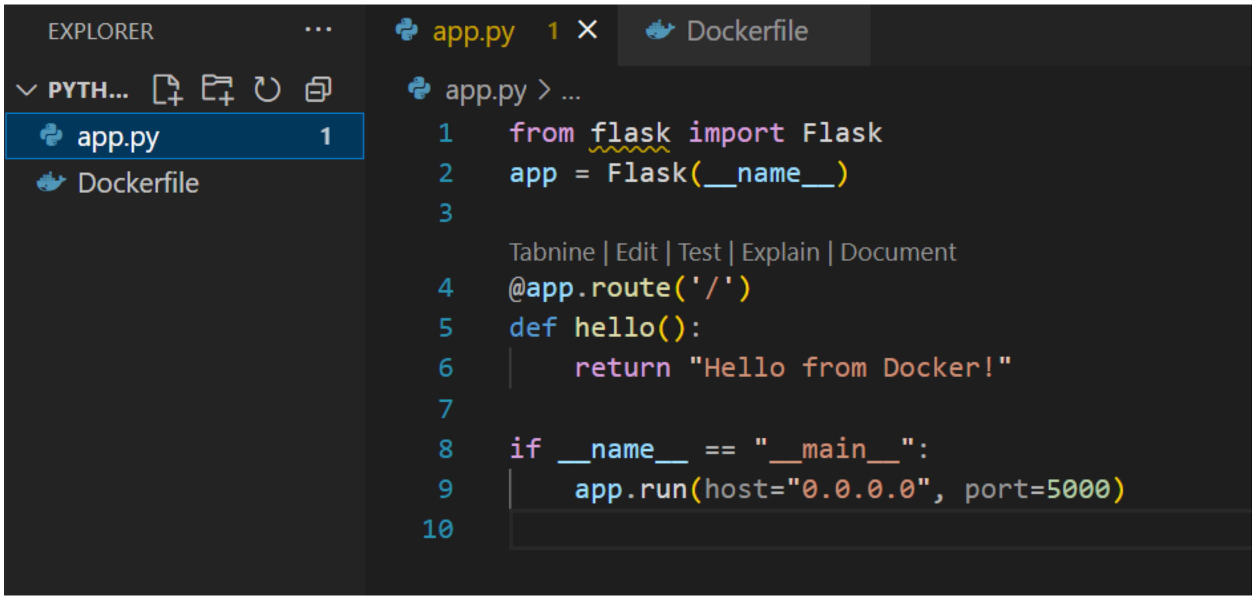


```
C:\Users\hp> docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS                        PORTS     NAMES
a2468333a6c0   ubuntu    "/bin/bash"   2 minutes ago   Exited (0) 16 seconds ago               thirsty_napier

C:\Users\hp> docker commit a2468333a6c0 ubuntu-tools
sha256:bc0aed353edf457aab458e4d138bdcdef69d3718b71307cf3596fd9c84649572

C:\Users\hp> docker run -it ubuntu-tools /bin/bash
root@39b75bc7e925:/# exit
exit

C:\Users\hp>docker tag ubuntu-tools ubuntu-tools:v1

C:\Users\hp>docker images
REPOSITORY     TAG       IMAGE ID       CREATED              SIZE
ubuntu-tools   latest    bc0aed353edf   About a minute ago   323MB
ubuntu-tools   v1        bc0aed353edf   About a minute ago   323MB
hello-world    latest    a0dfb02aac21   11 days ago          20.3kB
ubuntu         latest    7c06e91f61fa   3 weeks ago          117MB
```

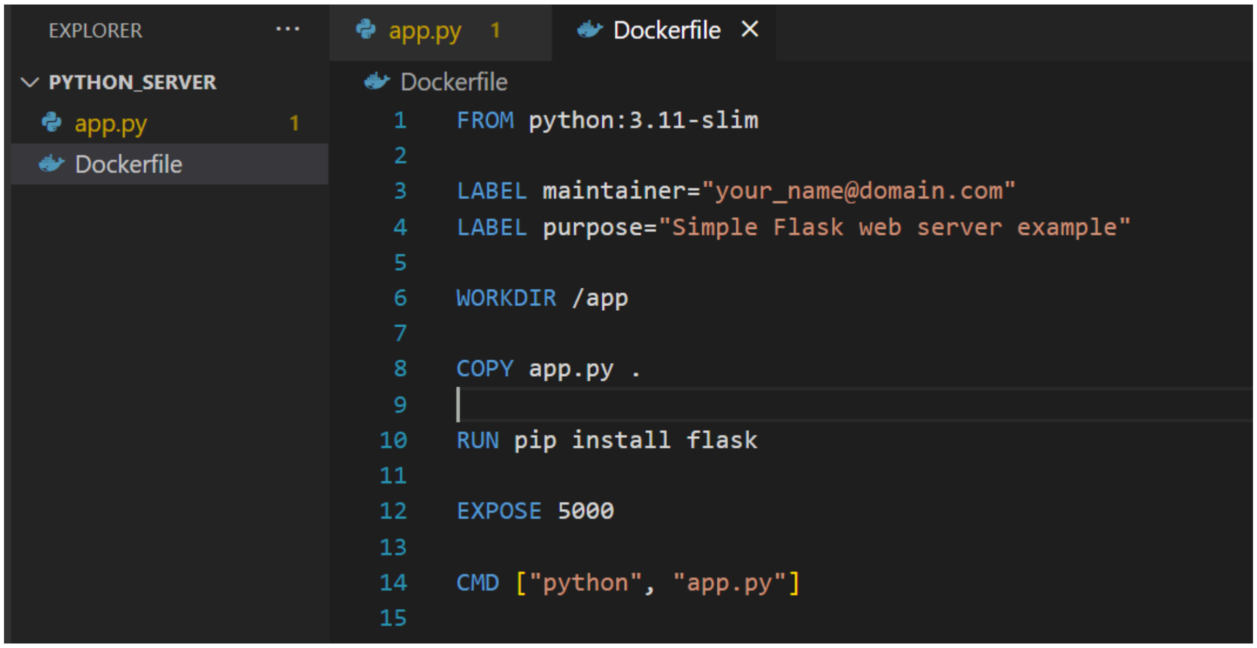
🔹 Exercise 3: Build Custom Images Using Dockerfile

Objective: Write Dockerfiles and build your own image.

- Create a simple Node or Python web server.
- Write a Dockerfile to copy the code and expose a port.
- Add metadata using `LABEL` and set `CMD` or `ENTRYPOINT`.
- Build and run the image. Test with `curl`.

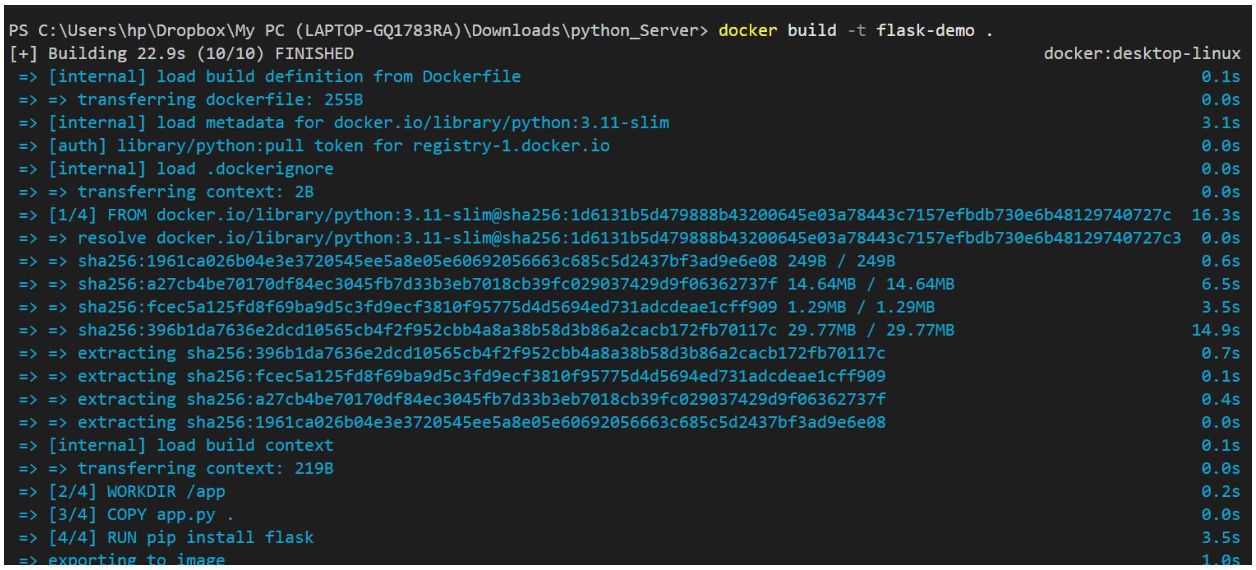


```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello from Docker!"

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



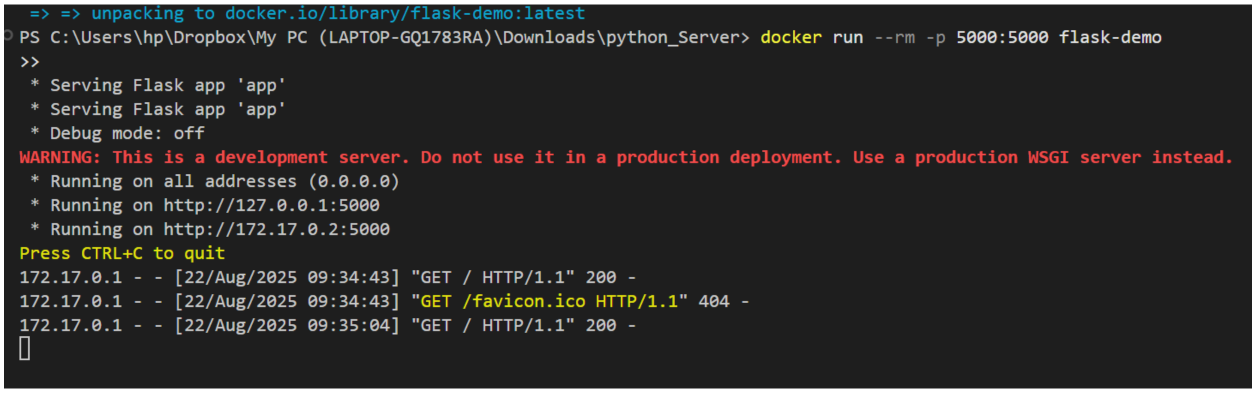
```dockerfile
FROM python:3.11-slim

LABEL maintainer="your_name@domain.com"
LABEL purpose="Simple Flask web server example"

WORKDIR /app

COPY app.py .

