Assignment 1: Your First Docker Commands

What you'll learn: Basic Docker commands - the bread-and-butter stuff

What to do:

1. Download the latest nginx image from Docker Hub
2. Start a nginx container in the background on port 8080
3. Check what containers are running
4. Stop that container
5. Delete the container
6. See what images you have on your machine

Show me: Your command history and some screenshots proving you did it

Answer:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\uday2> docker --version
Docker version 28.3.2, build 578ccf6
PS C:\Users\uday2> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:fb01117203ff38c2f9af91db1a7409459182a37c87cced5cb442d1d8fcc66d19
Status: Image is up to date for nginx:latest
docker.io/library/nginx:latest
PS C:\Users\uday2> docker run -d -p 8080:80 --name mynginx nginx
298d93ae3c142150514575a41c788c859d99e99467a3660b2db2b423c5bc3674
PS C:\Users\uday2> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                                   NAMES
298d93ae3c14   nginx     "/docker-entrypoint.…"   13 seconds ago   Up 12 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   mynginx
PS C:\Users\uday2> docker stop mynginx
mynginx
PS C:\Users\uday2> docker rm mynginx
mynginx
PS C:\Users\uday2> docker images
REPOSITORY                     TAG         IMAGE ID       CREATED        SIZE
flask-app                      latest      d9aff67031a2   4 days ago     200MB
nginx                          latest      fb01117203ff   5 days ago     225MB
academic-credentials           latest      3281f2fabc98   7 days ago     796MB
blockcred-system-frontend      latest      b23f51a50dca   3 months ago   1.22GB
blockcred-system-backend       latest      121bdd3a8517   3 months ago   283MB
block-chain-project-frontend   latest      f2f018469909   3 months ago   2.53GB
block-chain-project-backend    latest      2e91e6ddd167   3 months ago   237MB
postgres                       15          bc51cf4f1fe0   4 months ago   629MB
python                         3.11-slim   1d6131b5d479   4 months ago   186MB
node                           20          572a90df10a5   5 months ago   1.58GB
PS C:\Users\uday2>
```

Explaination:

Opened Docker Desktop and then opened the terminal. Verified that Docker was installed and running. Downloaded the latest nginx image from Docker Hub. Started an nginx container in detached mode and mapped container port 80 to local port 8080. Checked the running containers to confirm nginx was active and verified it in the browser using http://localhost:8080. After verification, stopped the running container and removed it from the system. Finally, listed all Docker images available on the machine.

Commands Used (In Order):

docker --version

docker pull nginx

docker run -d -p 8080:80 --name mynginx nginx

docker ps

docker stop mynginx

docker rm mynginx

docker images

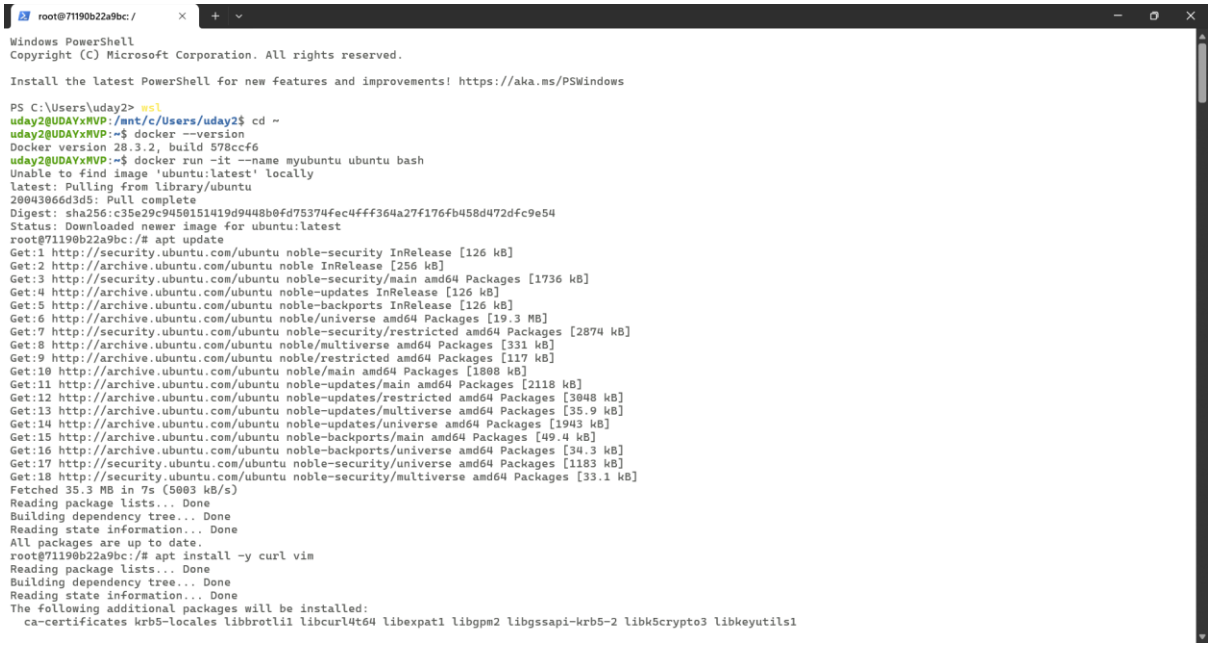
Assignment 2: Jump Inside a Container

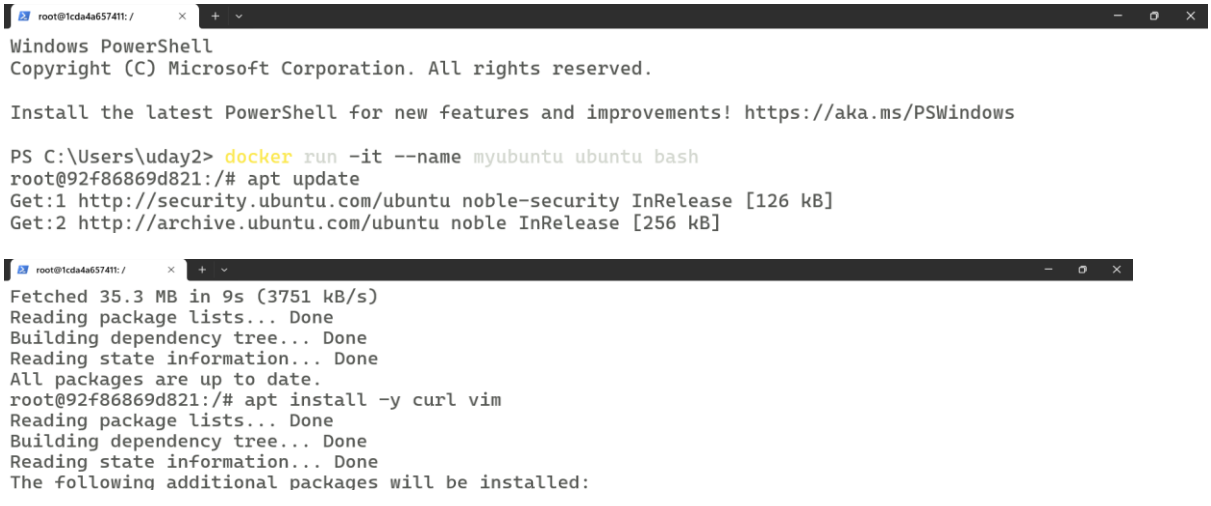
What you'll learn: How to work inside a running container

What to do:

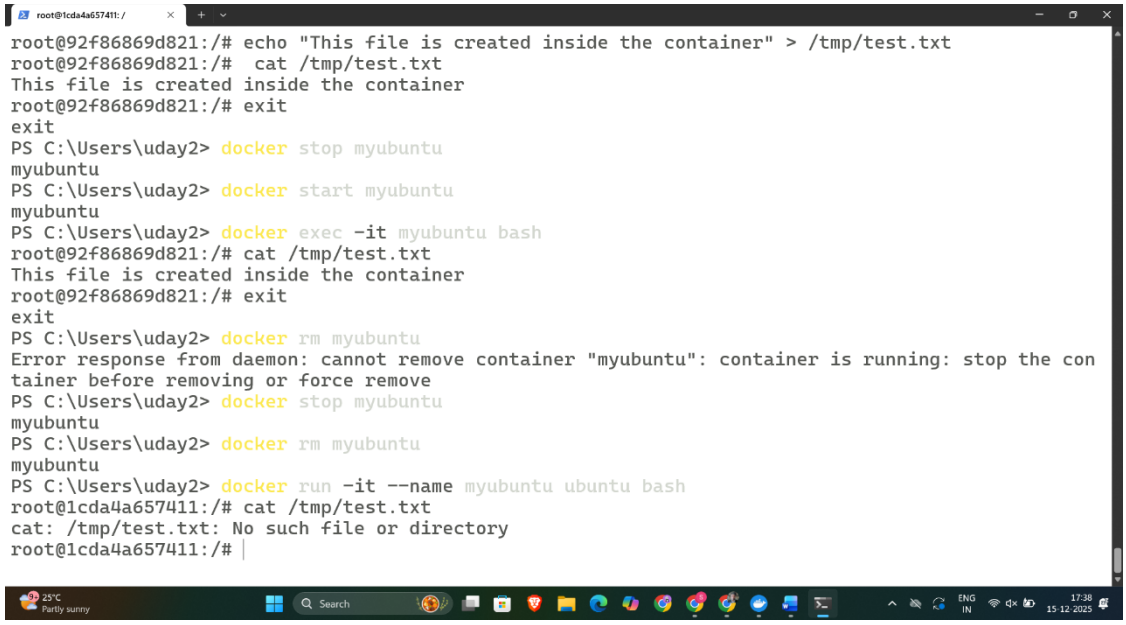
1. Start an Ubuntu container and jump into it with bash
2. Once you're inside:
    - Update the package list
    - Install curl and vim
    - Create a file at /tmp/test.txt with whatever content you want
3. Exit and restart the container
4. Check if your file is still there (spoiler: it won't be!)
5. Explain why your changes disappeared

Show me: Your explanation of why containers forget everything when you restart them


root@71190b22a9bc: /
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\uday2> wsl
uday2@UDAYxMVP:/mnt/c/Users/uday2$ cd ~
uday2@UDAYxMVP:~$ docker --version
Docker version 28.3.2, build 578ccf6
uday2@UDAYxMVP:~$ docker run -it --name myubuntu ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
20043066d3d5: Pull complete
Digest: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f176fb458d472dfc9e54
Status: Downloaded newer image for ubuntu:latest
