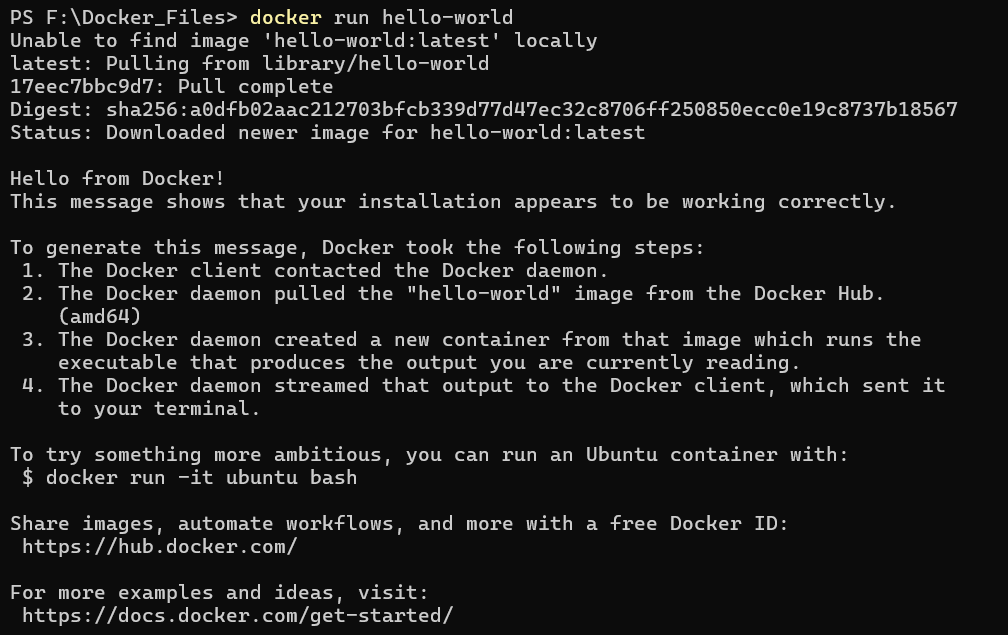
**Exercise 1: Getting Started with Containers**