RUN pip install flask

EXPOSE 5000

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

- docker build -t flask-demo .

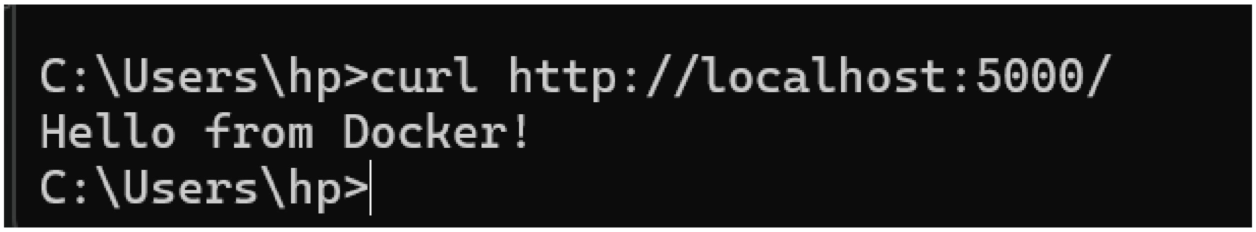


```
PS C:\Users\hp\Dropbox\My PC (LAPTOP-GQ1783RA)\Downloads\python_Server> docker build -t flask-demo .
[+] Building 22.9s (10/10) FINISHED                                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                         0.1s
 => => transferring dockerfile: 255B                                                                         0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                         3.1s
 => [auth] library/python:pull token for registry-1.docker.io                                                0.0s
 => [internal] load .dockerignore                                                                            0.0s
 => => transferring context: 2B                                                                              0.0s
 => [1/4] FROM docker.io/library/python:3.11-slim@sha256:1d6131b5d479888b43200645e03a78443c7157efbdb730e6b48129740727c  16.3s
 => => resolve docker.io/library/python:3.11-slim@sha256:1d6131b5d479888b43200645e03a78443c7157efbdb730e6b48129740727c3  0.0s
 => => sha256:1961ca026b04e3e3720545ee5a8e05e60692056663c685c5d2437bf3ad9e6e08 249B / 249B                    0.6s
 => => sha256:a27cb4be70170df84ec3045fb7d33b3eb7018cb39fc029037429d9f06362737f 14.64MB / 14.64MB              6.5s
 => => sha256:fcec5a125fd8f69ba9d5c3fd9ecf3810f95775d4d5694ed731adcdeae1cff909 1.29MB / 1.29MB                3.5s
 => => sha256:396b1da7636e2dcd10565cb4f2f952cbb4a8a38b58d3b86a2cacb172fb70117c 29.77MB / 29.77MB             14.9s
 => => extracting sha256:396b1da7636e2dcd10565cb4f2f952cbb4a8a38b58d3b86a2cacb172fb70117c                     0.7s
 => => extracting sha256:fcec5a125fd8f69ba9d5c3fd9ecf3810f95775d4d5694ed731adcdeae1cff909                     0.1s
 => => extracting sha256:a27cb4be70170df84ec3045fb7d33b3eb7018cb39fc029037429d9f06362737f                     0.4s
 => => extracting sha256:1961ca026b04e3e3720545ee5a8e05e60692056663c685c5d2437bf3ad9e6e08                     0.0s
 => [internal] load build context                                                                            0.1s
 => => transferring context: 219B                                                                            0.0s
 => [2/4] WORKDIR /app                                                                                       0.2s
 => [3/4] COPY app.py .                                                                                      0.0s
 => [4/4] RUN pip install flask                                                                              3.5s
 => exporting to image                                                                                       1.0s
```

- docker run –rm -p 5000:5000 flask-demo



```
 => => unpacking to docker.io/library/flask-demo:latest
PS C:\Users\hp\Dropbox\My PC (LAPTOP-GQ1783RA)\Downloads\python_Server> docker run --rm -p 5000:5000 flask-demo
>>
 * Serving Flask app 'app'
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [22/Aug/2025 09:34:43] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [22/Aug/2025 09:34:43] "GET /favicon.ico HTTP/1.1" 404 -
172.17.0.1 - - [22/Aug/2025 09:35:04] "GET / HTTP/1.1" 200 -
```



```
C:\Users\hp>curl http://localhost:5000/
Hello from Docker!
C:\Users\hp>
```

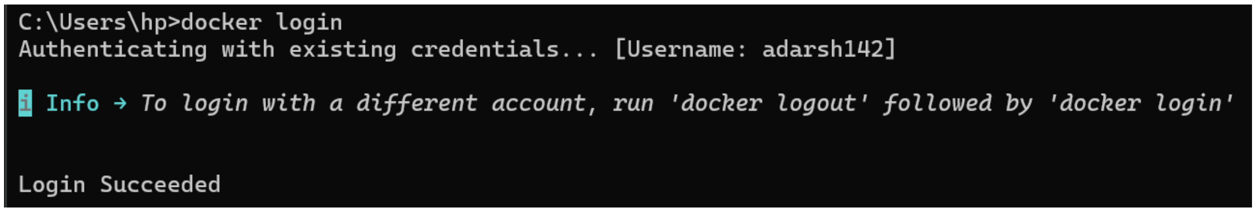
🔹 Exercise 4: Sharing Images

Objective: Push images to Docker Hub.

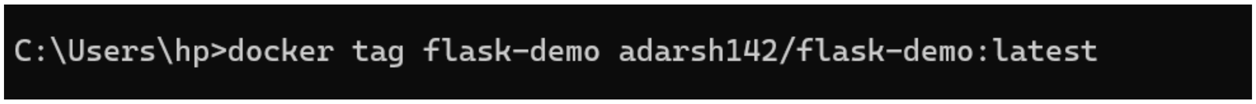
- Create a Docker Hub account.
- Tag your custom image.
- Push it to Docker Hub.
- Pull it from Docker Hub

Reflection:
Why is image tagging important, and what sort of tagging strategy can we use?

- docker login

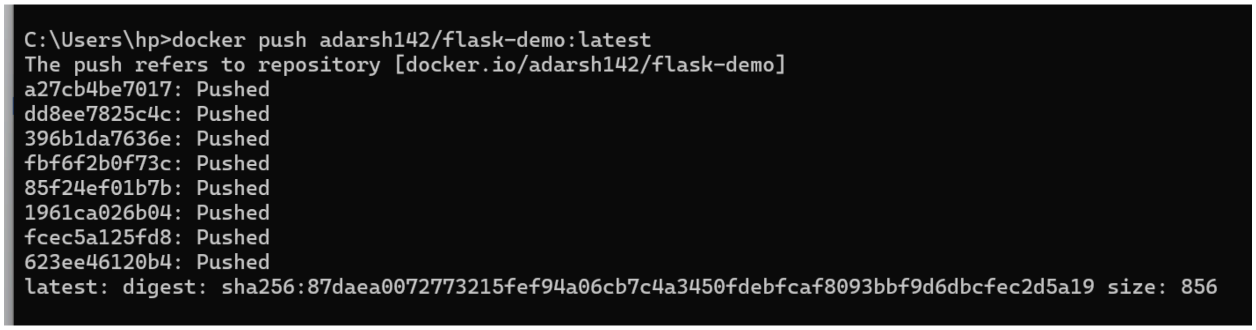


```
C:\Users\hp>docker login
Authenticating with existing credentials... [Username: adarsh142]

i Info → To login with a different account, run 'docker logout' followed by 'docker login'


Login Succeeded
```

- 

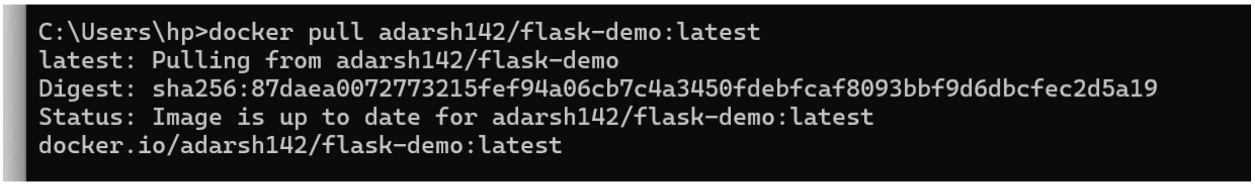
```
C:\Users\hp>docker tag flask-demo adarsh142/flask-demo:latest
```

- docker push adarsh142/flask-demo:latest

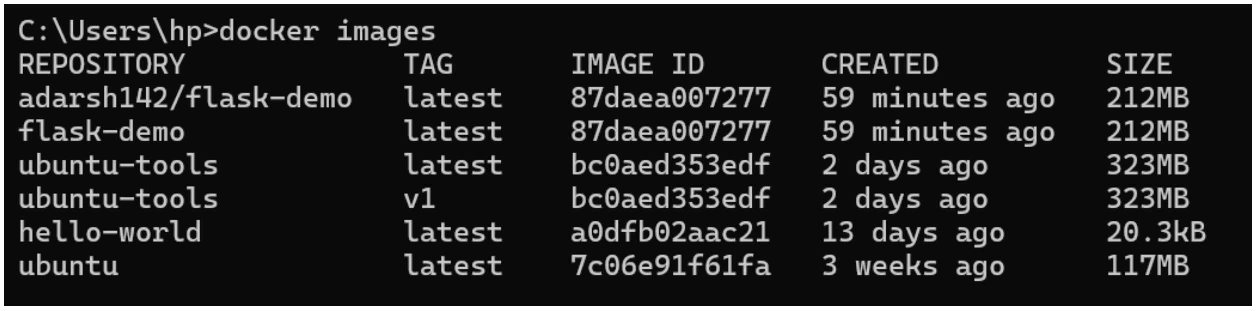


```
C:\Users\hp>docker push adarsh142/flask-demo:latest
The push refers to repository [docker.io/adarsh142/flask-demo]
a27cb4be7017: Pushed
dd8ee7825c4c: Pushed
396b1da7636e: Pushed
fbf6f2b0f73c: Pushed
85f24ef01b7b: Pushed
1961ca026b04: Pushed
fcec5a125fd8: Pushed
623ee46120b4: Pushed
latest: digest: sha256:87daea0072773215fef94a06cb7c4a3450fdebfcaf8093bbf9d6dbcfec2d5a19 size: 856
```

- docker pull adarsh142/flask-demo:latest



```
C:\Users\hp>docker pull adarsh142/flask-demo:latest
latest: Pulling from adarsh142/flask-demo
Digest: sha256:87daea0072773215fef94a06cb7c4a3450fdebfcaf8093bbf9d6dbcfec2d5a19
Status: Image is up to date for adarsh142/flask-demo:latest
docker.io/adarsh142/flask-demo:latest
```



```
C:\Users\hp>docker images
REPOSITORY             TAG       IMAGE ID       CREATED          SIZE
adarsh142/flask-demo   latest    87daea007277   59 minutes ago   212MB
flask-demo             latest    87daea007277   59 minutes ago   212MB
ubuntu-tools           latest    bc0aed353edf   2 days ago       323MB
ubuntu-tools           v1        bc0aed353edf   2 days ago       323MB
hello-world            latest    a0dfb02aac21   13 days ago      20.3kB
ubuntu                 latest    7c06e91f61fa   3 weeks ago      117MB
```

- Image tagging in Docker is important because it helps you organize, identify, and manage different versions of your Docker images efficiently. Tags act like version labels, allowing you to specify exactly which image version to pull, deploy, or share.