root@71190b22a9bc:/# apt 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/main amd64 Packages [1736 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/restricted amd64 Packages [2874 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:10 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [2118 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [3048 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [35.9 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1943 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [49.4 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [34.3 kB]
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1183 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [33.1 kB]
Fetched 35.3 MB in 7s (5003 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@71190b22a9bc:/# apt install -y curl vim
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
ca-certificates krb5-locales libbrotli1 libcurl4t64 libexpat1 libgpm2 libgssapi-krb5-2 libk5crypto3 libkeyutils1


root@1cda4a657411: /
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\uday2> docker run -it --name myubuntu ubuntu bash
root@92f86869d821:/# apt update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]

root@1cda4a657411: /
Fetched 35.3 MB in 9s (3751 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
root@92f86869d821:/# apt install -y curl vim
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:


root@1cda4a657411: /
root@92f86869d821:/# echo "This file is created inside the container" > /tmp/test.txt
root@92f86869d821:/# cat /tmp/test.txt
This file is created inside the container
root@92f86869d821:/# exit
exit
PS C:\Users\uday2> docker stop myubuntu
myubuntu
PS C:\Users\uday2> docker start myubuntu
myubuntu
PS C:\Users\uday2> docker exec -it myubuntu bash
root@92f86869d821:/# cat /tmp/test.txt
This file is created inside the container
root@92f86869d821:/# exit
exit
PS C:\Users\uday2> docker rm myubuntu
Error response from daemon: cannot remove container "myubuntu": container is running: stop the con
tainer before removing or force remove
PS C:\Users\uday2> docker stop myubuntu
myubuntu
PS C:\Users\uday2> docker rm myubuntu
myubuntu
PS C:\Users\uday2> docker run -it --name myubuntu ubuntu bash
root@1cda4a657411:/# cat /tmp/test.txt
cat: /tmp/test.txt: No such file or directory
root@1cda4a657411:/#
25°C
Partly sunny
Search
ENG
IN
17:38
15-12-2025

Answer:

Step 0: Clean up existing container (if any previous is running)

docker stop myubuntu

docker rm myubuntu

Old container removed to start from scratch.

Step 1: Start a fresh Ubuntu container and enter bash

docker run -it --name myubuntu ubuntu bash

Entered Ubuntu container shell.

Step 2: Work inside the container

apt update

apt install -y curl vim

echo "This file is created inside the container" > /tmp/test.txt

cat /tmp/test.txt

Step 3: Exit and restart the same container

exit

docker stop myubuntu

docker start myubuntu

docker exec -it myubuntu bash

Step 4: Check file after restart

cat /tmp/test.txt

Result: File is still present (can see in image)

Step 5: Exit, delete, and recreate the container

exit

docker rm myubuntu

docker run -it --name myubuntu ubuntu bash

Step 6: Check file after recreation

cat /tmp/test.txt

Result: File is not found (can see in image)

Explanation:

Restarting a container preserves its filesystem state, so files remain after docker stop and docker start. When a container is deleted and recreated, Docker initializes a new filesystem and all previous changes are lost. Containers are ephemeral across removal, not across restarts. Since no Docker volume was used, the file did not persist after recreation.

Conclusion:

- Restarting the same container keeps data intact.
- Deleting and recreating a container removes all internal data.
- Docker volumes are required for persistent storage.

Assignment 3: Build Your First Docker Image

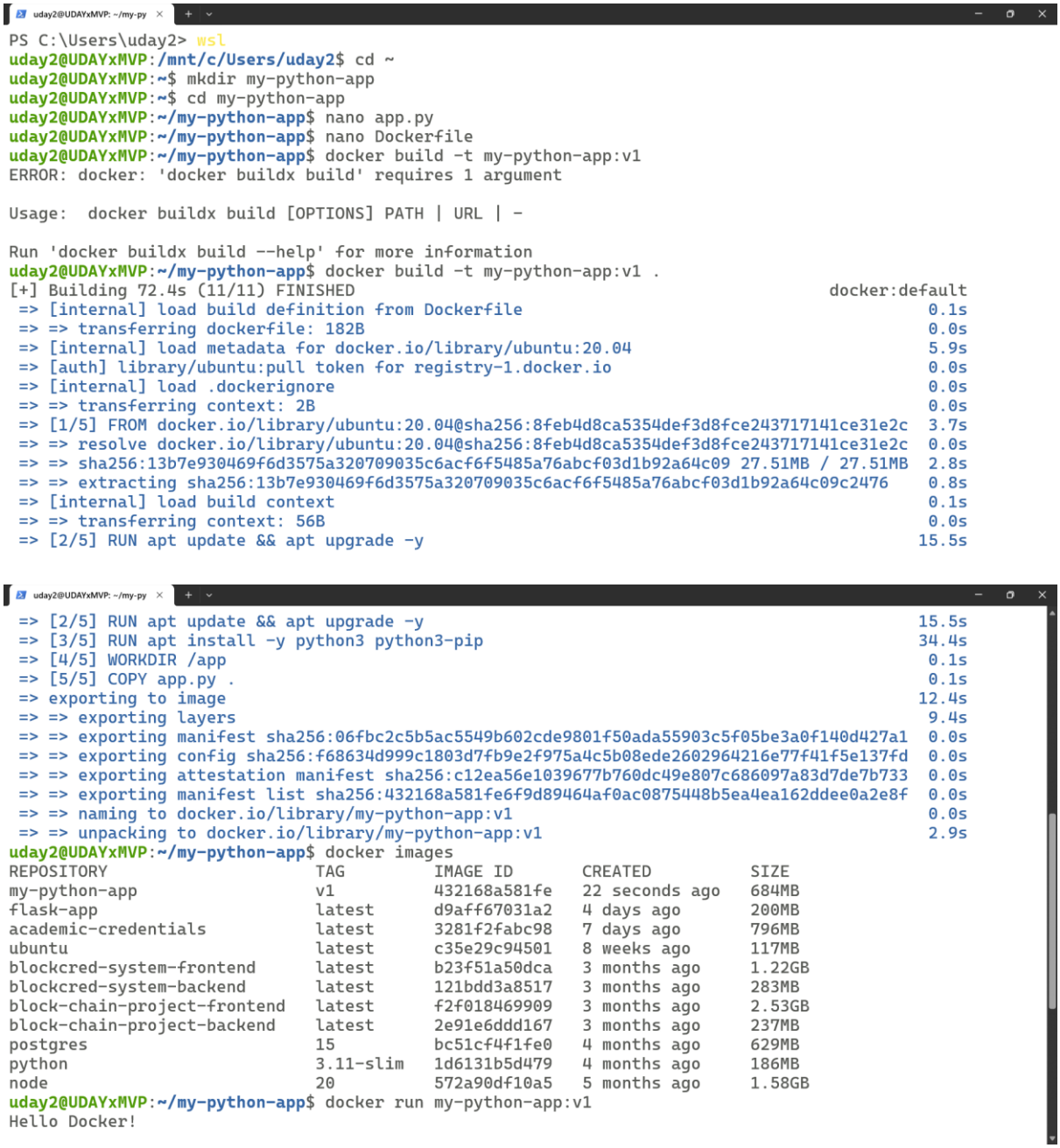
What you'll learn: How to write a Dockerfile from scratch

What to do:

1. Write a Dockerfile that:
    - Starts with Ubuntu 20.04
    - Updates all packages
    - Installs Python 3 and pip
    - Sets /app as the working folder
    - Copies a simple Python script that says "Hello Docker!"
    - Runs that script automatically when the container starts
2. Build your image and call it my-python-app:v1
3. Run a container from your new image

Show me: Your Dockerfile, the build output, and proof it's running

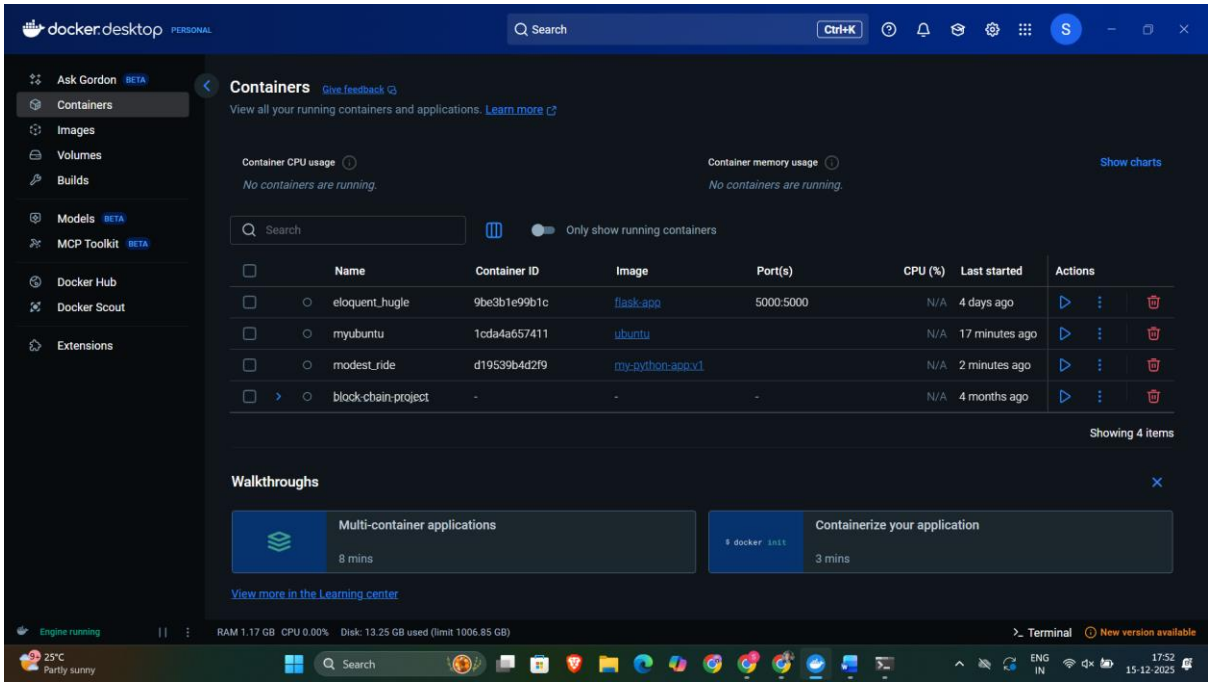
Answer:

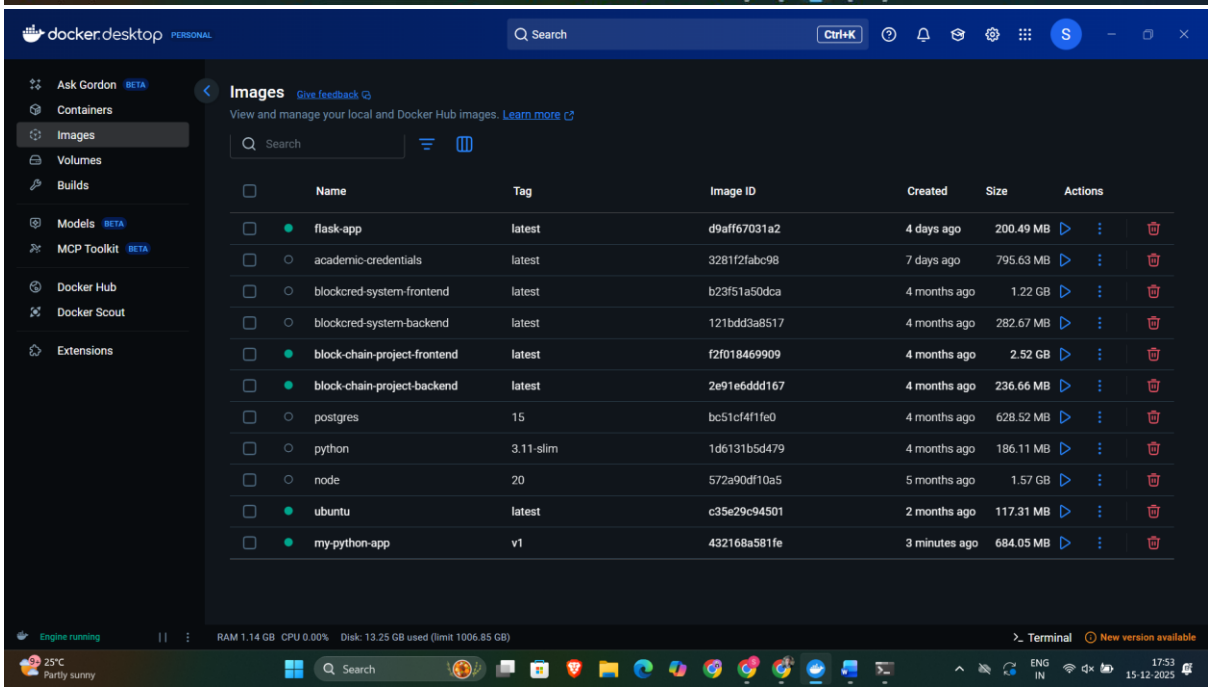