**Objective:** Run and inspect containers.

**Commands & Explanation:**

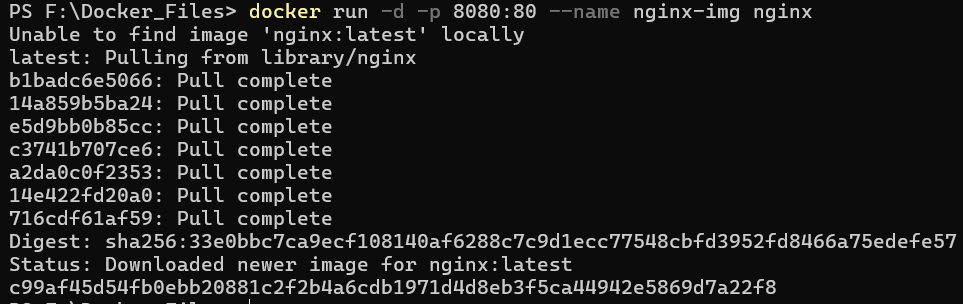
Run hello-world container

* docker run hello-world



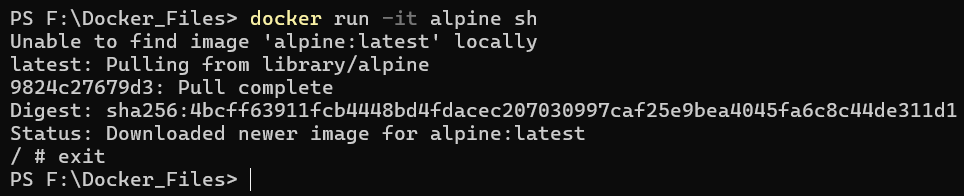
Run nginx container

* docker run -d -p 8080:80 --name nginx-img nginx



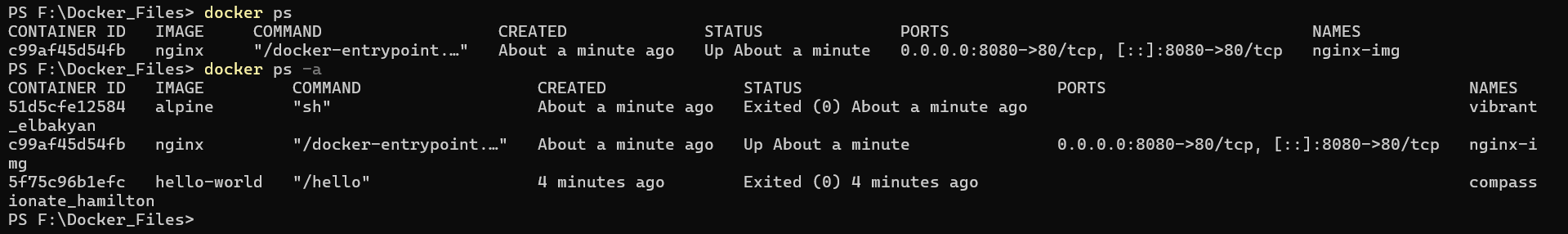
Run alpine container

* docker run -it alpine sh



List all running containers & List all containers including stopped

* docker ps
* docker ps -a



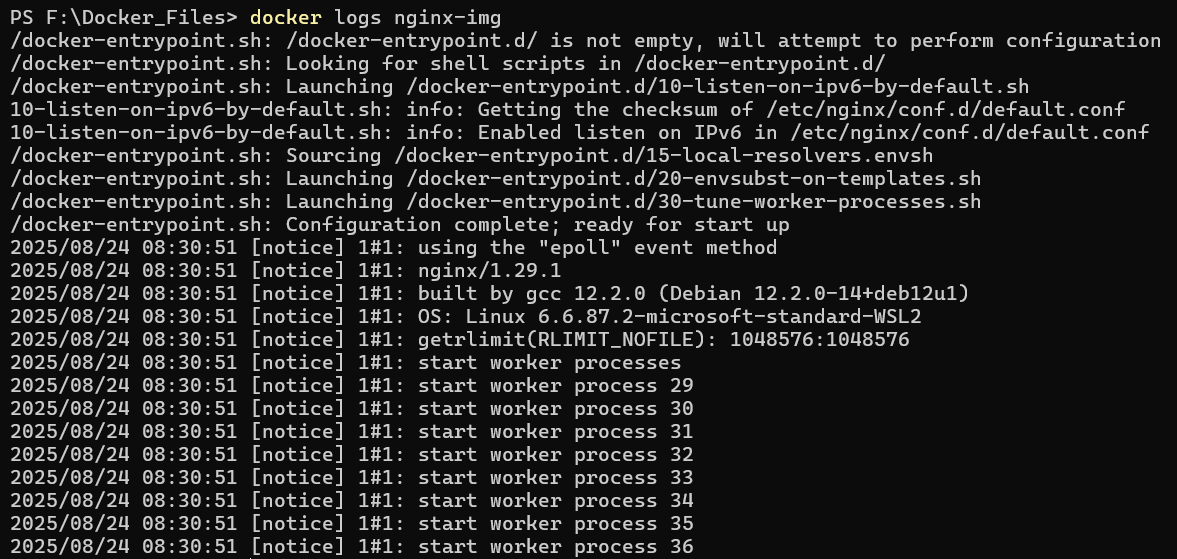
Execute command inside a running container

* docker exec -it nginx-img sh



View logs of nginx container

* docker logs nginx-img



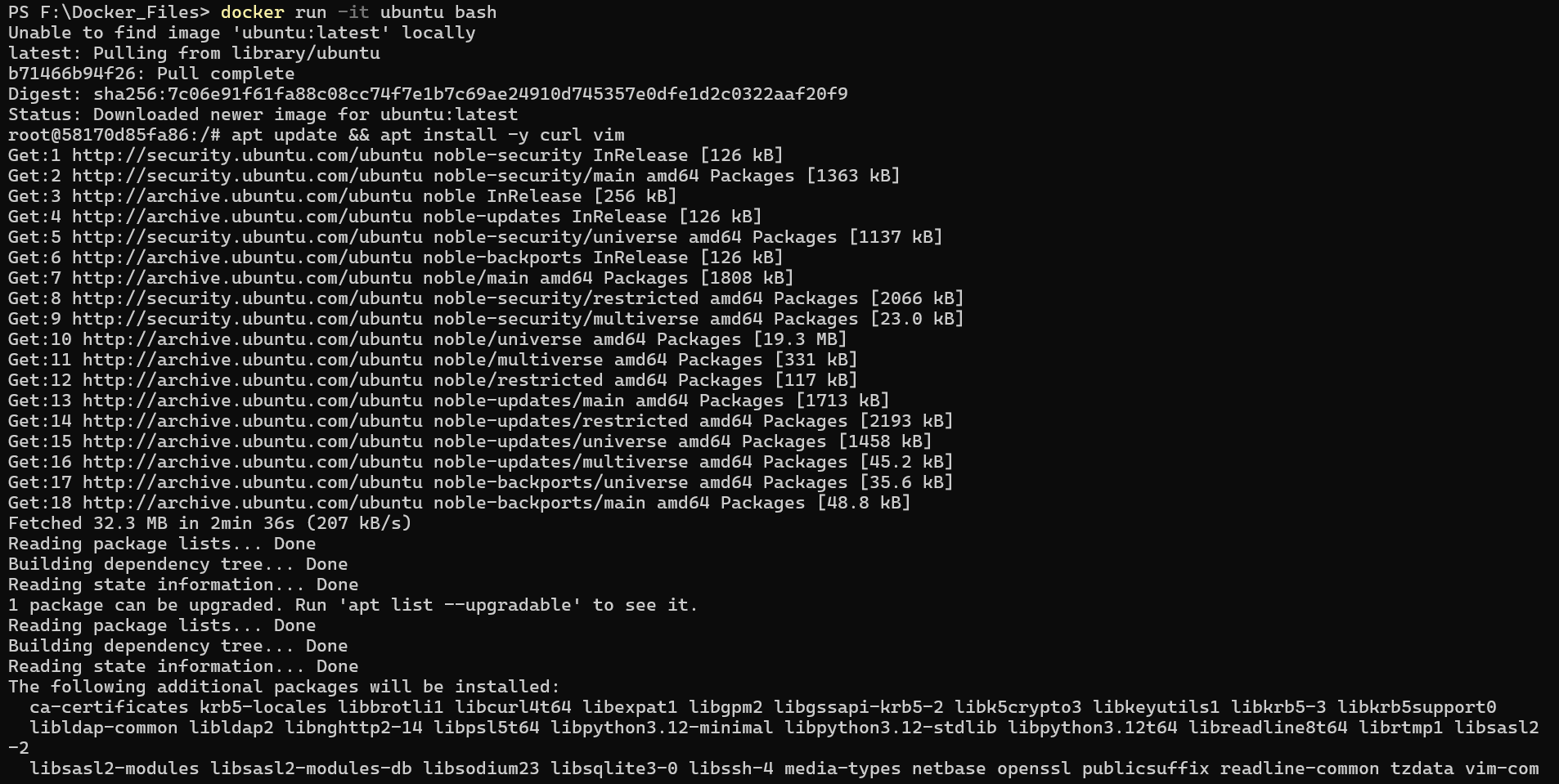
**Checkpoint answers:**

* If a container doesn’t run in detached mode (-d), it occupies the terminal until stopped.
* If ports aren’t mapped (-p host:container), the service inside the container cannot be accessed from the host machine.
* **Exercise 2: Working with Container State**