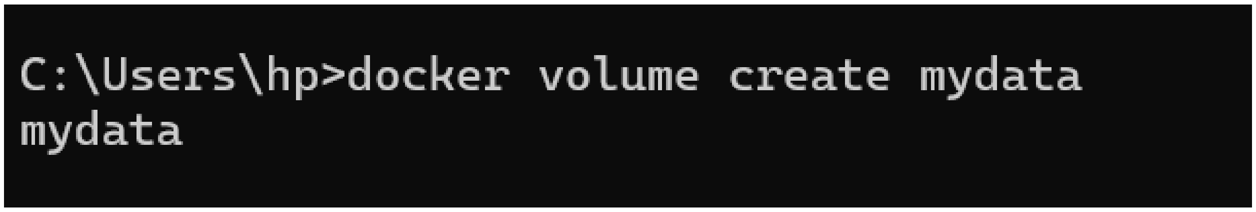
- We can use :latest tagging strategy or :1.0.0 like versioning strategy to tag our images as we keep building upon them.

🔹 Exercise 5: Data Persistence with Volumes

Objective: Use volumes to persist container data.

- Launch a busybox container with a named volume.
- Insert sample data.
- Stop, remove, and relaunch to verify persistence.

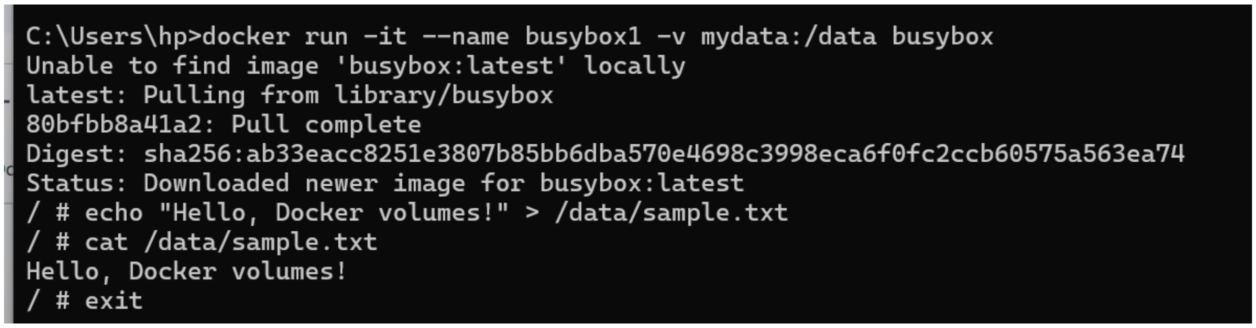
- docker volume create mydata



```
C:\Users\hp>docker volume create mydata
mydata
```

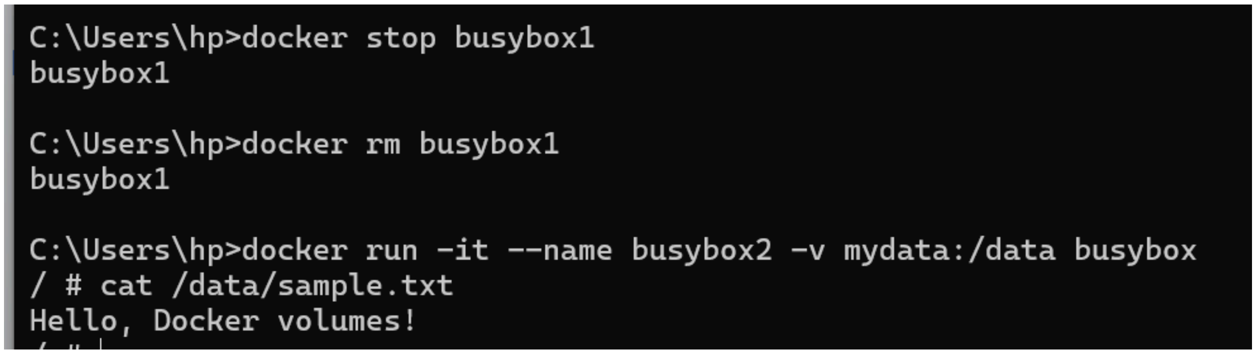
- docker run -it --name busybox1 -v mydata:/data busybox
- / # echo "Hello, Docker volumes!" > /data/sample.txt

- / # cat /data/sample.txt
- exit



```
C:\Users\hp>docker run -it --name busybox1 -v mydata:/data busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
80bfbb8a41a2: Pull complete
Digest: sha256:ab33eacc8251e3807b85bb6dba570e4698c3998eca6f0fc2ccb60575a563ea74
Status: Downloaded newer image for busybox:latest
/ # echo "Hello, Docker volumes!" > /data/sample.txt
/ # cat /data/sample.txt
Hello, Docker volumes!
/ # exit
```

- docker stop busybox1
- docker rm busybox1
- docker run -it --name busybox2 -v mydata:/data busybox
- / # cat /data/sample.txt



```
C:\Users\hp>docker stop busybox1
busybox1

C:\Users\hp>docker rm busybox1
busybox1

C:\Users\hp>docker run -it --name busybox2 -v mydata:/data busybox
/ # cat /data/sample.txt
Hello, Docker volumes!
```

🔹 Exercise 6: Container Networking Basics

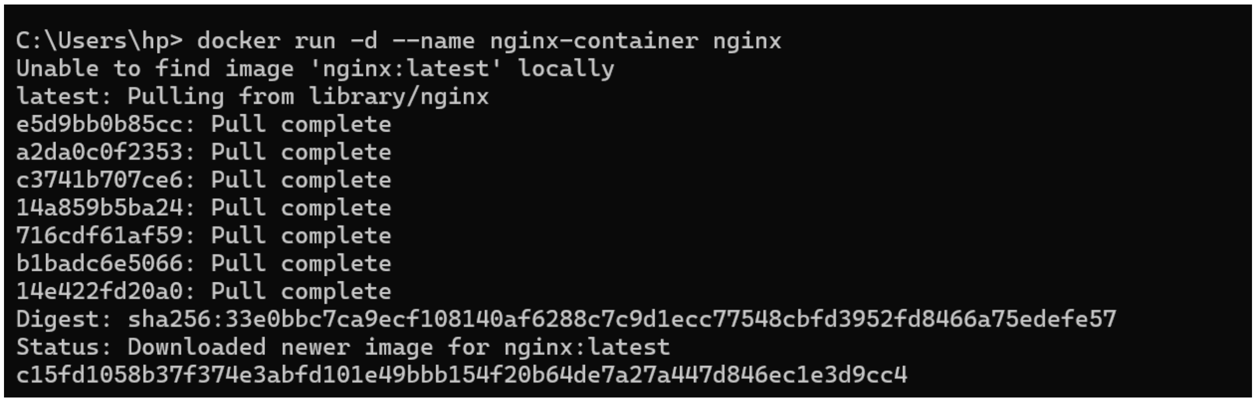
Objective: Set up communication between containers.

- Start an `nginx` container and a `busybox` container.
- Create a user-defined bridge network.
- Attach both containers to the network.
- From busybox, use `wget` or `curl` to access nginx.

Explore:

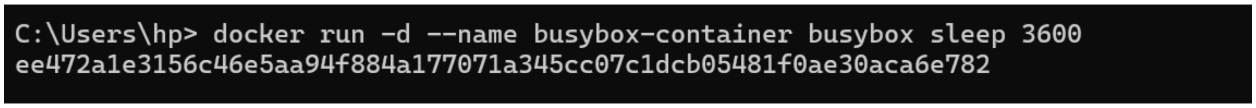
View IPs with `docker inspect`. Try without a custom network—what's different?

- docker run -d --name nginx-container nginx



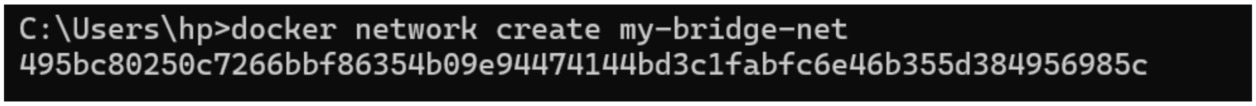
```
C:\Users\hp> docker run -d --name nginx-container nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
e5d9bb0b85cc: Pull complete
a2da0c0f2353: Pull complete
c3741b707ce6: Pull complete
14a859b5ba24: Pull complete
716cdf61af59: Pull complete
b1badc6e5066: Pull complete
14e422fd20a0: Pull complete
Digest: sha256:33e0bbc7ca9ecf108140af6288c7c9d1ecc77548cbfd3952fd8466a75edefe57
Status: Downloaded newer image for nginx:latest
c15fd1058b37f374e3abfd101e49bbb154f20b64de7a27a447d846ec1e3d9cc4
```

- docker run -d --name busybox-container busybox sleep 3600



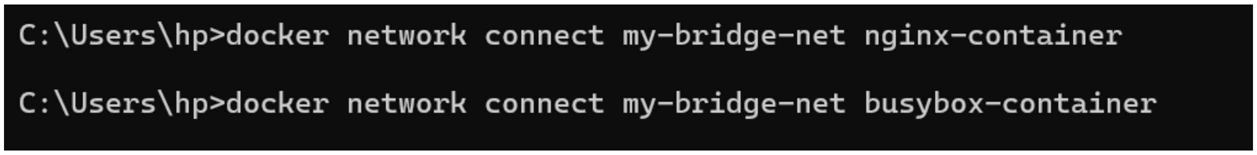
```
C:\Users\hp> docker run -d --name busybox-container busybox sleep 3600
ee472a1e3156c46e5aa94f884a177071a345cc07c1dcb05481f0ae30aca6e782
```