```
uday2@UDAYxMVP: ~/my-py
PS C:\Users\uday2> wsl
uday2@UDAYxMVP:/mnt/c/Users/uday2$ cd ~
uday2@UDAYxMVP:~$ mkdir my-python-app
uday2@UDAYxMVP:~$ cd my-python-app
uday2@UDAYxMVP:~/my-python-app$ nano app.py
uday2@UDAYxMVP:~/my-python-app$ nano Dockerfile
uday2@UDAYxMVP:~/my-python-app$ docker build -t my-python-app:v1
ERROR: docker: 'docker buildx build' requires 1 argument

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Run 'docker buildx build --help' for more information
uday2@UDAYxMVP:~/my-python-app$ docker build -t my-python-app:v1 .
[+] Building 72.4s (11/11) FINISHED                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                          0.1s
 => => transferring dockerfile: 182B                                                                          0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                                               5.9s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                                                 0.0s
 => [internal] load .dockerignore                                                                             0.0s
 => => transferring context: 2B                                                                               0.0s
 => [1/5] FROM docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c                   3.7s
 => => resolve docker.io/library/ubuntu:20.04@sha256:8feb4d8ca5354def3d8fce243717141ce31e2c                   0.0s
 => => sha256:13b7e930469f6d3575a320709035c6acf6f5485a76abcf03d1b92a64c09 27.51MB / 27.51MB                   2.8s
 => => extracting sha256:13b7e930469f6d3575a320709035c6acf6f5485a76abcf03d1b92a64c09c2476                      0.8s
 => [internal] load build context                                                                             0.1s
 => => transferring context: 56B                                                                              0.0s
 => [2/5] RUN apt update && apt upgrade -y                                                                   15.5s
```



```
uday2@UDAYxMVP: ~/my-py
 => [2/5] RUN apt update && apt upgrade -y                                                                   15.5s
 => [3/5] RUN apt install -y python3 python3-pip                                                             34.4s
 => [4/5] WORKDIR /app                                                                                        0.1s
 => [5/5] COPY app.py .                                                                                       0.1s
 => exporting to image                                                                                       12.4s
 => => exporting layers                                                                                       9.4s
 => => exporting manifest sha256:06fbc2c5b5ac5549b602cde9801f50ada55903c5f05be3a0f140d427a1                    0.0s
 => => exporting config sha256:f68634d999c1803d7fb9e2f975a4c5b08ede2602964216e77f41f5e137fd                    0.0s
 => => exporting attestation manifest sha256:c12ea56e1039677b760dc49e807c686097a83d7de7b733                    0.0s
 => => exporting manifest list sha256:432168a581fe6f9d89464af0ac0875448b5ea4ea162ddee0a2e8f                    0.0s
 => => naming to docker.io/library/my-python-app:v1                                                           0.0s
 => => unpacking to docker.io/library/my-python-app:v1                                                        2.9s
uday2@UDAYxMVP:~/my-python-app$ docker images
REPOSITORY                      TAG         IMAGE ID       CREATED          SIZE
my-python-app                   v1          432168a581fe   22 seconds ago   684MB
flask-app                       latest      d9aff67031a2   4 days ago       200MB
academic-credentials            latest      3281f2fabc98   7 days ago       796MB
ubuntu                          latest      c35e29c94501   8 weeks ago      117MB
blockcred-system-frontend       latest      b23f51a50dca   3 months ago     1.22GB
blockcred-system-backend        latest      121bdd3a8517   3 months ago     283MB
block-chain-project-frontend    latest      f2f018469909   3 months ago     2.53GB
block-chain-project-backend     latest      2e91e6ddd167   3 months ago     237MB
postgres                        15          bc51cf4f1fe0   4 months ago     629MB
python                          3.11-slim   1d6131b5d479   4 months ago     186MB
node                            20          572a90df10a5   5 months ago     1.58GB
uday2@UDAYxMVP:~/my-python-app$ docker run my-python-app:v1
Hello Docker!
```





Step 0: Open terminal

Opened WSL terminal after ensuring Docker Engine was running.

Step 1: Create project directory

Created a new directory for the Docker project and moved into it.

```
mkdir my-python-app
cd my-python-app
```

Step 2: Create a simple Python script

Created a Python file that prints a message.

nano app.py

Content of app.py:

print("Hello Docker!")

Saved and exited the file.

Step 3: Create Dockerfile

Created a Dockerfile from scratch.

nano Dockerfile

Content of Dockerfile:

FROM ubuntu:20.04

RUN apt update && apt upgrade -y

RUN apt install -y python3 python3-pip

WORKDIR /app

COPY app.py .

CMD ["python3", "app.py"]

Saved and exited the file.

Step 4: Build Docker image

Built the Docker image using the Dockerfile and tagged it as my-python-app:v1.

docker build -t my-python-app:v1 .

During the build process, Docker pulled the Ubuntu 20.04 image, updated packages, installed Python and pip, and successfully built the image.

Step 5: Verify image creation

Checked available Docker images to confirm the image was created.

docker images

The image my-python-app with tag v1 is visible in the list.

Step 6: Run container from the image

Ran a container using the newly built image.

docker run my-python-app:v1

Output displayed:

Hello Docker!

This confirms that the Python script ran automatically when the container started.

Command History:

docker build -t my-python-app:v1 .

docker images

docker run my-python-app:v1

Conclusion:

A Dockerfile was written from scratch using Ubuntu 20.04. Python 3 and pip were installed, a working directory was set, a Python script was copied into the image, and the script executed automatically when the container started. The image was successfully built and run.

Assignment 4: Make Smaller, Smarter Images

What you'll learn: Multi-stage builds to create tiny images

What to do:

1. Create a Dockerfile for a Go app that uses two stages:
    - First stage: Build your Go program
    - Second stage: Copy just the final program to a tiny Alpine image
2. Build it both ways (single-stage and multi-stage)
3. Compare how big the images are
4. Explain why multi-stage builds are awesome

Show me: Your Dockerfile, size comparison, and your thoughts on why this matters

Answer:

mkdir go-docker-demo

cd go-docker-demo

 # Create Go app

nano main.go

```
package main
import "fmt"
func main() {
    fmt.Println("Hello from Go Docker App!")
}
```

 # Single-stage Dockerfile

nano Dockerfile.single

```
FROM golang:1.21
ENV CGO_ENABLED=0 GOOS=linux GOARCH=amd64
WORKDIR /app
COPY main.go .
RUN go build -o app main.go
CMD ["./app"]
```

 # Build single-stage image

```
docker build -f Dockerfile.single -t go-single .
```

 # Multi-stage Dockerfile

nano Dockerfile.multi

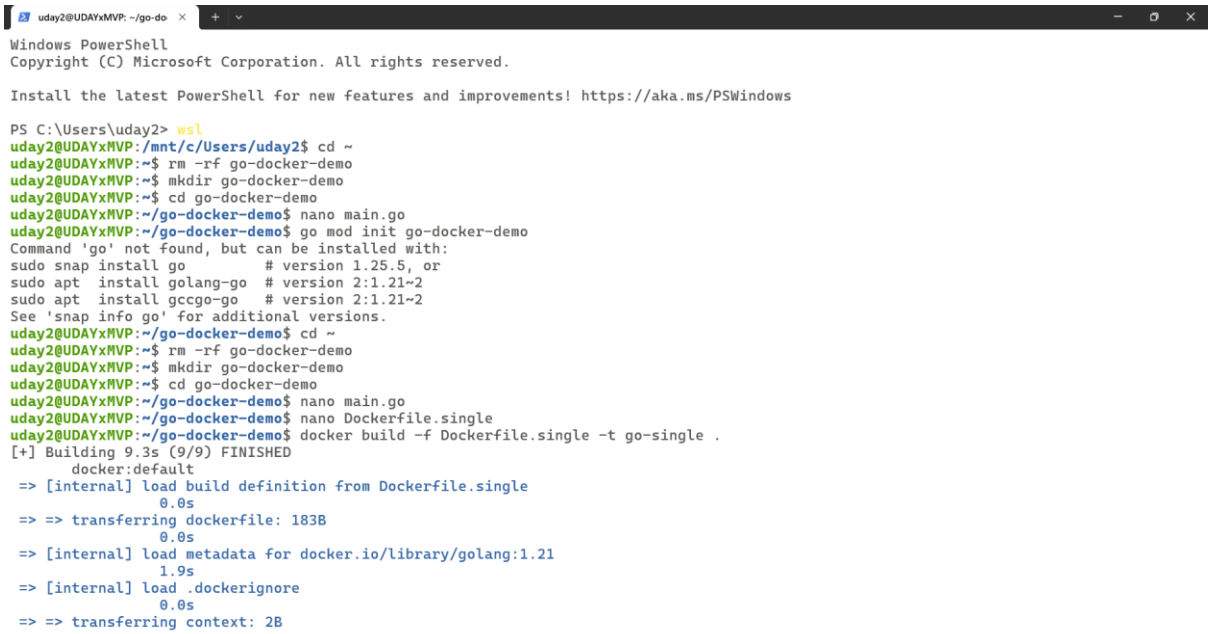
```
FROM golang:1.21 AS builder
ENV CGO_ENABLED=0 GOOS=linux GOARCH=amd64
WORKDIR /app
COPY main.go .
RUN go build -o app main.go
FROM alpine:latest
WORKDIR /app
COPY --from=builder /app/app .
CMD ["./app"]
```

 # Build multi-stage image

```
docker build -f Dockerfile.multi -t go-multi .
```

 # Compare image sizes

```
docker images | grep go-
```



```
uday2@UDAYxMVP:~/go-docker-demo$ nano Dockerfile.multi
uday2@UDAYxMVP:~/go-docker-demo$ docker build -f Dockerfile.multi -t go-multi .
[+] Building 3.5s (15/15) FINISHED
       docker:default
 => [internal] load build definition from Dockerfile.multi
                0.0s
 => => transferring dockerfile: 256B
                0.0s
 => [internal] load metadata for docker.io/library/alpine:latest
                2.2s
 => [internal] load metadata for docker.io/library/golang:1.21
                2.6s
 => [auth] library/golang:pull token for registry-1.docker.io
                0.0s
 => [auth] library/alpine:pull token for registry-1.docker.io
                0.0s
 => [internal] load .dockerignore
                0.0s

uday2@UDAYxMVP:~/go-docker-demo$ docker build -f Dockerfile.multi -t go-multi .
[+] Building 0.9s (13/13) FINISHED
       docker:default
 => [internal] load build definition from Dockerfile.multi
                0.0s
 => => transferring dockerfile: 256B
                0.0s
 => [internal] load metadata for docker.io/library/alpine:latest
                0.6s
 => [internal] load metadata for docker.io/library/golang:1.21
                0.6s
 => [internal] load .dockerignore
                0.0s
```

Result:

```
uday2@UDAYxMVP:~/go-docker-demo$ docker images | grep go-
go-multi                      latest      0038fa72a081   12 seconds ago       15.9MB
go-single                     latest      b481edf2a4cb   About a minute ago   1.23GB
uday2@UDAYxMVP:~/go-docker-demo$
```

Why This Matters

Multi-stage builds help reduce Docker image size by keeping only the final program and removing extra build tools.
This makes the image faster to download, more secure, and better for real-world use.

Assignment 5: Keep Your Data Safe with Volumes

What you'll learn: How to make sure your data doesn't disappear

What to do:

1. Create a volume named mysql-data
2. Run a MySQL container that uses this volume
3. Create a database with a table and some data in it

4. Stop and completely remove the container
5. Start a fresh MySQL container using the same volume
6. Check if your data is still there (it should be!)

Show me: All your commands and proof that the data survived

```
uday2@UDAYxMVP:~$ mkdir docker-assignment-5
uday2@UDAYxMVP:~$ cd docker-assignment-5
uday2@UDAYxMVP:~/docker-assignment-5$ docker volume create mysql-data
mysql-data
uday2@UDAYxMVP:~/docker-assignment-5$ docker volume ls
DRIVER    VOLUME NAME
local     fd4c6ec0682c682758c05f0292242a731e4378da4fdf99488c937d18c89871d3
local     mysql-data
uday2@UDAYxMVP:~/docker-assignment-5$ docker run -d \
> -e MYSQL_ROOT_PASSWORD=root123 \
> -p 3306:3306 \
> -v mysql-data:/var/lib/mysql \
> mysql:8.0
Unable to find image 'mysql:8.0' locally
8.0: Pulling from library/mysql
98c09a342d40: Pull complete
a3c731ffd1a3: Pull complete
6b8812fa3131: Pull complete
3a7bd505eaa0: Pull complete
4f1ef7d4d76b: Pull complete
0ddab5f772d8: Pull complete
907afe780e24: Pull complete
8c3cd0b3401c: Pull complete
96682d77f5a0: Pull complete
5f838d3bc363: Pull complete
7a5e1e917526: Pull complete
Digest: sha256:0275a35e79c60caae68fac520602d9f6897feb9b0941a1471196b1a01760e581
Status: Downloaded newer image for mysql:8.0
6d4082d9b5f64bb1ad20956caf558c4008844d89fbc5b6bdda59d090cb9dfa3f
docker: Error response from daemon: ports are not available: exposing port TCP 0.0.0.0:3306 -> 127.0.0.1:0: listen tcp 0.0.0.0:3306: b
ind: Only one usage of each socket address (protocol/network address/port) is normally permitted.