**Objective:** Modify containers and commit custom images.

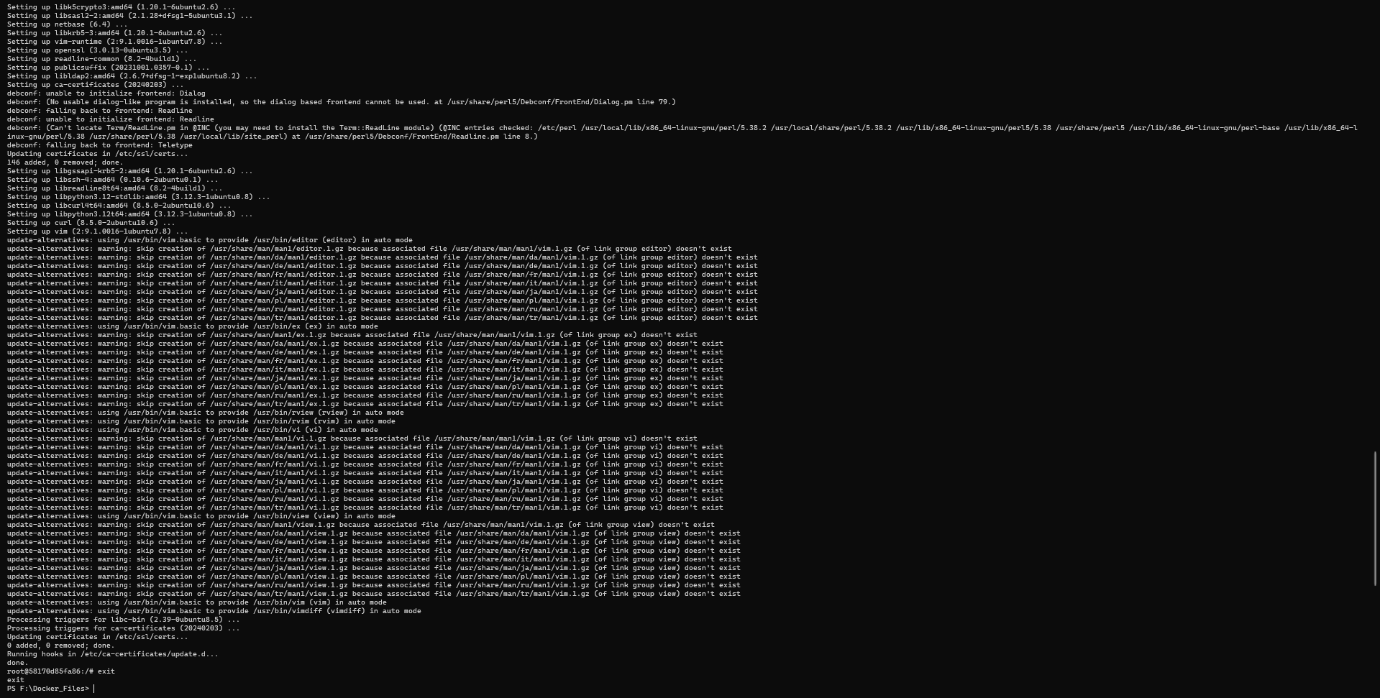
Run Ubuntu container interactively

* docker run -it ubuntu bash



Inside container, install curl and vim & Exit container

* apt update && apt install -y curl vim
* exit



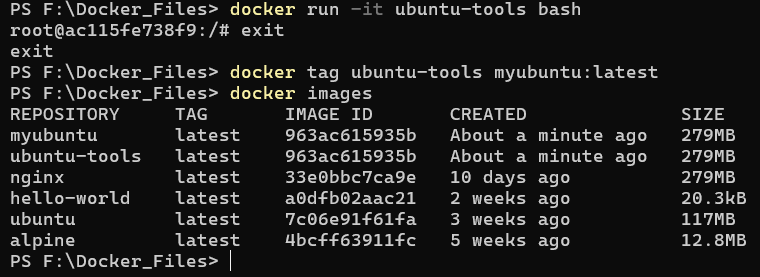
Commit container as new image

* docker commit nginx-img ubuntu-tools



Run container from committed image, Tag and List image

* docker run -it ubuntu-tools bash
* docker tag ubuntu-tools myubuntu:latest
* docker images