- docker network create my-bridge-net



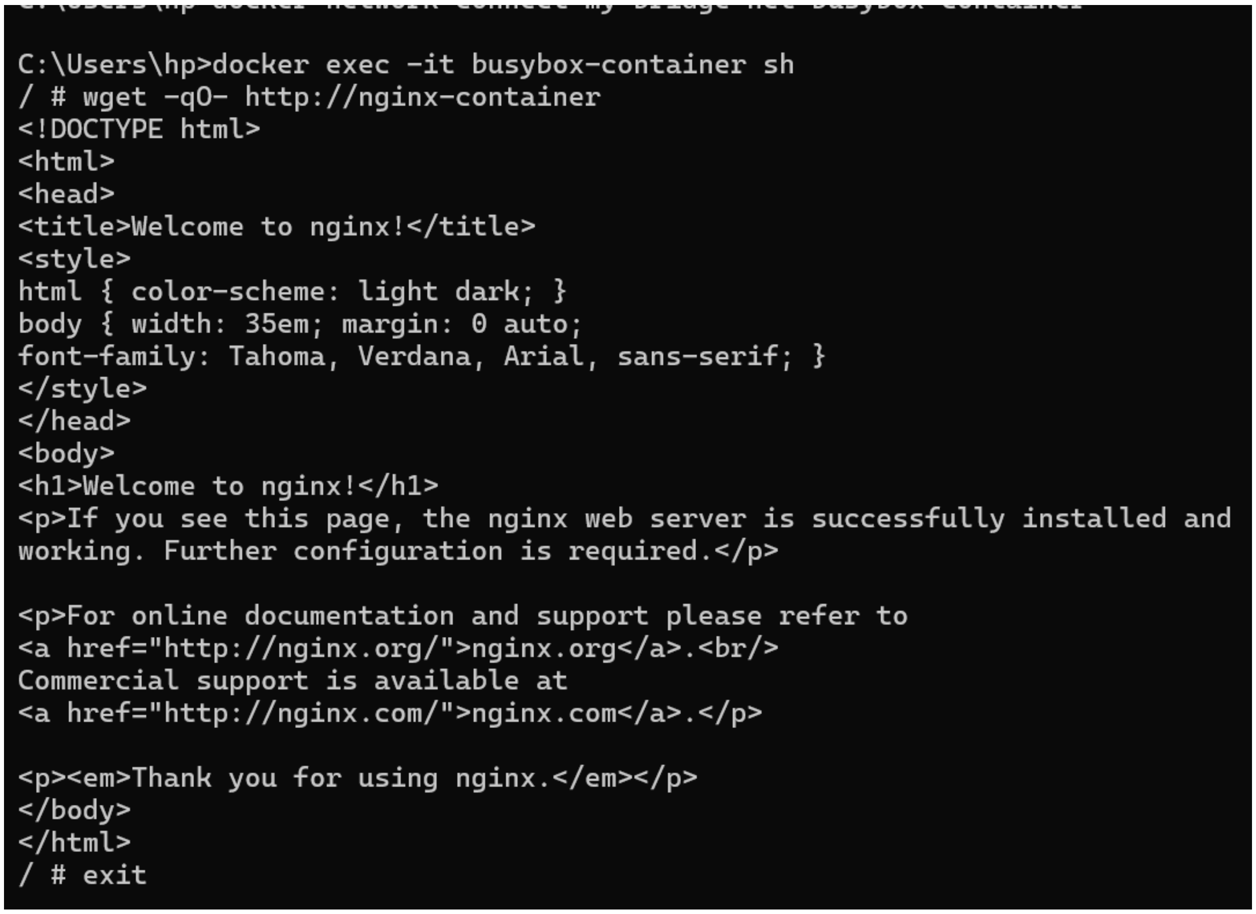
```
C:\Users\hp>docker network create my-bridge-net
495bc80250c7266bbf86354b09e94474144bd3c1fabfc6e46b355d384956985c
```

- docker network connect my-bridge-net nginx-container
- docker network connect my-bridge-net busybox-container



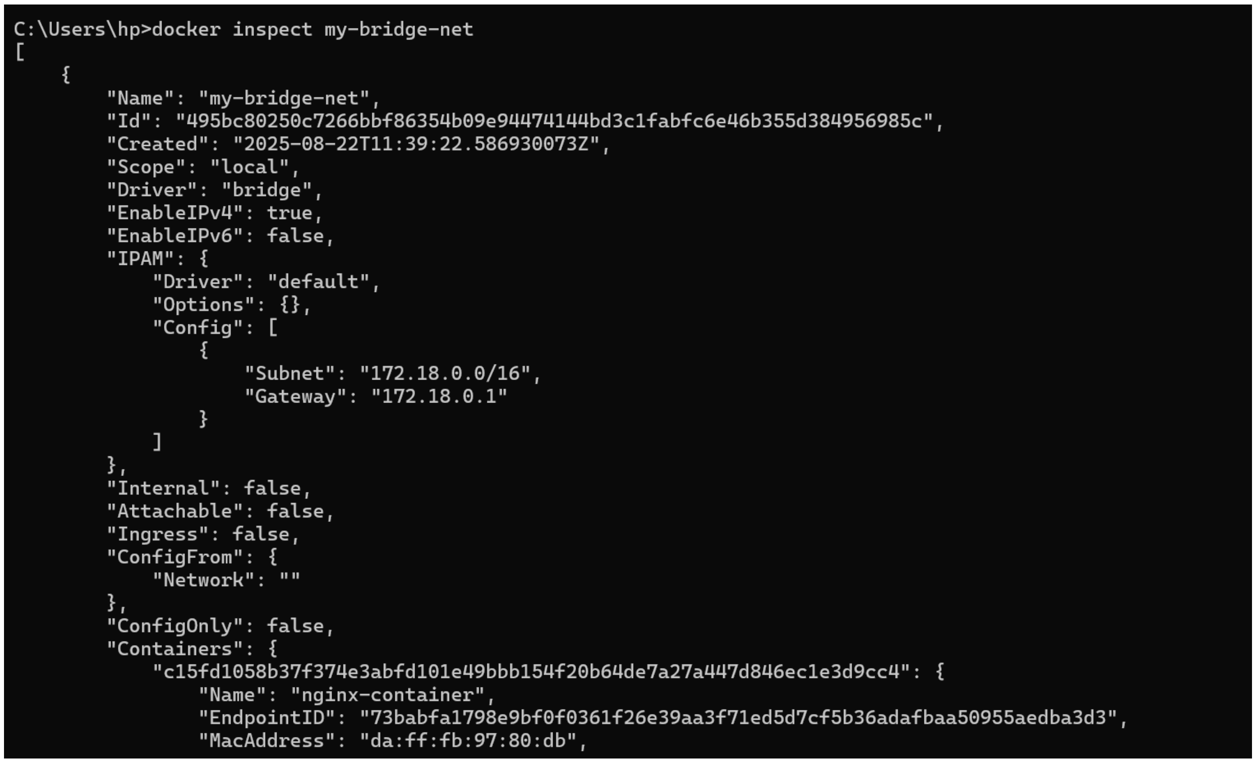
```
C:\Users\hp>docker network connect my-bridge-net nginx-container

C:\Users\hp>docker network connect my-bridge-net busybox-container
```

- docker exec -it busybox-container sh
- wget -qO- http://nginx-container



```
C:\Users\hp>docker exec -it busybox-container sh
/ # wget -qO- http://nginx-container
<!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>
/ # exit
```

- docker inspect my-bridge-net



```
C:\Users\hp>docker inspect my-bridge-net
[
    {
        "Name": "my-bridge-net",
        "Id": "495bc80250c7266bbf86354b09e94474144bd3c1fabfc6e46b355d384956985c",
        "Created": "2025-08-22T11:39:22.586930073Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "c15fd1058b37f374e3abfd101e49bbb154f20b64de7a27a447d846ec1e3d9cc4": {
                "Name": "nginx-container",
                "EndpointID": "73babfa1798e9bf0f0361f26e39aa3f71ed5d7cf5b36adafbaa50955aedba3d3",
                "MacAddress": "da:ff:fb:97:80:db",
```

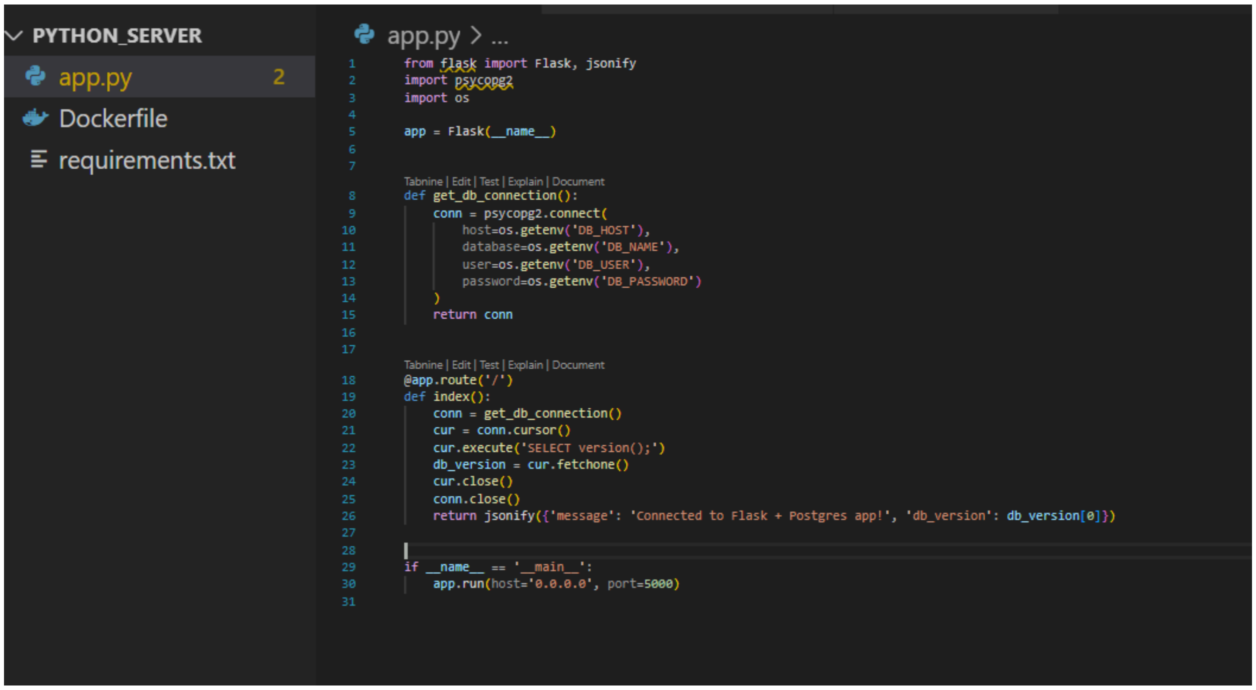
Without a custom network, default network is used for both the containers.
Default bridge network has limited functionality. Containers cannot resolve each other's names, making communication more difficult, only using the IP Addresses assigned to each container.