Run 'docker run --help' for more information
uday2@UDAYxMVP:~/docker-assignment-5$ docker exec -it mysql-container mysql -u root -p
Error response from daemon: No such container: mysql-container
```

```
ind: Only one usage of each socket address (protocol/network address/port) is normally permitted.

Run 'docker run --help' for more information
uday2@UDAYxMVP:~/docker-assignment-5$ docker exec -it mysql-container mysql -u root -p
Error response from daemon: No such container: mysql-container
uday2@UDAYxMVP:~/docker-assignment-5$ docker exec -it mysql-container mysql -u root -p
Error response from daemon: No such container: mysql-container
uday2@UDAYxMVP:~/docker-assignment-5$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                     PORTS     NAMES
6d4082d9b5f6   mysql:8.0      "docker-entrypoint.s…"   34 seconds ago   Created                              stoic_johnson
815764b8838a   b37334f0b51a   "./app"                  4 hours ago      Exited (0) 4 hours ago               frosty_merkle
d5e50d59a94b   46f917cb12f4   "./app"                  4 hours ago      Exited (0) 4 hours ago               festive_hawking
uday2@UDAYxMVP:~/docker-assignment-5$ docker stop $(docker ps -aq)
6d4082d9b5f6
815764b8838a
d5e50d59a94b
uday2@UDAYxMVP:~/docker-assignment-5$ docker rm $(docker ps -aq)
6d4082d9b5f6
815764b8838a
d5e50d59a94b
uday2@UDAYxMVP:~/docker-assignment-5$ docker run -d \
--name mysql-container \
-e MYSQL_ROOT_PASSWORD=root123 \
-p 3307:3306 \
-v mysql-data:/var/lib/mysql \
mysql:8.0\
> ^C
uday2@UDAYxMVP:~/docker-assignment-5$ docker run -d \
--name mysql-container \
-e MYSQL_ROOT_PASSWORD=root123 \
-p 3307:3306 \
-v mysql-data:/var/lib/mysql \
mysql:8.0
0cf03483c250f469d670ea94c9c69d5173642cb2cf2628b5b3df0a6231908b5a
uday2@UDAYxMVP:~/docker-assignment-5$ docker ps
```

Commands Used:

 # Open WSL and go to home

wsl

cd ~

 # Create working directory

mkdir docker-assignment-5

cd docker-assignment-5

 # Create Docker volume

```
docker volume create mysql-data
docker volume ls
```

 # Run MySQL container with volume

```
docker run -d \
--name mysql-container \
-e MYSQL_ROOT_PASSWORD=root123 \
-p 3307:3306 \
-v mysql-data:/var/lib/mysql \
mysql:8.0
```

 # Enter MySQL container

```
docker exec -it mysql-container mysql -u root -p
```

-- Create database and table

```
CREATE DATABASE testdb;
USE testdb;
CREATE TABLE users (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(50)
);
INSERT INTO users (name) VALUES ('Uday');
SELECT * FROM users;
```

(Result see in above picture)

 # Stop and remove container

```
docker stop mysql-container
docker rm mysql-container
```

 # Run new container with same volume

docker run -d \

--name mysql-container-new \

-e MYSQL_ROOT_PASSWORD=root123 \

-p 3307:3306 \

-v mysql-data:/var/lib/mysql \

mysql:8.0

 # Enter new container

docker exec -it mysql-container-new mysql -u root -p

-- Check if data survived

USE testdb;

SELECT * FROM users;

```
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> INSERT INTO users (name) VALUES ('Uday');
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM users;
+----+------+
| id | name |
+----+------+
|  1 | Uday |
+----+------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users;
+----+------+
| id | name |
+----+------+
|  1 | Uday |
+----+------+
1 row in set (0.01 sec)

mysql> use testdb;
Database changed
mysql> Select*From users;
+----+------+
| id | name |
+----+------+
|  1 | Uday |
+----+------+
1 row in set (0.08 sec)

mysql>
```

Assignment 6: Connect Your Local Files to Containers

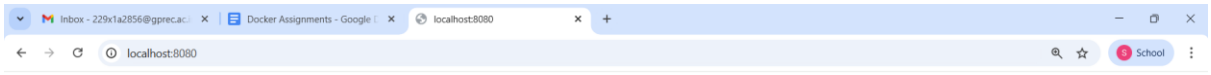
What you'll learn: Bind mounts for live development

What to do:

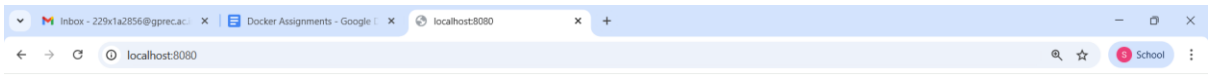
1. Make a folder on your computer with an index.html file
2. Run nginx and connect your local folder to the container's web folder
3. Edit your index.html file on your computer
4. Refresh your browser - see the changes without restarting anything!
5. Explain the difference between volumes and bind mounts

Show me: Your setup and an explanation of when to use each

ANSWER:



Hello from Bind Mount



Updated Live Without Restart

```
uday2@UDAYxMVP:~/docker-assignment-5$ cd ..
uday2@UDAYxMVP:~$ mkdir docker-assignment-6
uday2@UDAYxMVP:~$ cd docker-assignment-6
uday2@UDAYxMVP:~/docker-assignment-6$ nano index.html
uday2@UDAYxMVP:~/docker-assignment-6$ docker run -d \
--name nginx-bind \
-p 8080:80 \
-v $(pwd):/usr/share/nginx/html \
nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
7382b41547b8: Pull complete
9ee60c6c0558: Pull complete
5b219a92f92a: Pull complete
ee3a09d2248a: Pull complete
114e699da838: Pull complete
5b5fa0b64d74: Pull complete
Digest: sha256:fb01117203ff38c2f9af91db1a7409459182a37c87cced5cb442d1d8fcc66d19
Status: Downloaded newer image for nginx:latest
d4f888d96d9624898c4435405a8448795b8a067c72051971fa213b1e7ec19766
uday2@UDAYxMVP:~/docker-assignment-6$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED         STATUS         PORTS
    NAMES
d4f888d96d96   nginx       "/docker-entrypoint.…"   3 seconds ago   Up 2 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
    nginx-bind
0cf03483c250   mysql:8.0   "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes   0.0.0.0:3307->3306/tcp, [::]:3307->3306/tc
p   mysql-container
uday2@UDAYxMVP:~/docker-assignment-6$ nano index.html
uday2@UDAYxMVP:~/docker-assignment-6$ nano index.html
uday2@UDAYxMVP:~/docker-assignment-6$
```

Difference Between Volumes and Bind Mounts

- **Volumes** are managed by Docker and store data safely outside the container. They are mainly used to keep important data like databases, even if the container is deleted.
- **Bind mounts** link a folder from the local computer directly to a container. They are mainly used during development so that file changes on the local system appear instantly inside the container.

When to Use Each

- Use **bind mounts** for live development and testing (HTML, code files).
- Use **volumes** for persistent data storage (MySQL, MongoDB, PostgreSQL).

Bind mounts are used for live file updates, while volumes are used for permanent data storage.

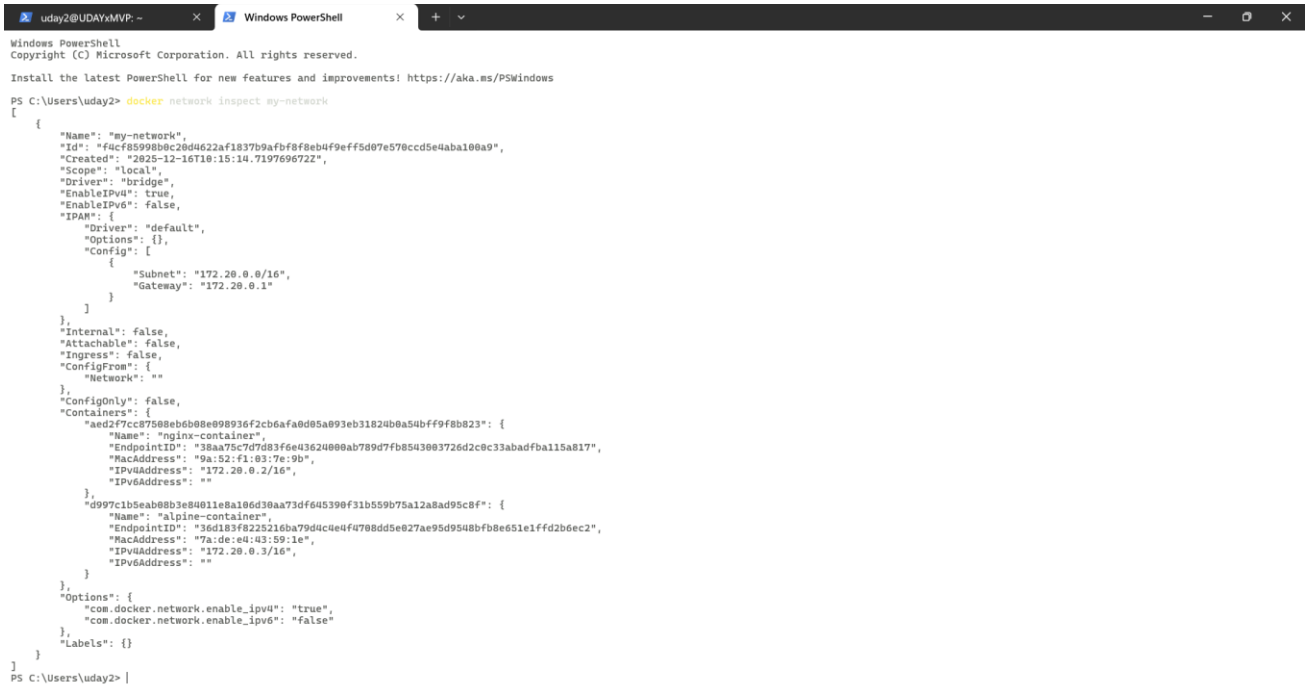
Assignment 7: Let Containers Talk to Each Other

What you'll learn: Docker networking basics

What to do:

1. **Create a network called my-network**
2. **Start two containers (nginx and alpine) on this network**
3. **From inside the alpine container, ping the nginx container by name**
4. **Look at what's connected to your network**
5. **Explain how containers can find each other by name**

Show me: Your network commands and proof that containers can communicate



Explanation:

Docker provides an internal DNS for custom networks.
Containers connected to the same network can reach each other using **container names instead of IP addresses**.

Containers on the same Docker network can find each other by name using Docker's built-in DNS.

Assignment 8: Build a Real Two-Part App

What you'll learn: Connecting multiple containers to work together

What to do:

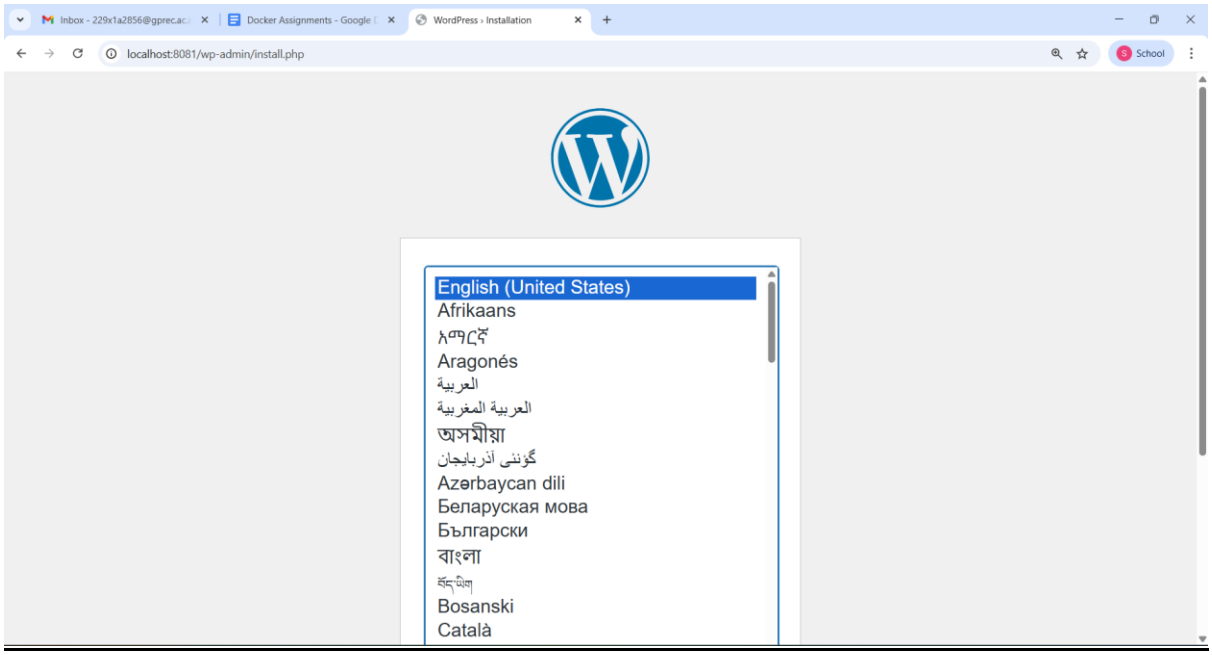
1. Set up a network

2. Start a MySQL container on that network
3. Start a WordPress container on the same network
4. Make WordPress talk to MySQL
5. Open WordPress in your browser and set it up

Show me: A working WordPress site with its database connection