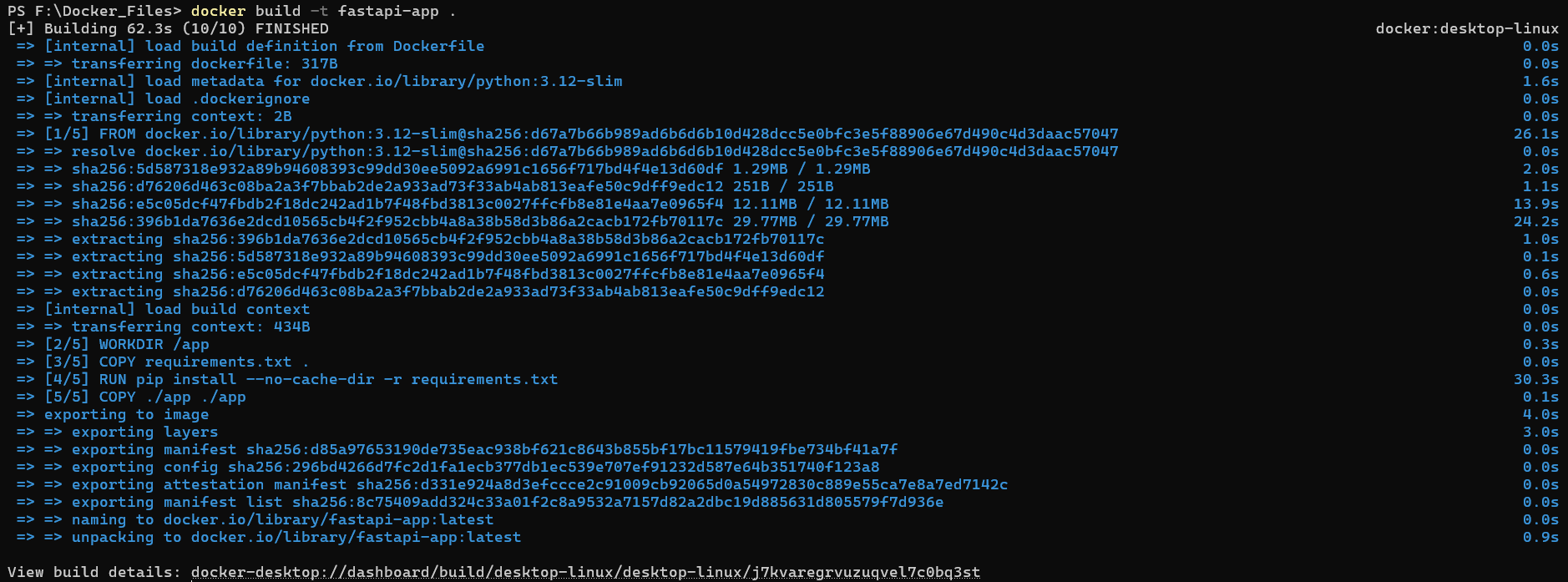
**Exercise 3: Build Custom Images Using Dockerfile**

**Objective:** Build a FastAPI web server Docker image.

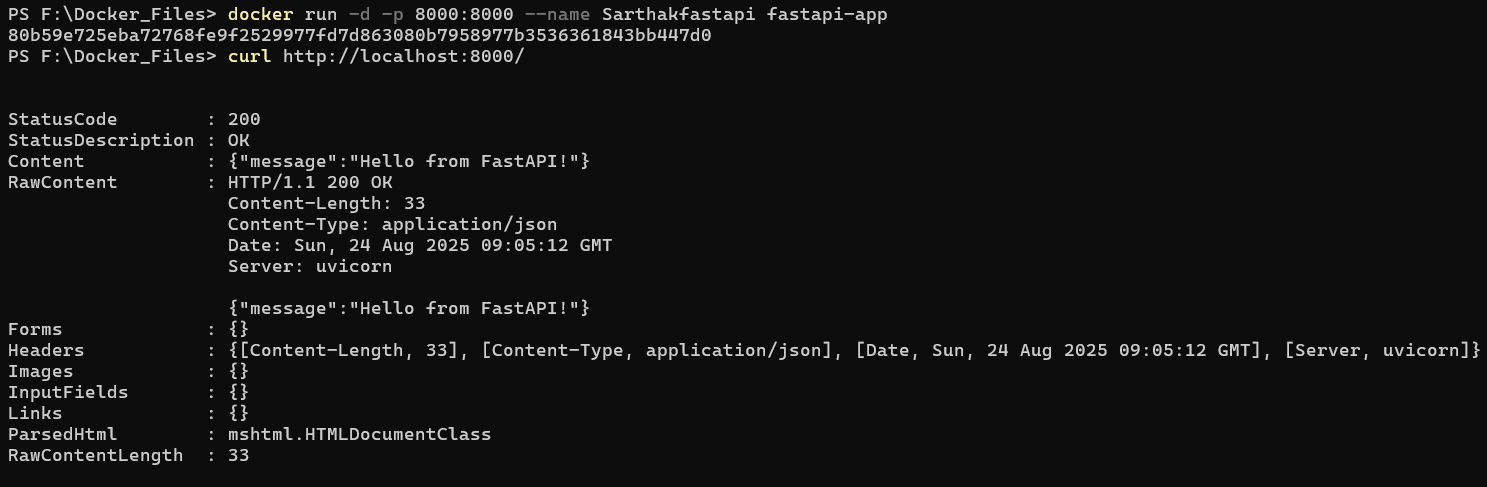
* FastAPI app file and Created Dockerfile for FastAPI

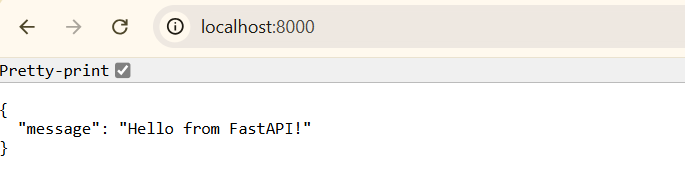
**Build & run commands:**

* docker build -t fastapi-app .