🔹 Exercise 7: Building a Two-Tier App

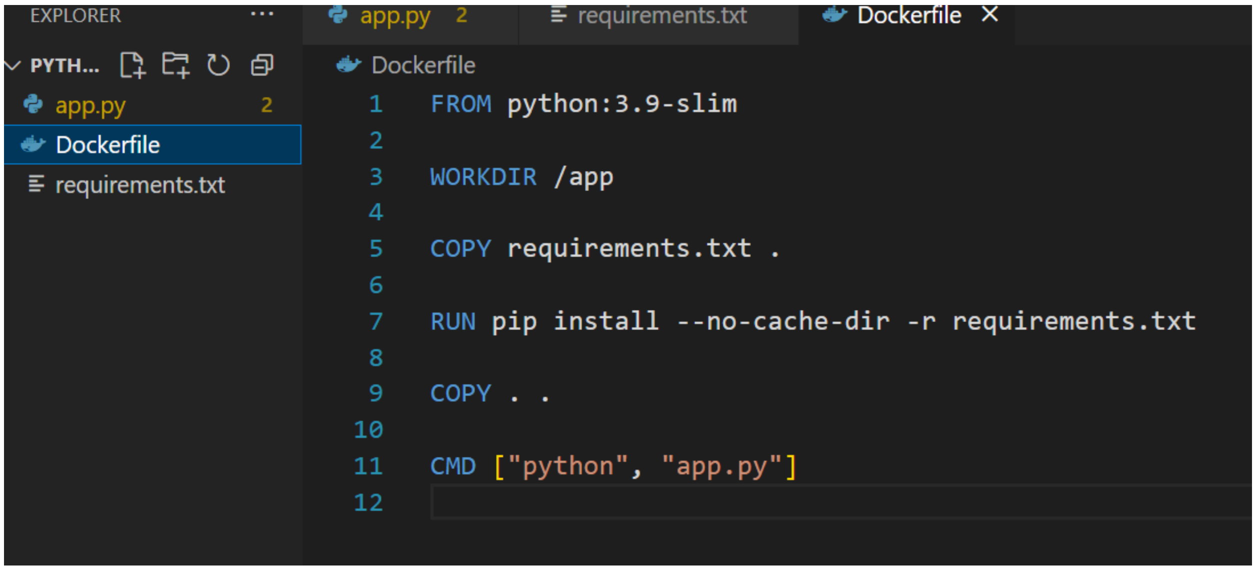
Objective: Deploy a small web + DB stack without Compose.

- Manually run a Python/Flask app container and a Postgres container.
- Use environment variables to configure the connection.
- Verify the app connects to the DB and serves content.

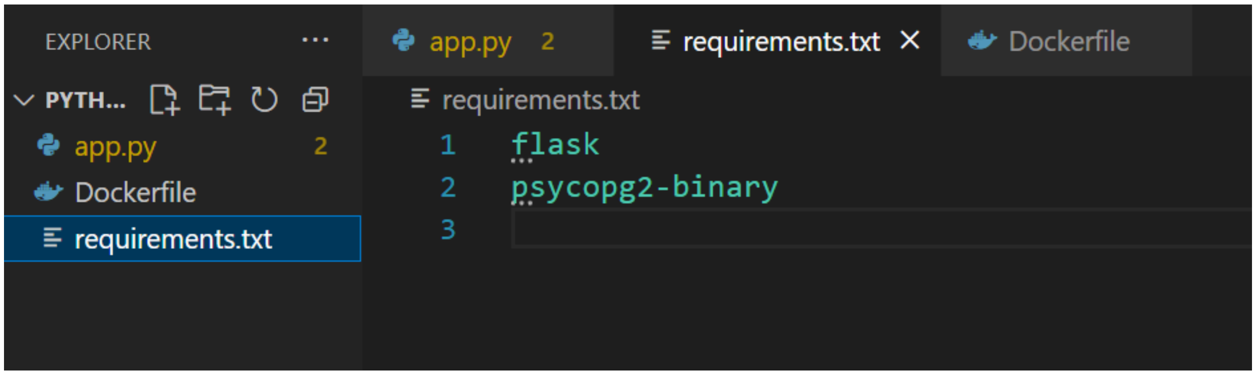
Extension:
Add a volume to persist the DB data.



```python
from flask import Flask, jsonify
import psycopg2
import os

app = Flask(__name__)


def get_db_connection():
    conn = psycopg2.connect(
        host=os.getenv('DB_HOST'),
        database=os.getenv('DB_NAME'),
        user=os.getenv('DB_USER'),
        password=os.getenv('DB_PASSWORD')
    )
    return conn


@app.route('/')
def index():
    conn = get_db_connection()
    cur = conn.cursor()
    cur.execute('SELECT version();')
    db_version = cur.fetchone()
    cur.close()
    conn.close()
    return jsonify({'message': 'Connected to Flask + Postgres app!', 'db_version': db_version[0]})


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

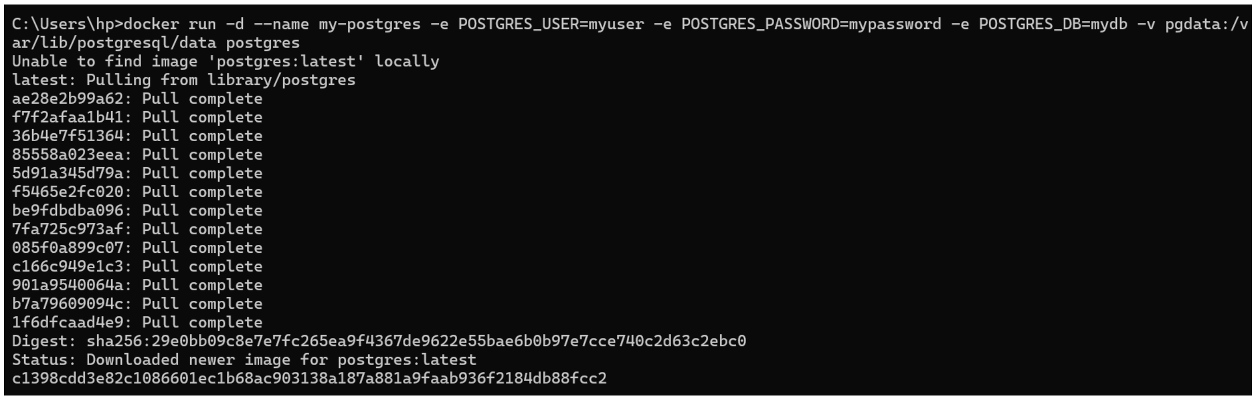


```dockerfile
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .

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

COPY . .

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



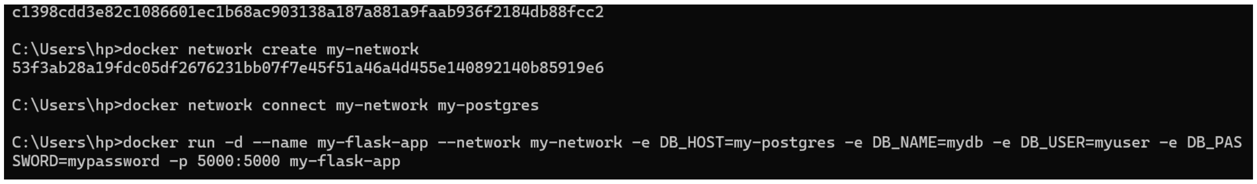
```
flask
psycopg2-binary
```

- docker run -d --name my-postgres -e POSTGRES_USER=myuser -e POSTGRES_PASSWORD=mypassword -e POSTGRES_DB=mydb -v pgdata:/var/lib/postgresql/data postgres

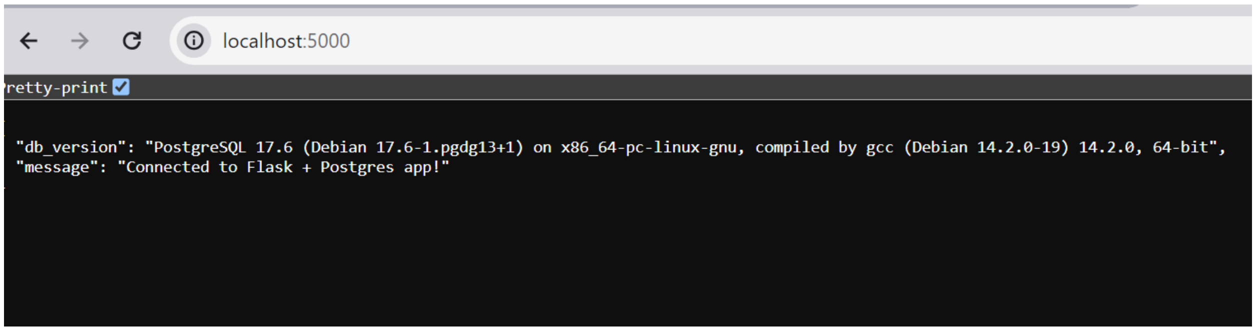


```
C:\Users\hp>docker run -d --name my-postgres -e POSTGRES_USER=myuser -e POSTGRES_PASSWORD=mypassword -e POSTGRES_DB=mydb -v pgdata:/v
ar/lib/postgresql/data postgres
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
ae28e2b99a62: Pull complete
f7f2afaa1b41: Pull complete
36b4e7f51364: Pull complete
85558a023eea: Pull complete
5d91a345d79a: Pull complete
f5465e2fc020: Pull complete
be9fdbdba096: Pull complete
7fa725c973af: Pull complete
085f0a899c07: Pull complete
c166c949e1c3: Pull complete
901a9540064a: Pull complete
b7a79609094c: Pull complete
1f6dfcaad4e9: Pull complete
Digest: sha256:29e0bb09c8e7e7fc265ea9f4367de9622e55bae6b0b97e7cce740c2d63c2ebc0
Status: Downloaded newer image for postgres:latest
c1398cdd3e82c1086601ec1b68ac903138a187a881a9faab936f2184db88fcc2
```