```
uday2@UDAYxMVP: ~/docker
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\uday2> wsl
uday2@UDAYxMVP:/mnt/c/Users/uday2$ cd ~
uday2@UDAYxMVP:~$ mkdir docker-assignment-8
uday2@UDAYxMVP:~$ cd docker-assignment-8
uday2@UDAYxMVP:~/docker-assignment-8$ docker network create wp-network
a94d21eb9fbf5f769e14367a325c973ae136836a80d85b416a2d4d06c10bc384
uday2@UDAYxMVP:~/docker-assignment-8$ docker network ls
NETWORK ID     NAME                                    DRIVER    SCOPE
c0e91dd51411   block-chain-project_default             bridge    local
e455863b5958   blockchain-credentials-webapp_default   bridge    local
ca63aeaaf990   bridge                                  bridge    local
7035050f5b66   host                                    host      local
f4cf85998b0c   my-network                              bridge    local
56ea782f6bcd   none                                    null      local
a94d21eb9fbf   wp-network                              bridge    local
uday2@UDAYxMVP:~/docker-assignment-8$ docker run -d \
--name wp-mysql \
--network wp-network \
-e MYSQL_DATABASE=wordpress \
-e MYSQL_USER=wpuser \
-e MYSQL_PASSWORD=wppass \
-e MYSQL_ROOT_PASSWORD=rootpass \
mysql:8.0
1a152cdb39b460cd8a9d70fb89f00831854728b2959f533bcbf1f2ba3769e954
uday2@UDAYxMVP:~/docker-assignment-8$ docker run -d \
--name wordpress \
--network wp-network \
-p 8081:80 \
-e WORDPRESS_DB_HOST=wp-mysql:3306 \
-e WORDPRESS_DB_USER=wpuser \
-e WORDPRESS_DB_PASSWORD=wppass \
-e WORDPRESS_DB_NAME=wordpress \
wordpress

Digest: sha256:91eeee906a3edacc66550eeca85aff027e13754aba64a98bdb8f9598e8dadeee
Status: Downloaded newer image for wordpress:latest
e3ccf127e7df0b3ae9ee268a822bed6e35a7a22bf204e04c1204e52f0f8b687b
uday2@UDAYxMVP:~/docker-assignment-8$
```

```
Digest: sha256:91eeee906a3edacc66550eeca85aff027e13754aba64a98bdb8f9598e8dadeee
Status: Downloaded newer image for wordpress:latest
e3ccf127e7df0b3ae9ee268a822bed6e35a7a22bf204e04c1204e52f0f8b687b
uday2@UDAYxMVP:~/docker-assignment-8$ docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
e3ccf127e7df   wordpress   "docker-entrypoint.s…"   44 seconds ago   Up 43 seconds   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp                wordpress
1a152cdb39b4   mysql:8.0   "docker-entrypoint.s…"   2 minutes ago    Up 2 minutes    3306/tcp, 33060/tcp                                    wp-mysql
d997c1b5eab0   alpine      "sh"                     11 minutes ago   Up 11 minutes                                                          alpine-contai
ner
aed2f7cc8750   nginx       "/docker-entrypoint.…"   11 minutes ago   Up 11 minutes   80/tcp                                                 nginx-contain
er
d4f888d96d96   nginx       "/docker-entrypoint.…"   18 minutes ago   Up 18 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp                nginx-bind
0cf03483c250   mysql:8.0   "docker-entrypoint.s…"   27 minutes ago   Up 27 minutes   0.0.0.0:3307->3306/tcp, [::]:3307->3306/tcp            mysql-contain
er
uday2@UDAYxMVP:~/docker-assignment-8$
```



1. WordPress and MySQL are running in separate containers but connected using a Docker network.
2. WordPress connects to MySQL using the database container name, which Docker resolves automatically.

Containers communicate using Docker networks and can connect to each other using container names.

Assignment 9: Fix This Messy Dockerfile

What you'll learn: Writing clean, efficient Dockerfiles

What to do:

1. Look at this terrible Dockerfile:

```
FROM ubuntu

RUN apt-get update

RUN apt-get install -y python3

RUN apt-get install -y python3-pip
```

RUN apt-get install -y curl

RUN apt-get install -y git

COPY . /app

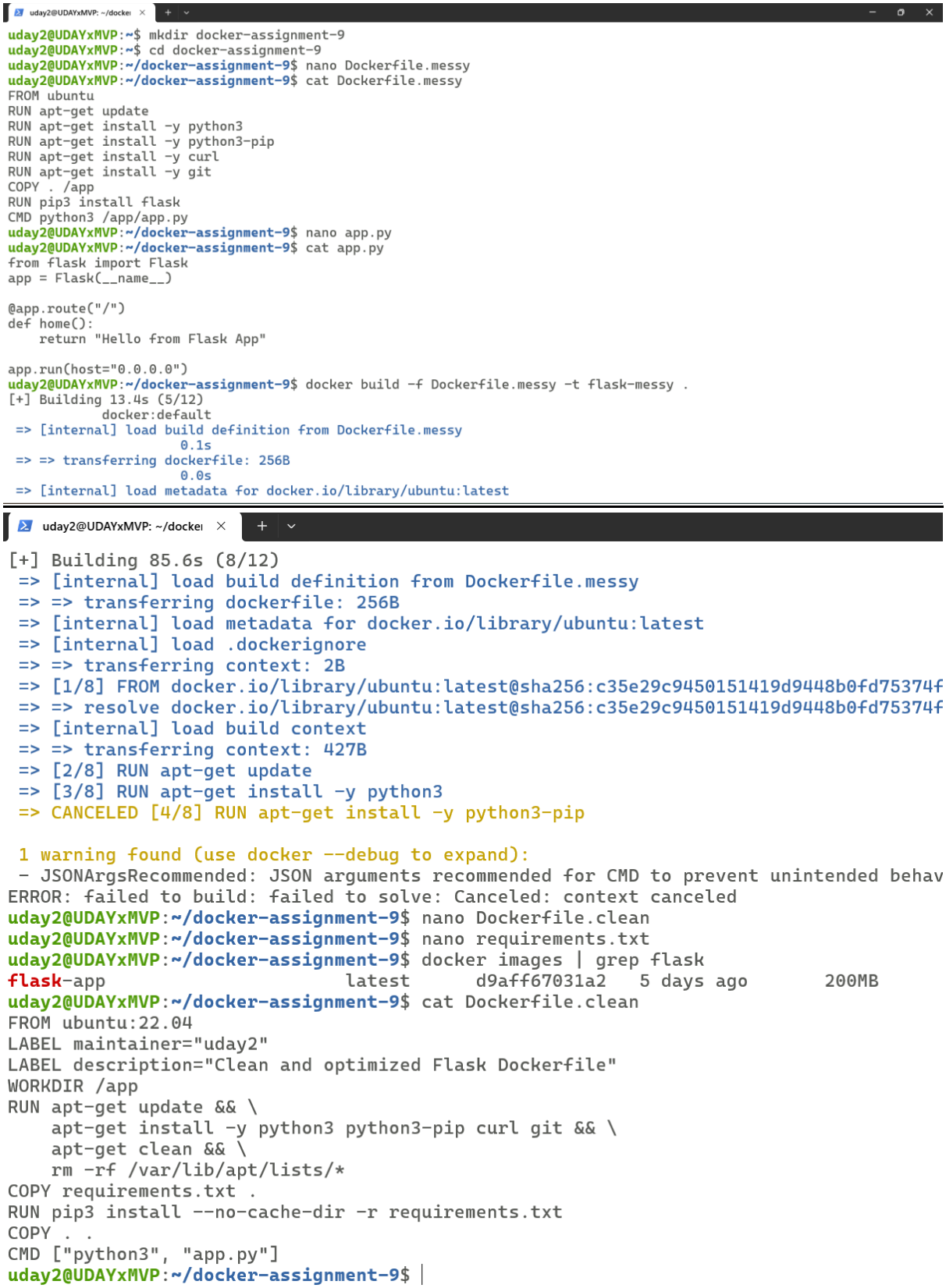
RUN pip3 install flask

CMD python3 /app/app.py

2. Rewrite it properly:
    - Put multiple RUN commands together
    - Use a specific version (not just "ubuntu")
    - Clean up after installing packages
    - Use WORKDIR instead of just /app everywhere
    - Add some useful labels
    - Make it cache-friendly
3. Build both and compare the sizes

Show me: Your improved Dockerfile and explain what you fixed

1. **I combined multiple RUN commands to reduce image layers.**
2. **I used a specific Ubuntu version and cleaned the package cache to reduce image size.**
3. **I added WORKDIR, labels, and improved caching to make the Dockerfile cleaner and efficient.**



Assignment 10: Your First Docker Compose File

What you'll learn: Managing multiple containers easily

What to do:

1. Write a docker-compose.yml with:
    - An nginx web server on port 80
    - A PostgreSQL database
    - A volume so the database doesn't lose data
    - A network connecting everything
2. Start it all with docker-compose up
3. Clean everything up with docker-compose down

Show me: Your compose file and screenshots of it running

OUTPUT:

```
uday2@UDAYxMVP:~$ mkdir docker-assignment-10
uday2@UDAYxMVP:~$ cd docker-assignment-10
uday2@UDAYxMVP:~/docker-assignment-10$ pwd
/home/uday2/docker-assignment-10
uday2@UDAYxMVP:~/docker-assignment-10$ nano docker-compose.yml
uday2@UDAYxMVP:~/docker-assignment-10$ docker compose up -d
WARN[0000] /home/uday2/docker-assignment-10/docker-compose.yml: the attribute `version` is obsolete, it will b
e ignored, please remove it to avoid potential confusion
[+] Running 4/4
 ✔Network docker-assignment-10_app-network  Created
                  0.1s
 ✔Volume "docker-assignment-10_pg-data"     Created
                  0.0s
 ✔Container docker-assignment-10-web-1      Started
                  0.8s
 ✔Container docker-assignment-10-db-1       Started
                  0.7s
uday2@UDAYxMVP:~/docker-assignment-10$ docker compose ps
WARN[0000] /home/uday2/docker-assignment-10/docker-compose.yml: the attribute `version` is obsolete, it will b
e ignored, please remove it to avoid potential confusion
NAME                        IMAGE          COMMAND                  SERVICE   CREATED         STATUS
PORTS
docker-assignment-10-db-1   postgres:15    "docker-entrypoint.s…"   db        5 seconds ago   Up 5 seconds
5432/tcp
docker-assignment-10-web-1  nginx:latest   "/docker-entrypoint.…"   web       5 seconds ago   Up 5 seconds
0.0.0.0:8083->80/tcp, [::]:8083->80/tcp
uday2@UDAYxMVP:~/docker-assignment-10$
```

Assignment 11: Handle Passwords and Settings

What you'll learn: Working with environment variables securely

What to do:

1. Make a Dockerfile that uses environment variables
2. Build an image that prints out environment variables
3. Run it three different ways:
    - Pass variables directly with the -e flag
    - Use a .env file
    - Define them in docker-compose.yml
4. Show all three methods working

Show me: Your Dockerfile, commands, and output from each method