* docker run -d -p 8000:8000 --name Sarthakfastapi fastapi-app and curl http://localhost:8000/

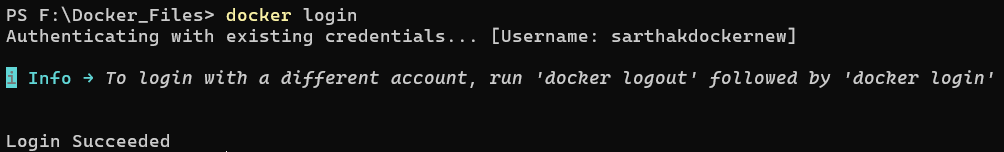




**Exercise 4: Sharing Images**

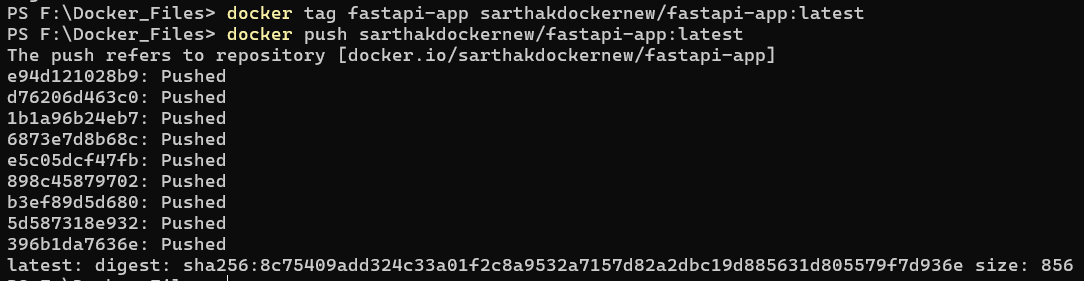
Docker Hub login

* docker login



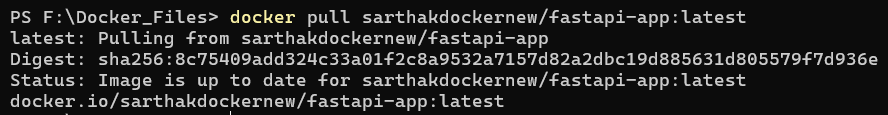
Tag image for Docker Hub and Push

* docker tag fastapi-app sarthakdockernew /fastapi-app:latest
* docker push sarthakdockernew fastapi-app:latest



Pull image

* docker pull sarthakdockernew/fastapi-app:latest

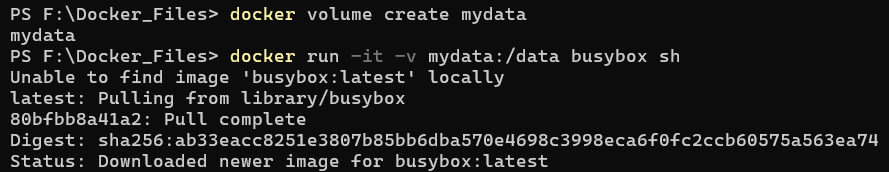


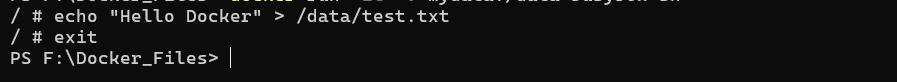
**Reflection:**  
Image tagging allows version control. A common strategy is <username>/<appname>:<version> like sarthakdockernew/fastapi-app:v1.

**Exercise 5: Data Persistence with Volumes**

Named volume

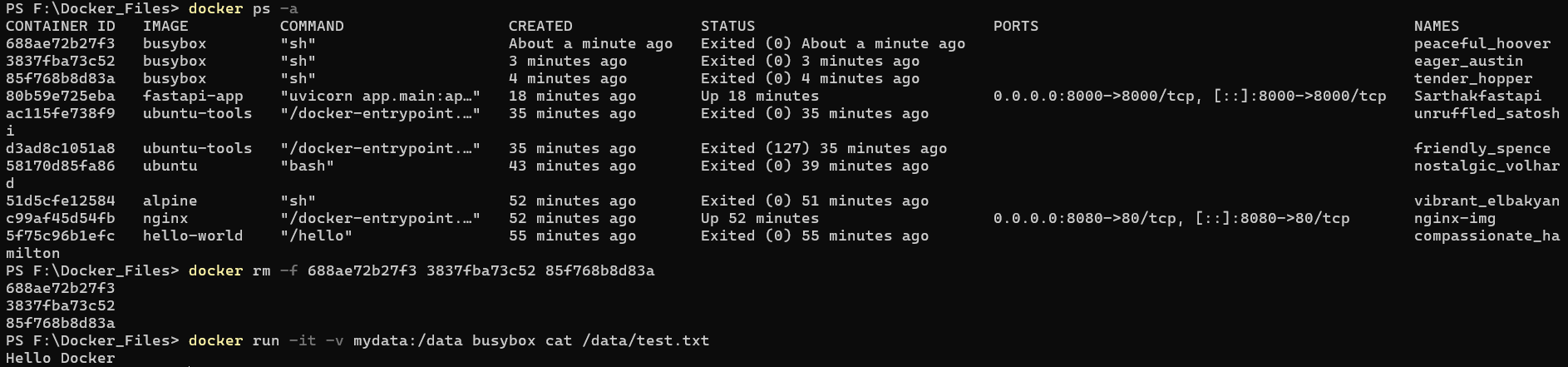
* docker volume create mydata
* docker run -it -v mydata:/data busybox sh
* echo "Hello Docker" > /data/test.txt