- docker network create my-network
- docker network connect my-network my-postgres
- docker run -d --name my-flask-app --network my-network -e DB_HOST=my-postgres -e DB_NAME=mydb -e DB_USER=myuser -e DB_PASSWORD=mypassword -p 5000:5000 my-flask-app



```
c1398cdd3e82c1086601ec1b68ac903138a187a881a9faab936f2184db88fcc2

C:\Users\hp>docker network create my-network
53f3ab28a19fdc05df2676231bb07f7e45f51a46a4d455e140892140b85919e6

C:\Users\hp>docker network connect my-network my-postgres

C:\Users\hp>docker run -d --name my-flask-app --network my-network -e DB_HOST=my-postgres -e DB_NAME=mydb -e DB_USER=myuser -e DB_PAS
SWORD=mypassword -p 5000:5000 my-flask-app
```

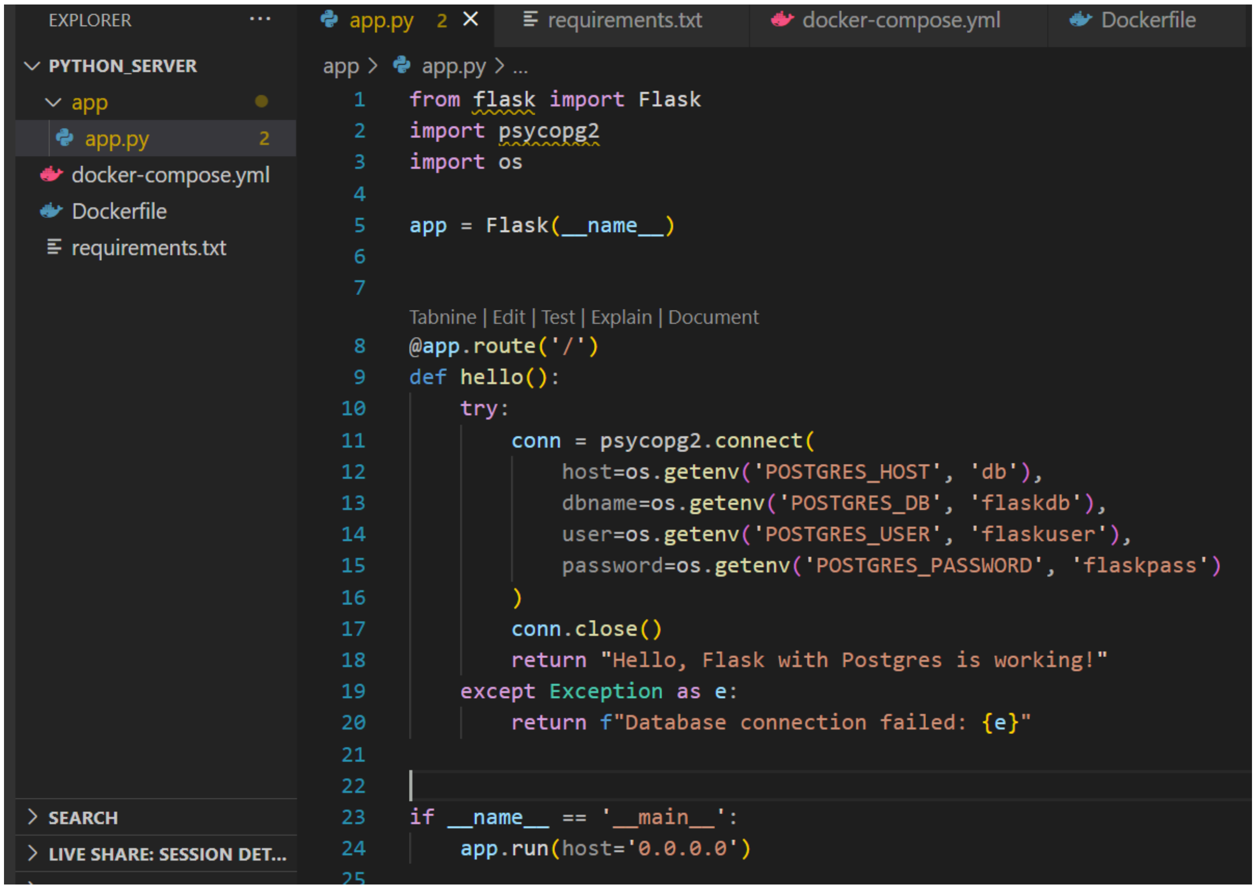


```
"db_version": "PostgreSQL 17.6 (Debian 17.6-1.pgdg13+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 14.2.0-19) 14.2.0, 64-bit",
"message": "Connected to Flask + Postgres app!"
```

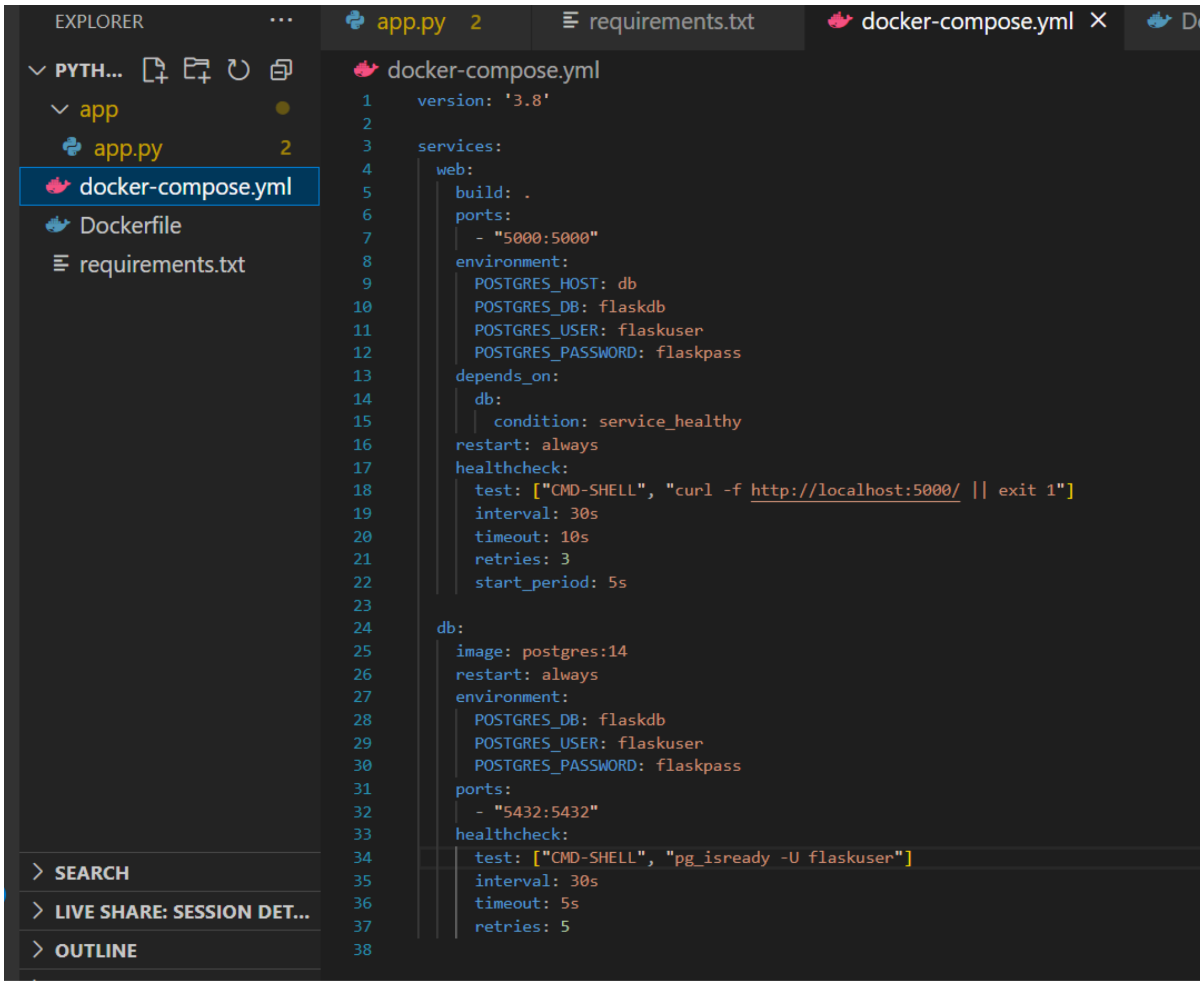
🔹 Exercise 8: Docker Compose Basics

Objective: Use Compose to simplify multi-container apps.