```
uday2@UDAYxMVP:~$ mkdir docker-assignment-11
uday2@UDAYxMVP:~$ cd docker-assignment-11
uday2@UDAYxMVP:~/docker-assignment-11$ nano Dockerfile
uday2@UDAYxMVP:~/docker-assignment-11$ docker build -t env-demo .
[+] Building 19.6s (7/7) FINISHED                                                                                docker:default
 => [internal] load build definition from Dockerfile                                                                      0.1s
 => => transferring dockerfile: 203B                                                                                      0.0s
 => [internal] load metadata for docker.io/library/ubuntu:22.04                                                           5.7s
 => [auth] library/ubuntu:pull token for registry-1.docker.io                                                             0.0s
 => [internal] load .dockerignore                                                                                         0.1s
 => => transferring context: 2B                                                                                           0.0s
 => [1/2] FROM docker.io/library/ubuntu:22.04@sha256:104ae83764a5119017b8e8d6218fa0832b09df65aae7d5a6d                    12.9s
 => => resolve docker.io/library/ubuntu:22.04@sha256:104ae83764a5119017b8e8d6218fa0832b09df65aae7d5a6de                    0.0s
 => => sha256:7e49dc6156b0b532730614d83a65ae5e7ce61e966b0498703d333b4d03505e4f 29.54MB / 29.54MB                          12.0s
 => => extracting sha256:7e49dc6156b0b532730614d83a65ae5e7ce61e966b0498703d333b4d03505e4f                                  0.8s
 => [2/2] WORKDIR /app                                                                                                    0.2s
 => exporting to image                                                                                                    0.4s
 => => exporting layers                                                                                                   0.2s
 => => exporting manifest sha256:5fabefb0dc1697b63feba3fbbbad102d22da22790a1fb1fc19f4bde2c714bb0e                          0.0s
 => => exporting config sha256:85aaab2278c7d557944895701d15e925ea31b7e4b9b7006df5f94ed85af7e68e                           0.0s
 => => exporting attestation manifest sha256:b422b3f51d9503f1e2c32a83fad5c3fbcac52c2009bd4653b894d576d3                   0.0s
 => => exporting manifest list sha256:4afeee9314d7b8fa1d14d4ddcf43a98d1cb68c9e6dbd878408e827d0c12f4085                    0.0s
 => => naming to docker.io/library/env-demo:latest                                                                        0.0s
 => => unpacking to docker.io/library/env-demo:latest                                                                     0.0s

 1 warning found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signal
s (line 6)
uday2@UDAYxMVP:~/docker-assignment-11$ docker run --rm \
```

```
 1 warning found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signal
s (line 6)
uday2@UDAYxMVP:~/docker-assignment-11$ docker run --rm \
-e APP_NAME=MyApp \
-e APP_ENV=production \
env-demo
App Name: MyApp && Environment: production
uday2@UDAYxMVP:~/docker-assignment-11$ nano .env
uday2@UDAYxMVP:~/docker-assignment-11$ docker run --rm --env-file .env env-demo
App Name: EnvFileApp && Environment: testing
uday2@UDAYxMVP:~/docker-assignment-11$ nano docker-compose.yml
uday2@UDAYxMVP:~/docker-assignment-11$ docker compose up
WARN[0000] /home/uday2/docker-assignment-11/docker-compose.yml: the attribute `version` is obsolete, it will b
e ignored, please remove it to avoid potential confusion
[+] Running 2/2
 ✔Network docker-assignment-11_default  Created                                                                           0.1s
 ✔Container docker-assignment-11-app-1  Created                                                                           0.1s
Attaching to app-1
app-1  | App Name: ComposeApp && Environment: staging
app-1 exited with code 0
uday2@UDAYxMVP:~/docker-assignment-11$ docker compose down
WARN[0000] /home/uday2/docker-assignment-11/docker-compose.yml: the attribute `version` is obsolete, it will b
e ignored, please remove it to avoid potential confusion
[+] Running 2/2
 ✔Container docker-assignment-11-app-1  Removed                                                                           0.0s
 ✔Network docker-assignment-11_default  Removed                                                                           0.3s
uday2@UDAYxMVP:~/docker-assignment-11$
```

```
uday2@UDAYxMVP:~/docker-assignment-11$ cat .env
APP_NAME=EnvFileApp
APP_ENV=testing
uday2@UDAYxMVP:~/docker-assignment-11$ cat docker-compose.yml
version: "3.8"

services:
  app:
    image: env-demo
    environment:
      APP_NAME: ComposeApp
      APP_ENV: staging
uday2@UDAYxMVP:~/docker-assignment-11$
```

Environment variables allow passing sensitive data like passwords and settings without hardcoding them into images.
They can be provided using command-line flags, env files, or Docker Compose.

Environment variables help manage configuration securely and flexibly in Docker containers.

Assignment 12: Check if Containers Are Healthy

What you'll learn: Built-in health monitoring

What to do:

1. Write a Dockerfile with a HEALTHCHECK
2. The check should make sure a web service is responding
3. Run the container and watch its health status
4. Show what happens when the service fails the health check

Show me: Your Dockerfile with the health check and the monitoring output

Docker HEALTHCHECK monitors whether a web service is responding.
When nginx stops, Docker detects the failure and marks the container unhealthy.

Health checks help Docker automatically detect and report service failures inside containers.

Assignment 13: SSH Into Containers (Carefully!)

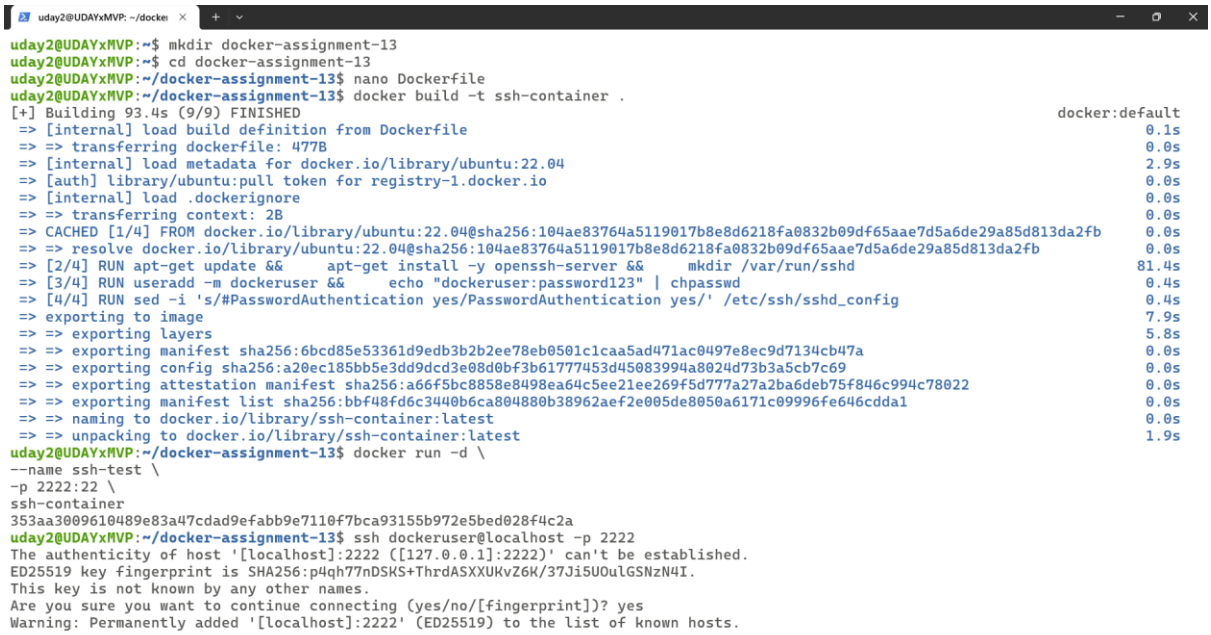
What you'll learn: Remote access to containers

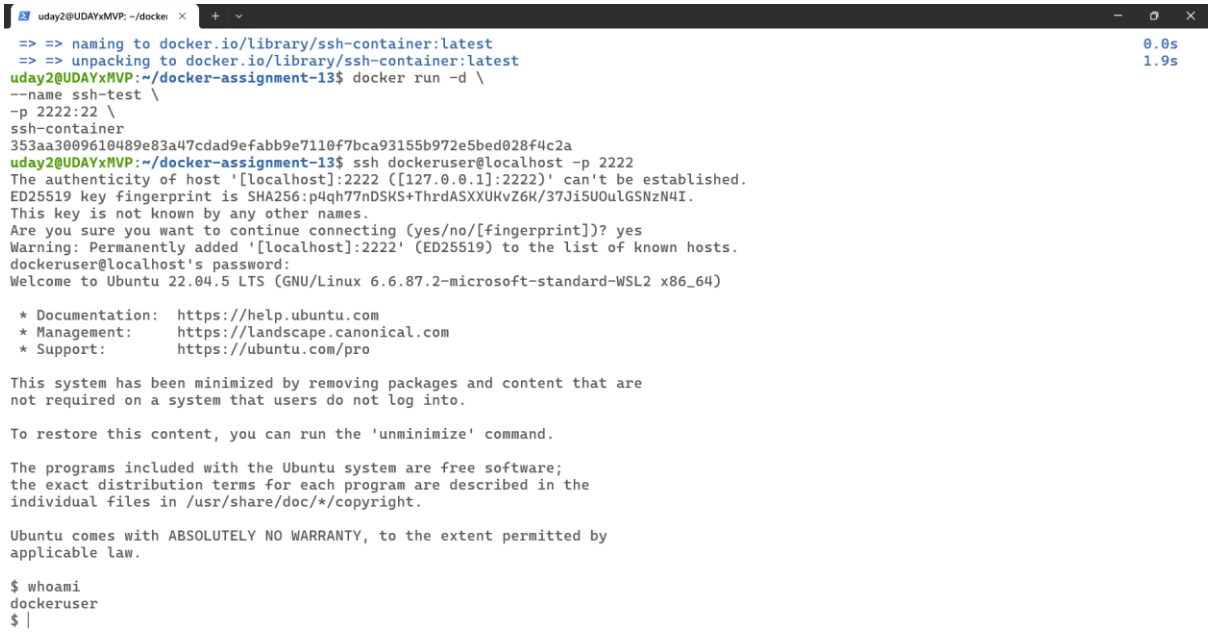
What to do:

1. Create a Dockerfile that:
    - Installs SSH server
    - Sets up SSH to allow passwords
    - Creates a user with a password
    - Opens port 22
2. Build and run it
3. SSH into your running container
4. Talk about why SSH in containers can be risky

Show me: Your Dockerfile, proof of SSH working, and your security thoughts

ANSWER:





Why SSH Into Containers Is Risky

- Containers are meant to run a **single process**, not act like full servers
- SSH increases the **attack surface** by opening extra ports
- Password-based SSH is vulnerable to brute-force attacks
- Best practice is to use docker exec instead of SSH

SSH in containers is risky because it increases security exposure and goes against container best practices.

Assignment 14: Control Container Resources

What you'll learn: Setting limits so containers don't hog everything

What to do:

1. Run a container with these limits:
    - Max 512MB of memory
    - Half a CPU core
    - At least 100MB memory reserved
2. Watch it with docker stats
3. Make a docker-compose.yml with limits for several services
4. Show what happens when a container hits its limits

Show me: Your commands and monitoring screenshots