Stop & remove container & Relaunch to verify persistence

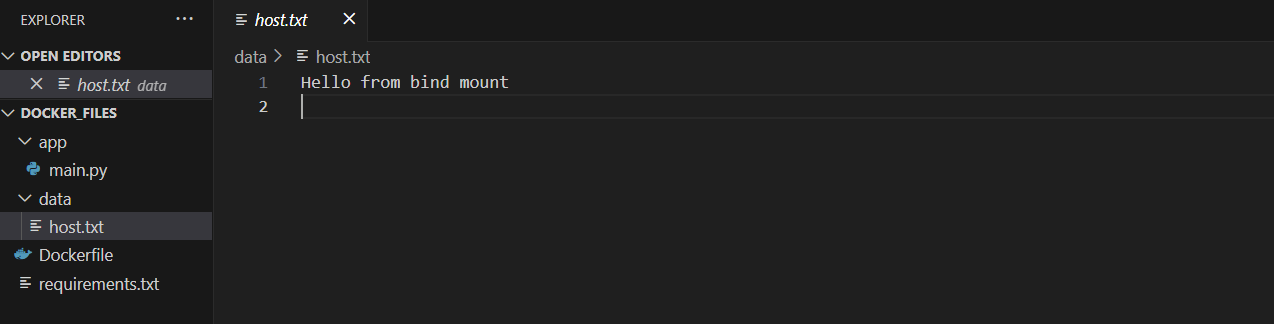
* docker rm -f all three running container after checking by command docker ps -a
* docker run -it -v mydata:/data busybox cat /data/test.txt



Bind mount

* docker run -it -v F:/Docker\_Files/data:/data busybox sh





**Answer:**

* Named volumes are managed by Docker and are good for persistent container data that isn’t tied to host paths.
* Bind mounts map host directories to containers, useful for development where live file changes are needed.

**Exercise 6: Container Networking Basics**

Create network

* docker network create my-network

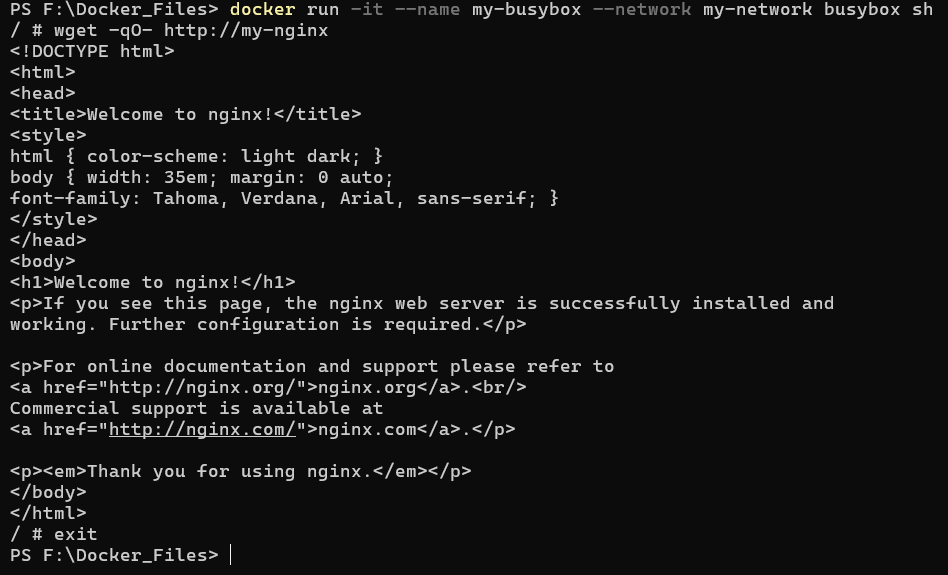


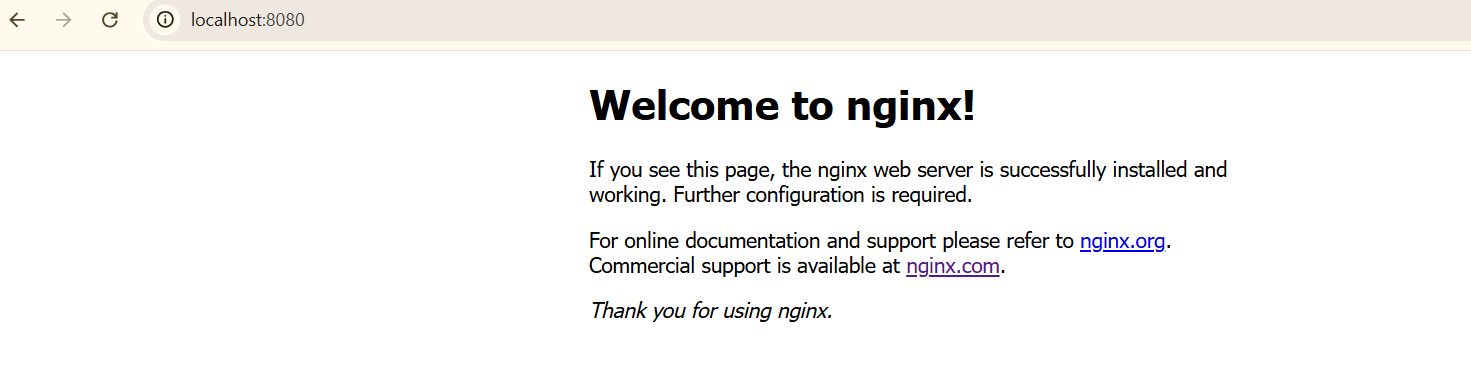
Run containers on network

* docker run -d --name my-nginx --network my-network nginx



* docker run -it --name my-busybox --network my-network busybox sh
* wget -qO- <http://my-nginx>