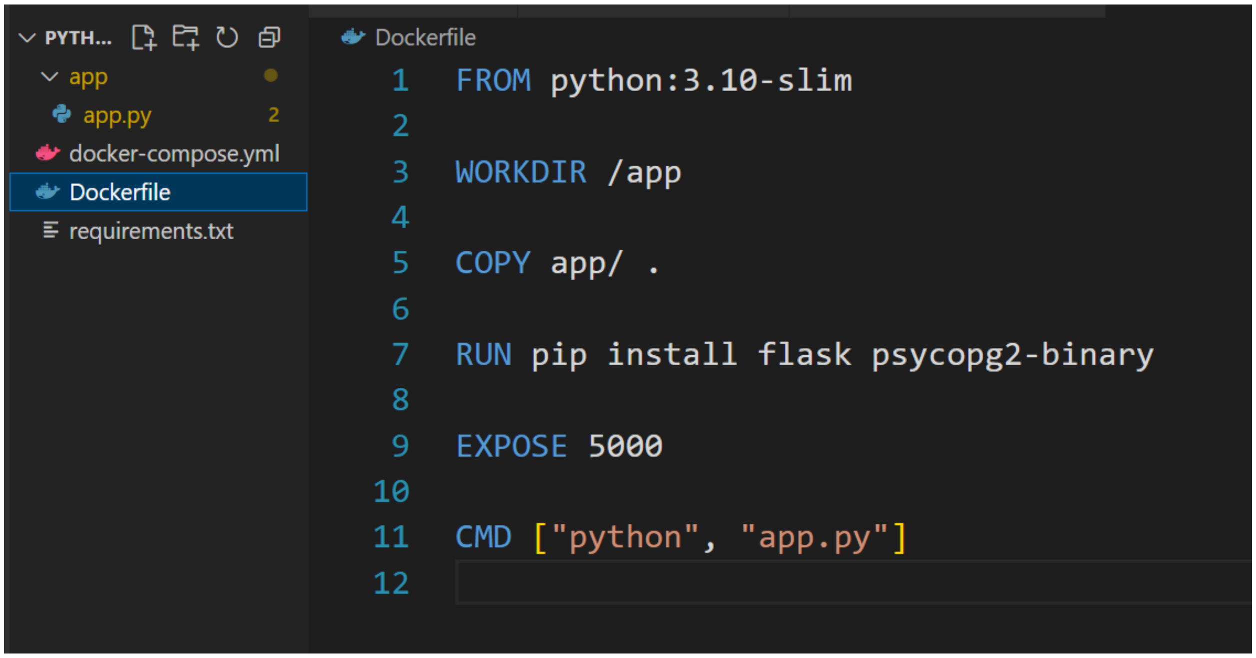
- Write a `docker-compose.yml` for FastAPI + Postgres.
- Use `docker compose up`, inspect logs and containers.
- Add health checks and environment variables.
- Use `depends_on`, restart policies.



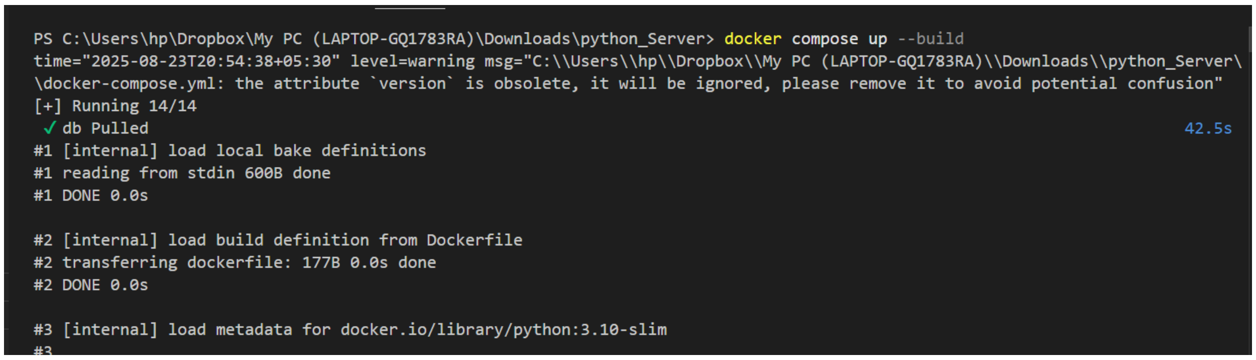
```python
from flask import Flask
import psycopg2
import os

app = Flask(__name__)


@app.route('/')
def hello():
    try:
        conn = psycopg2.connect(
            host=os.getenv('POSTGRES_HOST', 'db'),
            dbname=os.getenv('POSTGRES_DB', 'flaskdb'),
            user=os.getenv('POSTGRES_USER', 'flaskuser'),
            password=os.getenv('POSTGRES_PASSWORD', 'flaskpass')
        )
        conn.close()
        return "Hello, Flask with Postgres is working!"
    except Exception as e:
        return f"Database connection failed: {e}"


if __name__ == '__main__':
    app.run(host='0.0.0.0')
```



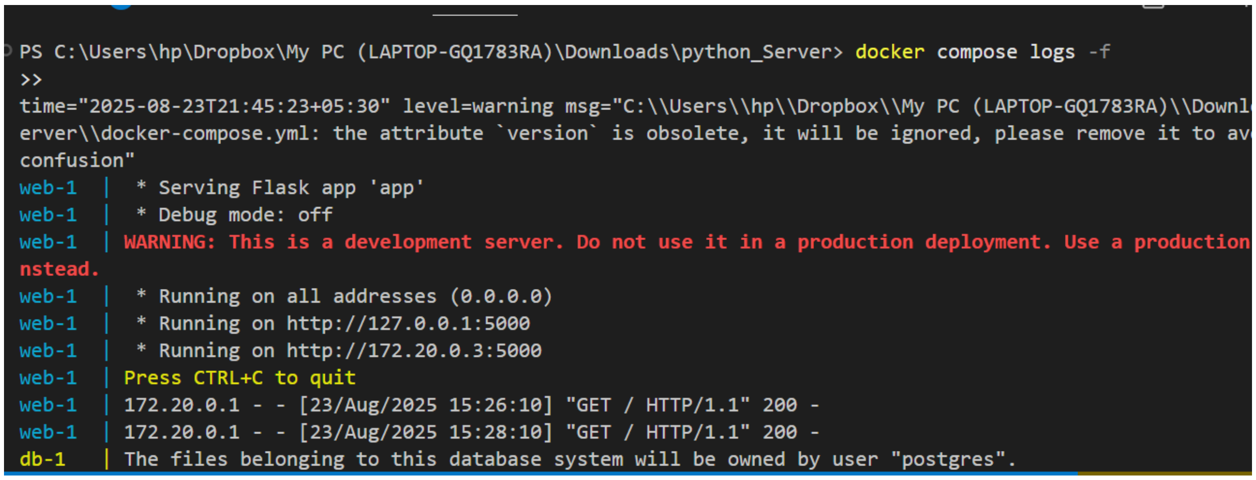
```yaml
version: '3.8'

services:
  web:
    build: .
    ports:
      - "5000:5000"
    environment:
      POSTGRES_HOST: db
      POSTGRES_DB: flaskdb
      POSTGRES_USER: flaskuser
      POSTGRES_PASSWORD: flaskpass
    depends_on:
      db:
        condition: service_healthy
    restart: always
    healthcheck:
      test: ["CMD-SHELL", "curl -f http://localhost:5000/ || exit 1"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 5s

  db:
    image: postgres:14
    restart: always
    environment:
      POSTGRES_DB: flaskdb
      POSTGRES_USER: flaskuser
      POSTGRES_PASSWORD: flaskpass
    ports:
      - "5432:5432"
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U flaskuser"]
      interval: 30s
      timeout: 5s
      retries: 5
```



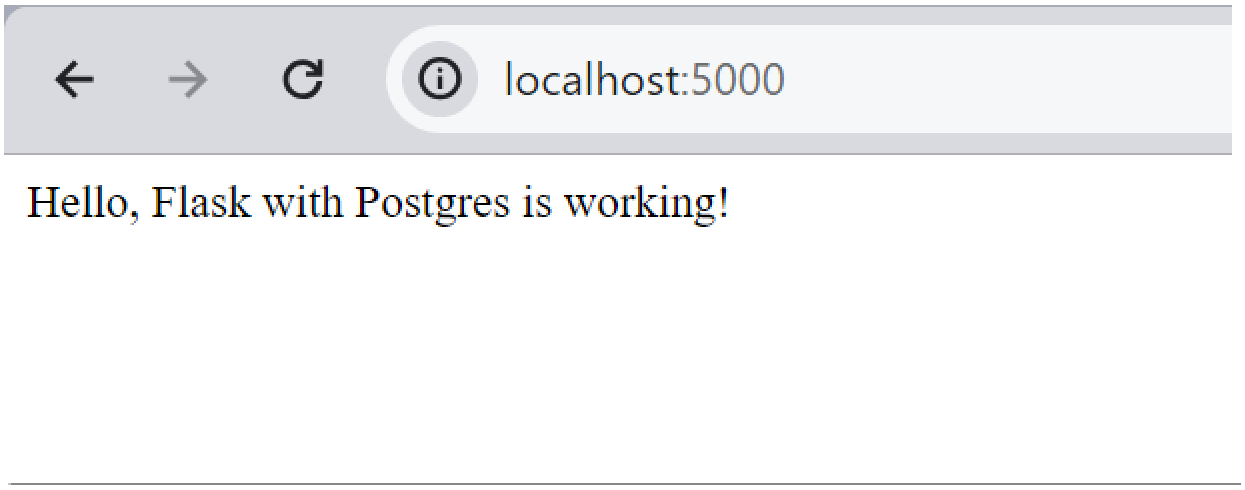
```dockerfile
FROM python:3.10-slim

WORKDIR /app

COPY app/ .

RUN pip install flask psycopg2-binary

EXPOSE 5000

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



```
PS C:\Users\hp\Dropbox\My PC (LAPTOP-GQ1783RA)\Downloads\python_Server> docker compose up --build
time="2025-08-23T20:54:38+05:30" level=warning msg="C:\\Users\\hp\\Dropbox\\My PC (LAPTOP-GQ1783RA)\\Downloads\\python_Server\
\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 14/14
 ✓ db Pulled                                                                                                              42.5s
#1 [internal] load local bake definitions
#1 reading from stdin 600B done
#1 DONE 0.0s

#2 [internal] load build definition from Dockerfile
#2 transferring dockerfile: 177B 0.0s done
#2 DONE 0.0s