```
uday2@UDAYxMVP:~/docker-assignment-14$ rm -rf docker-assignment-14
uday2@UDAYxMVP:~/docker-assignment-14$ mkdir docker-assignment-14
uday2@UDAYxMVP:~/docker-assignment-14$ cd docker-assignment-14
uday2@UDAYxMVP:~/docker-assignment-14/docker-assignment-14$ docker run -d \
--name limited-container \
--memory=512m \
--memory-reservation=100m \
--cpus="0.5" \
alexeiled/stress-ng \
--vm 1 --vm-bytes 600M --cpu 1
2e31a3347082a74d02ecf4a7b6e8f88226a985eac3cd44d5ca5fe308c7047952
uday2@UDAYxMVP:~/docker-assignment-14/docker-assignment-14$
```

```
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT   MEM %     NET I/O         BLOCK I/O         PIDS
2e31a3347082   limited-container   50.34%    511.9MiB / 512MiB   99.98%    1.17kB / 126B   1.29GB / 1.39GB   4
```

```
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT   MEM %     NET I/O   BLOCK I/O   PIDS
2e31a3347082   limited-container   --        -- / --             --        --        --          --

uday2@UDAYxMVP:~/docker-assignment-14/docker-assignment-14$ docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED              STATUS              PORTS     NAMES
2e31a3347082   alexeiled/stress-ng   "/stress-ng --vm 1 -…"   About a minute ago   Up About a minute             limited-container
uday2@UDAYxMVP:~/docker-assignment-14/docker-assignment-14$
```

```
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT   MEM %     NET I/O         BLOCK I/O         PIDS
2e31a3347082   limited-container   50.34%    511.9MiB / 512MiB   99.98%    1.17kB / 126B   1.29GB / 1.39GB   4
```

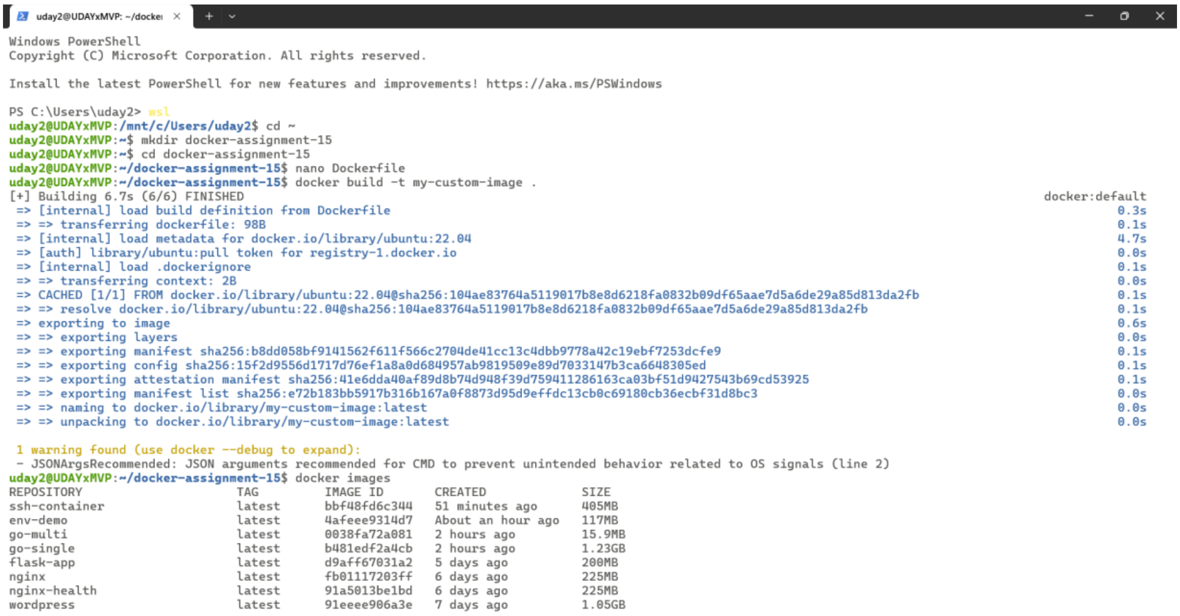
Assignment 15: Share Your Images Online

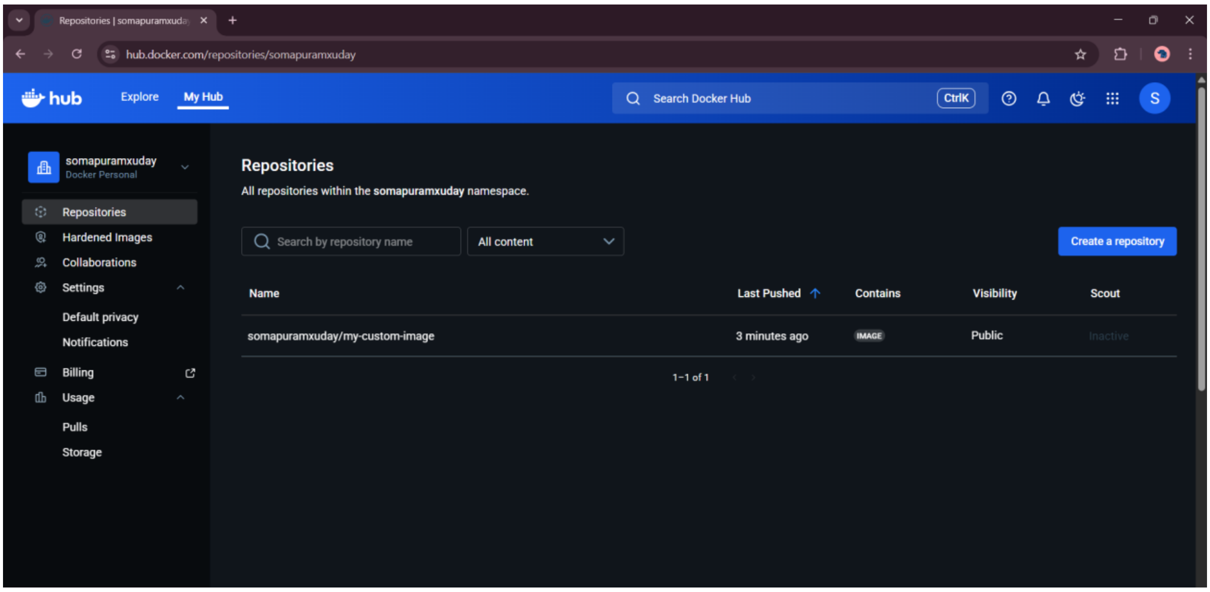
What you'll learn: Using Docker Hub to share images

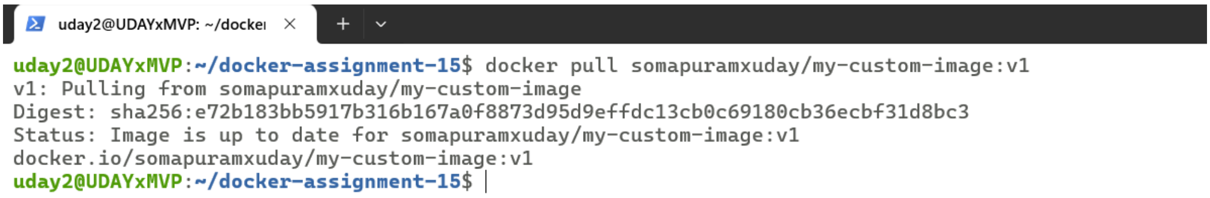
What to do:

1. Create a custom image (can be simple)
2. Tag it properly for Docker Hub
3. Push it to Docker Hub (make a free account first)
4. Delete your local copy
5. Pull it back down from Docker Hub
6. Document everything you did

Show me: Step-by-step documentation and your Docker Hub image link







Assignment 16: Fix Broken Containers

What you'll learn: Debugging when things go wrong

What to do:

1. Make a container that fails on purpose
2. Use docker logs to see what went wrong
3. Use docker exec to poke around inside a running container

4. Use docker inspect to see all the container details
5. Fix the problems and write up what you did

Show me: The problem, how you found it, and how you fixed it

Answer:

The Problem

The container failed because the Python app referenced an undefined variable.

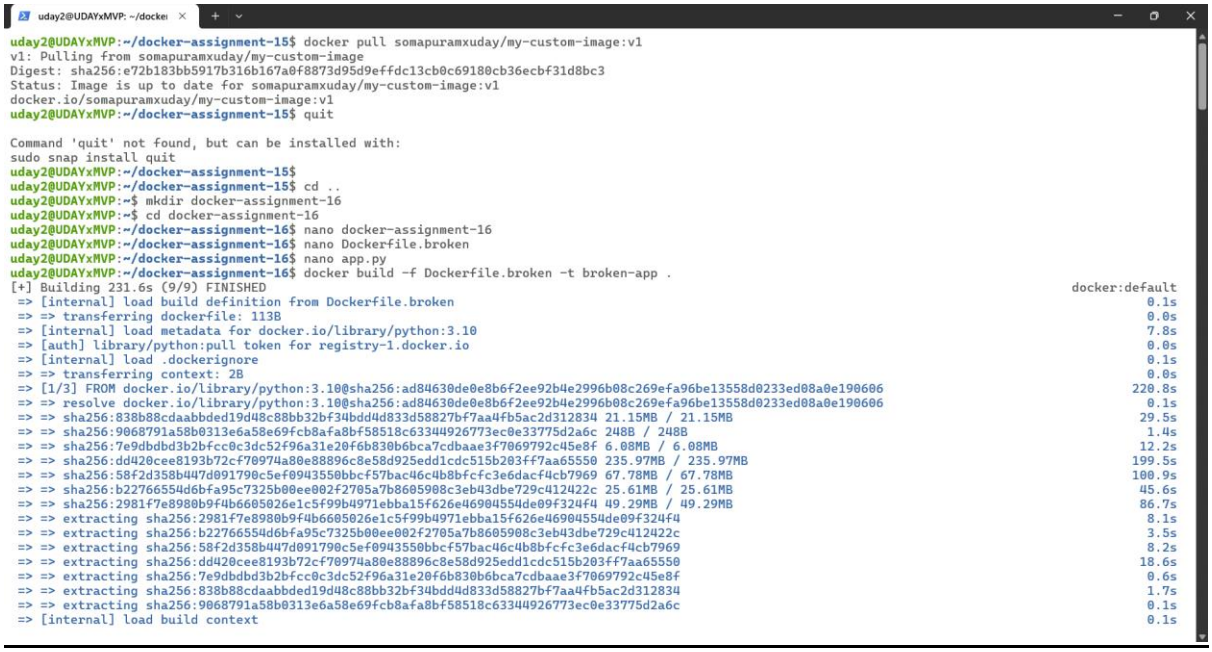
How I Found It

I used docker ps -a to see the container exited and docker logs to read the error message.

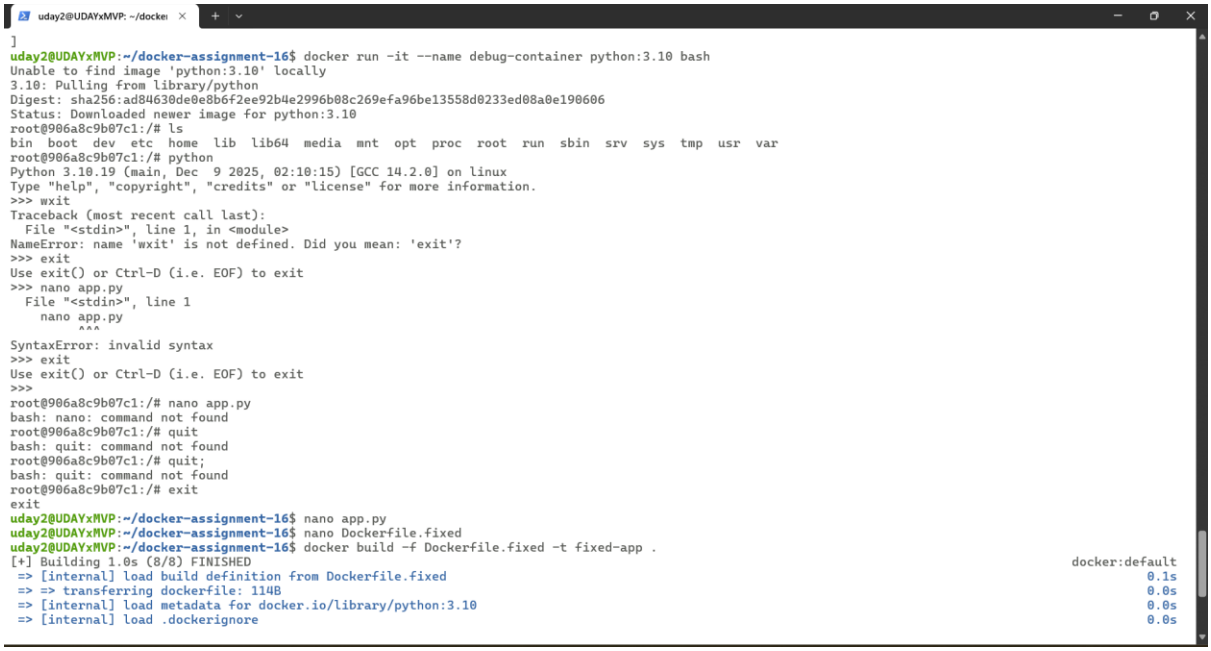
How I Fixed It

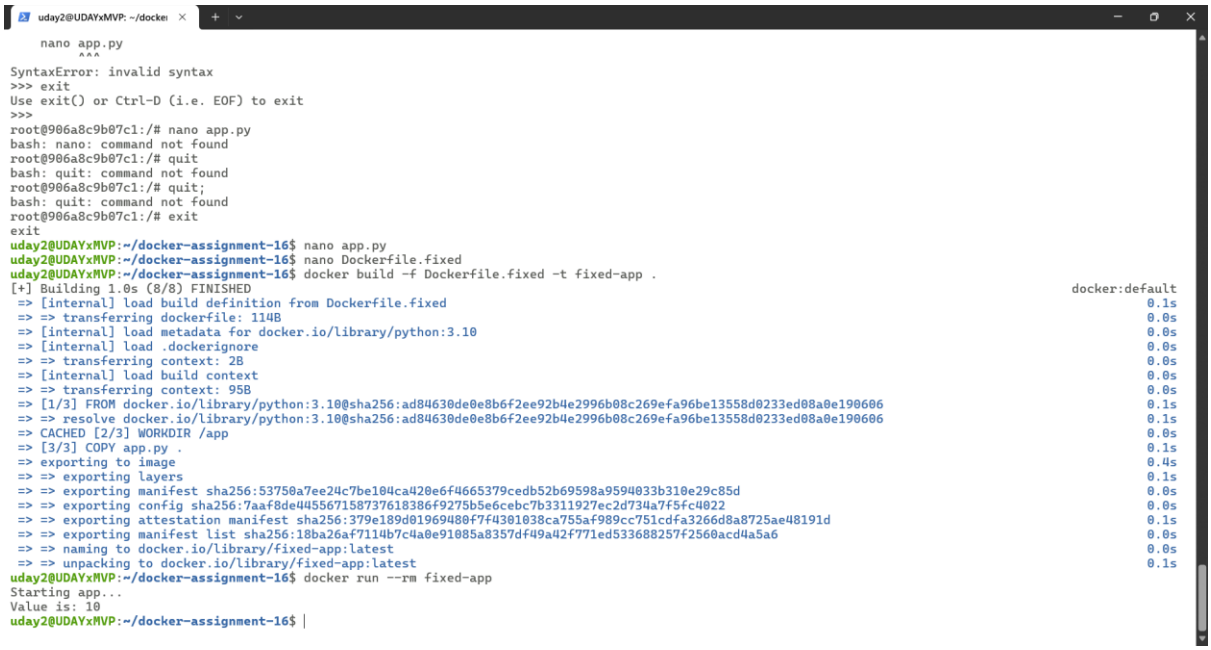
I corrected the code, rebuilt the image, and ran the container successfully.

Docker logs and inspect help identify why a container failed, and rebuilding after fixing the issue resolves it.


uday2@UDAYxMVP: ~/docker
uday2@UDAYxMVP:~/docker-assignment-15$ docker pull somapuramxuday/my-custom-image:v1
v1: Pulling from somapuramxuday/my-custom-image
Digest: sha256:e72b183bb5917b316b167a0f8873d95d9effdc13cb0c69180cb36ecbf31d8bc3
Status: Image is up to date for somapuramxuday/my-custom-image:v1
docker.io/somapuramxuday/my-custom-image:v1
uday2@UDAYxMVP:~/docker-assignment-15$ quit
Command 'quit' not found, but can be installed with:
sudo snap install quit
uday2@UDAYxMVP:~/docker-assignment-15$
uday2@UDAYxMVP:~/docker-assignment-15$ cd ..
uday2@UDAYxMVP:~$ mkdir docker-assignment-16
uday2@UDAYxMVP:~$ cd docker-assignment-16
uday2@UDAYxMVP:~/docker-assignment-16$ nano docker-assignment-16
uday2@UDAYxMVP:~/docker-assignment-16$ nano Dockerfile.broken
uday2@UDAYxMVP:~/docker-assignment-16$ nano app.py
uday2@UDAYxMVP:~/docker-assignment-16$ docker build -f Dockerfile.broken -t broken-app .
[+] Building 231.6s (9/9) FINISHED docker:default
=> [internal] load build definition from Dockerfile.broken 0.1s
=> => transferring dockerfile: 113B 0.0s
=> [internal] load metadata for docker.io/library/python:3.10 7.8s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> [1/3] FROM docker.io/library/python:3.10@sha256:ad84630de0e8b6f2ee92b4e2996b08c269efa96be13558d0233ed08a0e190606 220.8s
=> => resolve docker.io/library/python:3.10@sha256:ad84630de0e8b6f2ee92b4e2996b08c269efa96be13558d0233ed08a0e190606 0.1s
=> => sha256:838b88cdaabbded19d48c88bb32bf34bdd4d833d58827bf7aa4fb5ac2d312834 21.15MB / 21.15MB 29.5s
=> => sha256:9068791a58b0313e6a58e69fcb8afa8bf58518c63344926773ec0e33775d2a6c 248B / 248B 1.4s
=> => sha256:7e9dbdbd3b2bfcc0c3dc52f96a31e20f6b830b6bca7cdbaae3f7069792c45e8f 6.08MB / 6.08MB 12.2s
=> => sha256:dd420cee8193b72cf70974a80e88896c8e58d925edd1cdc515b203ff7aa65550 235.97MB / 235.97MB 199.5s
=> => sha256:58f2d358b447d091790c5ef0943550bbcf57bac46c4b8bfcfc3e6dacf4cb7969 67.78MB / 67.78MB 100.9s
=> => sha256:b22766554d6bfa95c7325b00ee002f2705a7b8605908c3eb43dbe729c412422c 25.61MB / 25.61MB 45.6s
=> => sha256:2981f7e8980b9f4b6605026e1c5f99b4971ebba15f626e46904554de09f324f4 49.29MB / 49.29MB 86.7s
=> => extracting sha256:2981f7e8980b9f4b6605026e1c5f99b4971ebba15f626e46904554de09f324f4 8.1s
=> => extracting sha256:b22766554d6bfa95c7325b00ee002f2705a7b8605908c3eb43dbe729c412422c 3.5s
=> => extracting sha256:58f2d358b447d091790c5ef0943550bbcf57bac46c4b8bfcfc3e6dacf4cb7969 8.2s
=> => extracting sha256:dd420cee8193b72cf70974a80e88896c8e58d925edd1cdc515b203ff7aa65550 18.6s
=> => extracting sha256:7e9dbdbd3b2bfcc0c3dc52f96a31e20f6b830b6bca7cdbaae3f7069792c45e8f 0.6s
=> => extracting sha256:838b88cdaabbded19d48c88bb32bf34bdd4d833d58827bf7aa4fb5ac2d312834 1.7s
=> => extracting sha256:9068791a58b0313e6a58e69fcb8afa8bf58518c63344926773ec0e33775d2a6c 0.1s
=> [internal] load build context 0.1s


uday2@UDAYxMVP: ~/docker
=> => unpacking to docker.io/library/broken-app:latest 0.1s
uday2@UDAYxMVP:~/docker-assignment-16$ docker run --name broken-container broken-app
Starting app...
Traceback (most recent call last):
File "/app/app.py", line 2, in <module>
print(undefined_variable) # this will cause error
NameError: name 'undefined_variable' is not defined
uday2@UDAYxMVP:~/docker-assignment-16$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
823d7a08fa5a broken-app "python app.py" 9 seconds ago Exited (1) 6 seconds ago broken-container
1002a9ccad8c nginx "/docker-entrypoint.…" 42 minutes ago Exited (0) 41 minutes ago docker-assignment-14-web-1
ed465f58801a alexeiled/stress-ng "/stress-ng --vm 1 -…" 42 minutes ago Exited (0) 41 minutes ago docker-assignment-14-stress-1
2e31a3347082 alexeiled/stress-ng "/stress-ng --vm 1 -…" 48 minutes ago Up 48 minutes limited-container
uday2@UDAYxMVP:~/docker-assignment-16$ docker logs broken-container
Starting app...
Traceback (most recent call last):
File "/app/app.py", line 2, in <module>
print(undefined_variable) # this will cause error
NameError: name 'undefined_variable' is not defined
uday2@UDAYxMVP:~/docker-assignment-16$ docker inspect broken-container
[
{
"Id": "823d7a08fa5a73c47aa6b2efc37be06ac53f57c515e7273cbfc9d989201f792d",
"Created": "2025-12-16T12:17:24.486115431Z",
"Path": "python",
"Args": [
"app.py"
],
"State": {
"Status": "exited",
"Running": false,
"Paused": false,
"Restarting": false,
"OOMKilled": false,
"Dead": false,
"Pid": 0,
"ExitCode": 1,
"Error": "",
"StartedAt": "2025-12-16T12:17:24.899822993Z",
"FinishedAt": "2025-12-16T12:17:26.757094704Z"
},





Assignment 17: Build a Complete App Stack

What you'll learn: Orchestrating multiple services

What to do: Create a docker-compose.yml that runs:

1. A Node.js web app
2. Redis for caching
3. MongoDB for data
4. Nginx as a reverse proxy
5. Custom networks connecting them

6. Volumes for keeping data
7. Environment variables for config
8. Health checks for everything

Show me: Your complete compose file with notes explaining each part