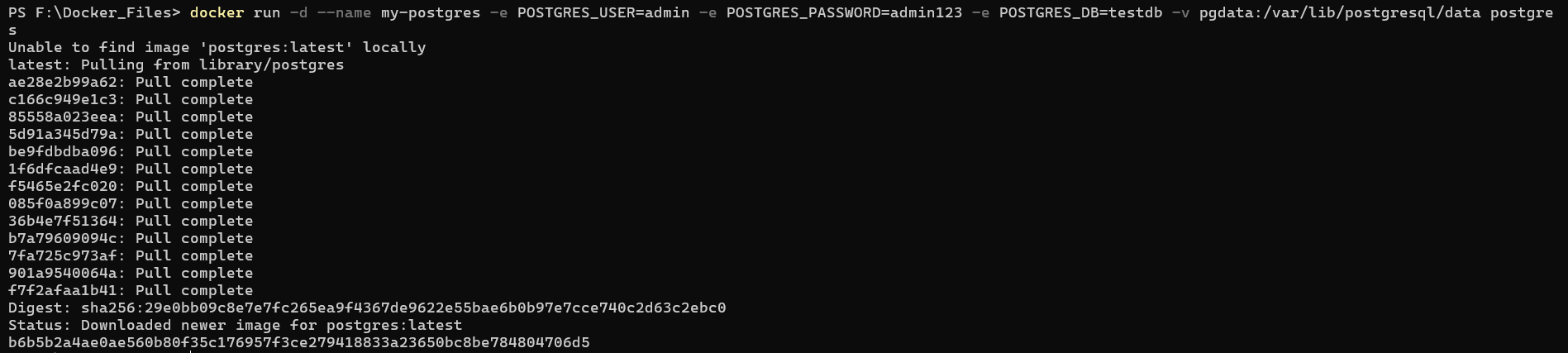


* Without a user-defined network, containers are on default bridge and cannot use container names for communication.

**Exercise 7: Building a Two-Tier App (FastAPI + Postgres)**

Run Postgres

* docker run -d --name my-postgres -e POSTGRES\_USER=admin -e POSTGRES\_PASSWORD=admin123 -e POSTGRES\_DB=testdb -v pgdata:/var/lib/postgresql/data postgres



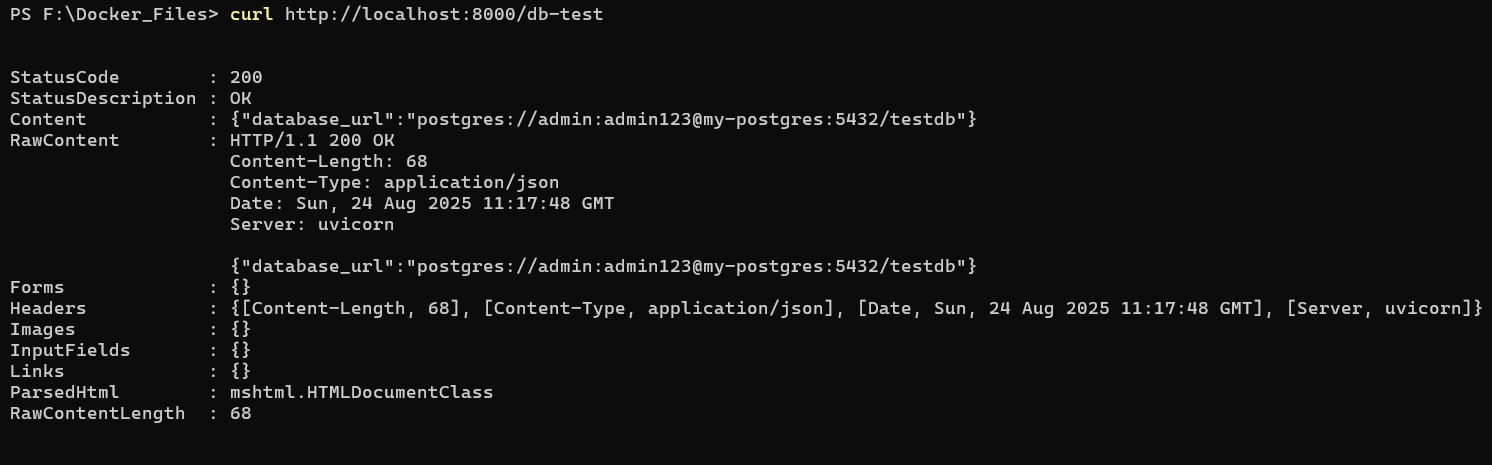
Run FastAPI app with env variable for DB

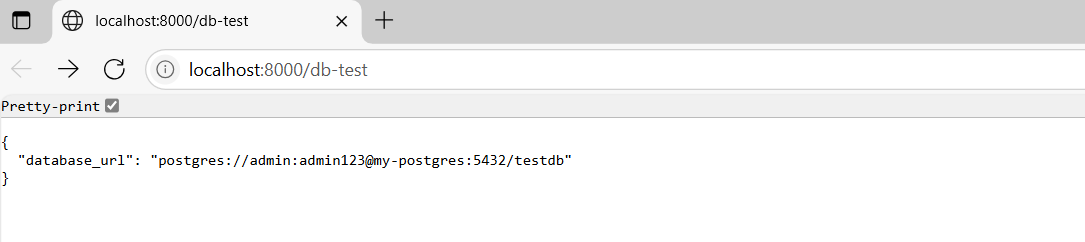
* docker run -d --name sarthakfastapi --network my-network -e POSTGRES\_URL=postgres://admin:admin123@my-postgres:5432/testdb fastapi-app