#3 [internal] load metadata for docker.io/library/python:3.10-slim
#3
```



```
PS C:\Users\hp\Dropbox\My PC (LAPTOP-GQ1783RA)\Downloads\python_Server> docker compose logs -f
>>
time="2025-08-23T21:45:23+05:30" level=warning msg="C:\\Users\\hp\\Dropbox\\My PC (LAPTOP-GQ1783RA)\\Downl
erver\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to av
confusion"
web-1  |  * Serving Flask app 'app'
web-1  |  * Debug mode: off
web-1  | WARNING: This is a development server. Do not use it in a production deployment. Use a production
nstead.
web-1  |  * Running on all addresses (0.0.0.0)
web-1  |  * Running on http://127.0.0.1:5000
web-1  |  * Running on http://172.20.0.3:5000
web-1  | Press CTRL+C to quit
web-1  | 172.20.0.1 - - [23/Aug/2025 15:26:10] "GET / HTTP/1.1" 200 -
web-1  | 172.20.0.1 - - [23/Aug/2025 15:28:10] "GET / HTTP/1.1" 200 -
db-1   | The files belonging to this database system will be owned by user "postgres".
```



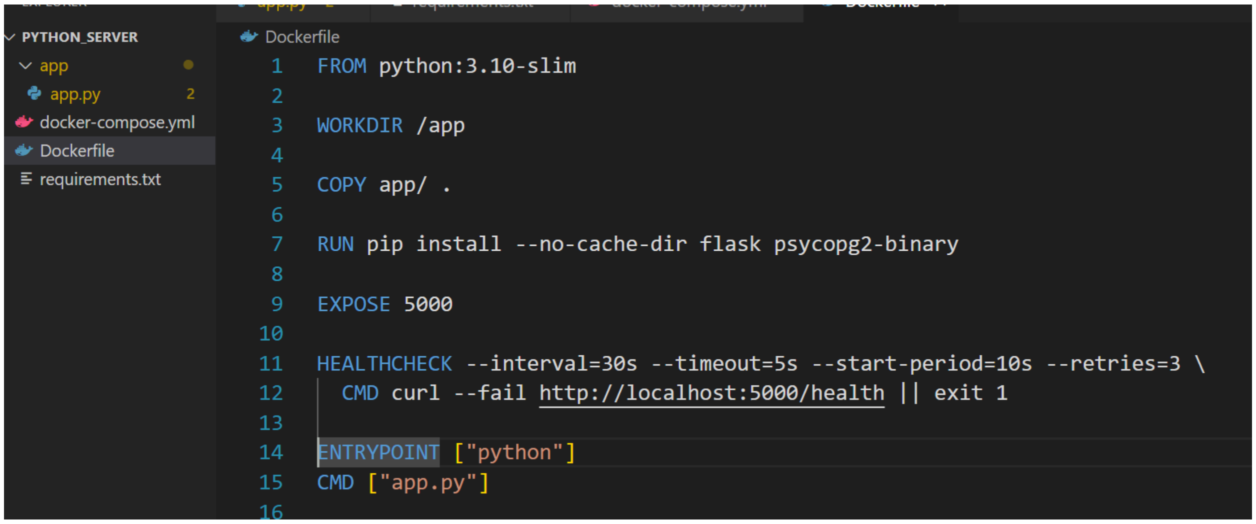
localhost:5000

Hello, Flask with Postgres is working!

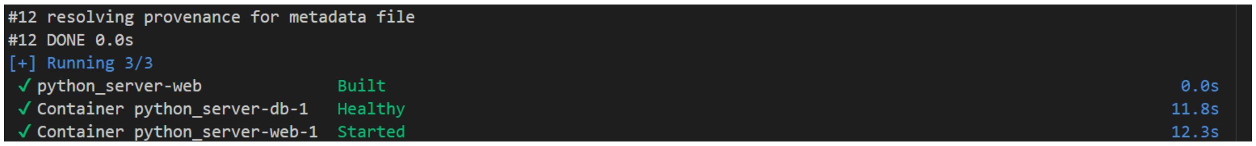
🔹 Exercise 9: Healthchecks and Best Practices

Objective: Make robust, production-like Dockerfiles.

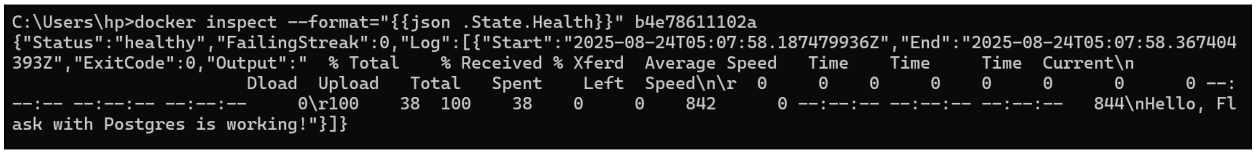
- Add `HEALTHCHECK` instruction to your Dockerfile.
- Use `ENTRYPOINT` vs `CMD` appropriately.
- Minimize layers and image size (e.g., using `alpine`).
- Inspect container health via `docker inspect`.



```dockerfile
FROM python:3.10-slim

WORKDIR /app

COPY app/ .

RUN pip install --no-cache-dir flask psycopg2-binary

EXPOSE 5000

HEALTHCHECK --interval=30s --timeout=5s --start-period=10s --retries=3 \
  CMD curl --fail http://localhost:5000/health || exit 1

ENTRYPOINT ["python"]
CMD ["app.py"]

```



```
#12 resolving provenance for metadata file
#12 DONE 0.0s
[+] Running 3/3
 ✓ python_server-web             Built                                                    0.0s
 ✓ Container python_server-db-1  Healthy                                                 11.8s
 ✓ Container python_server-web-1 Started                                                 12.3s
```

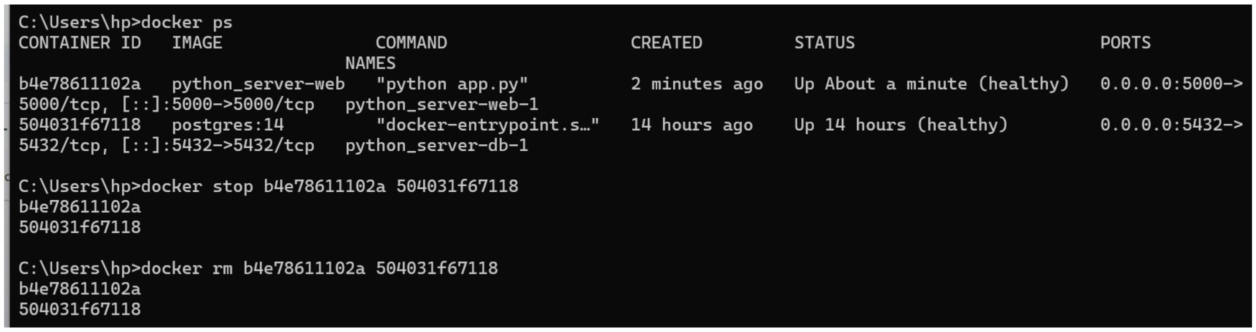


```
C:\Users\hp>docker inspect --format="{{json .State.Health}}" b4e78611102a
{"Status":"healthy","FailingStreak":0,"Log":[{"Start":"2025-08-24T05:07:58.187479936Z","End":"2025-08-24T05:07:58.367404
393Z","ExitCode":0,"Output":"  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current\n
                       Dload  Upload   Total   Spent    Left  Speed\n\r  0     0    0     0    0     0      0      0 --:
--:-- --:--:-- --:--:--     0\r100    38  100    38    0     0    842      0 --:--:-- --:--:-- --:--:--   844\nHello, Fl
ask with Postgres is working!"}]}
```

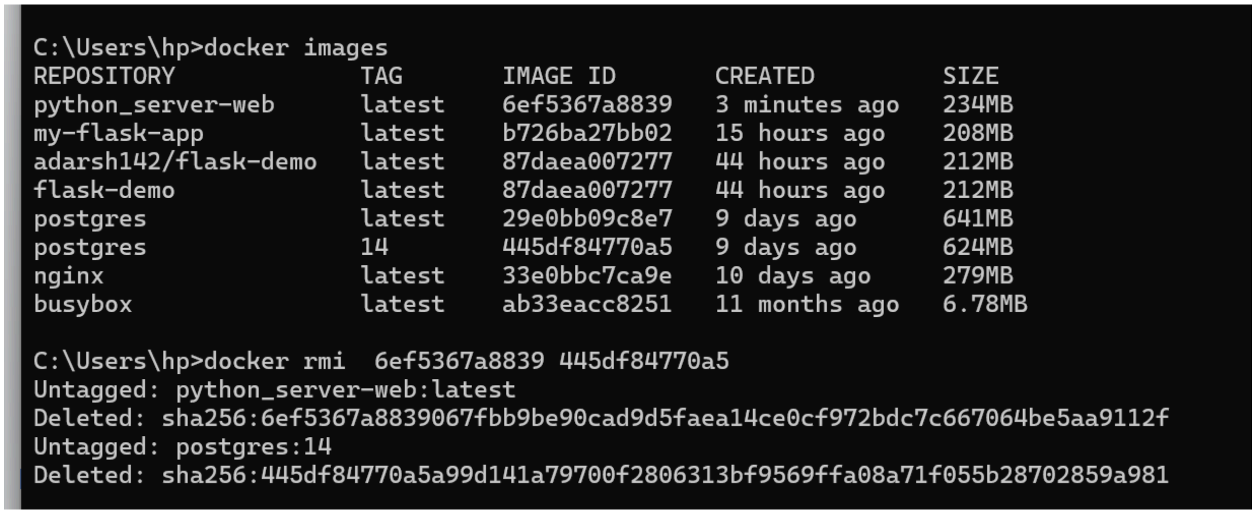
🔹 Exercise 10: Debugging, Cleanup & Troubleshooting

Objective: Learn to manage resources and solve issues.

- Run containers with bad commands or missing ports.
- Clean up unused images, containers, volumes with:



```
C:\Users\hp>docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED         STATUS                        PORTS
                                NAMES
b4e78611102a   python_server-web   "python app.py"          2 minutes ago   Up About a minute (healthy)   0.0.0.0:5000->
5000/tcp, [::]:5000->5000/tcp   python_server-web-1
504031f67118   postgres:14         "docker-entrypoint.s…"   14 hours ago    Up 14 hours (healthy)         0.0.0.0:5432->
5432/tcp, [::]:5432->5432/tcp   python_server-db-1

C:\Users\hp>docker stop b4e78611102a 504031f67118
b4e78611102a
504031f67118

C:\Users\hp>docker rm b4e78611102a 504031f67118
b4e78611102a
504031f67118
```



```
C:\Users\hp>docker images
REPOSITORY              TAG       IMAGE ID       CREATED          SIZE
python_server-web       latest    6ef5367a8839   3 minutes ago    234MB
my-flask-app            latest    b726ba27bb02   15 hours ago     208MB
adarsh142/flask-demo    latest    87daea007277   44 hours ago     212MB
flask-demo              latest    87daea007277   44 hours ago     212MB
postgres                latest    29e0bb09c8e7   9 days ago       641MB
postgres                14        445df84770a5   9 days ago       624MB
nginx                   latest    33e0bbc7ca9e   10 days ago      279MB
busybox                 latest    ab33eacc8251   11 months ago    6.78MB

C:\Users\hp>docker rmi  6ef5367a8839 445df84770a5
Untagged: python_server-web:latest
Deleted: sha256:6ef5367a8839067fbb9be90cad9d5faea14ce0cf972bdc7c667064be5aa9112f
Untagged: postgres:14
Deleted: sha256:445df84770a5a99d141a79700f2806313bf9569ffa08a71f055b28702859a981
```