Test endpoint

* curl http://localhost:8000/db-test



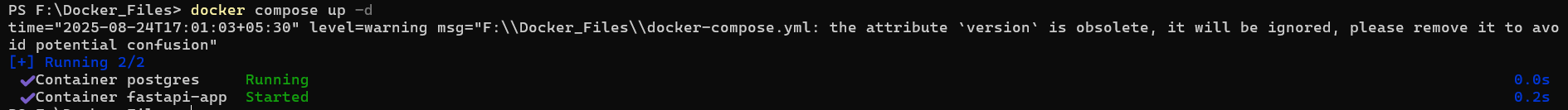


**Exercise 8: Docker Compose Basics**

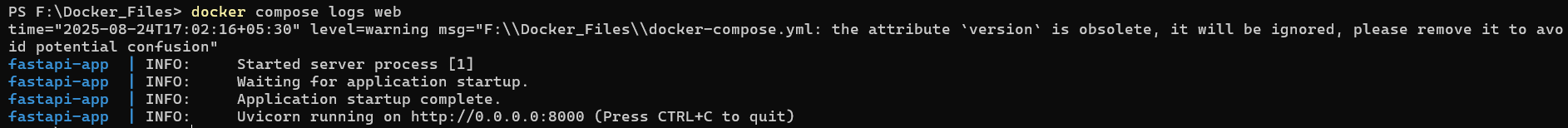
Created docker-compose.yml file

**Commands:**

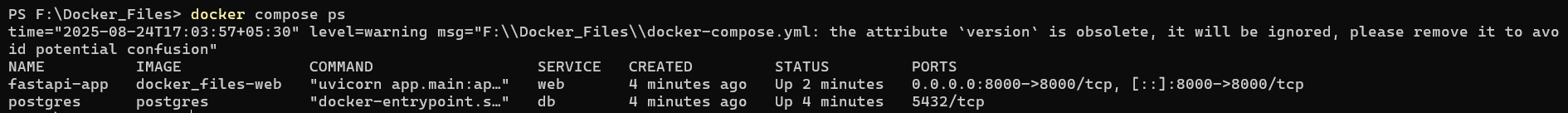
* docker compose up -d

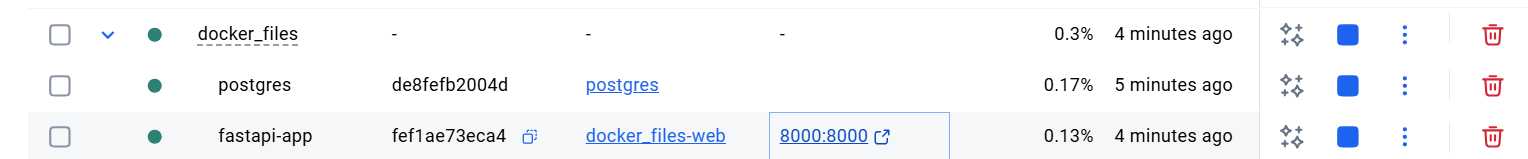


* docker compose logs



* docker compose ps







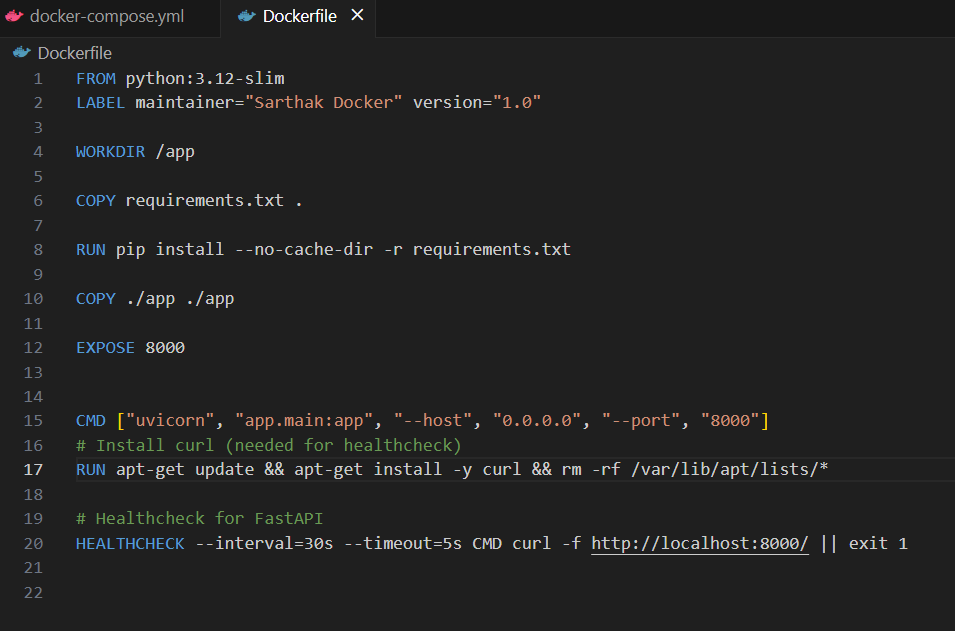
**Exercise 9: Healthchecks and Best Practices**

Updated Dockerfile:

HEALTHCHECK --interval=30s --timeout=5s CMD curl -f http://localhost:8000/ || exit 1

&

RUN apt-get update && apt-get install -y curl && rm -rf /var/lib/apt/lists/\*



* To Test and Verify





**Exercise 10: Debugging, Cleanup & Troubleshooting**

Run container with missing port

* docker run -d fastapi-app



Cleanup unused resources

* docker system prune -af



* docker volume prune -f