```
uday2@UDAYxMVP:~/docker-assignment-17$ cat docker-compose.yml
version: "3.8"

services:
  app:
    image: node:18
    working_dir: /app
    volumes:
      - ./app:/app
    command: node index.js
    environment:
      - REDIS_HOST=redis
      - MONGO_URL=mongodb://mongo:27017/mydb
    networks:
      - app-network
    depends_on:
      - mongo
      - redis
    healthcheck:
      test: ["CMD", "node", "-e", "process.exit(0)"]
      interval: 10s
      retries: 3

  redis:
    image: redis:7
    networks:
      - app-network
    healthcheck:
      test: ["CMD", "redis-cli", "ping"]
      interval: 10s
      retries: 3

  mongo:
    image: mongo:6
    volumes:
      - mongo-data:/data/db
    networks:
      - app-network
    healthcheck:
      test: ["CMD", "mongosh", "--eval", "db.runCommand({ ping: 1 })"]
      interval: 10s
      retries: 3

  nginx:
    image: nginx:latest
    ports:
      - "8080:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - app
    networks:
      - app-network
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost"]
      interval: 10s
      retries: 3

volumes:
  mongo-data:

networks:
  app-network:
uday2@UDAYxMVP:~/docker-assignment-17$
```


```
Hello from Node App
```

1. Docker Compose orchestrates multiple services like Node.js, Redis, MongoDB, and Nginx.
2. Custom networks allow service communication, volumes provide data persistence, environment variables handle configuration, and health checks monitor service status.

Docker Compose manages a complete application stack using a single configuration file.

Assignment 18: Try Different Network Types

What you'll learn: When to use which network mode

What to do:

1. Test containers with:
    - Bridge network (the normal one)
    - Host network (shares your computer's network)

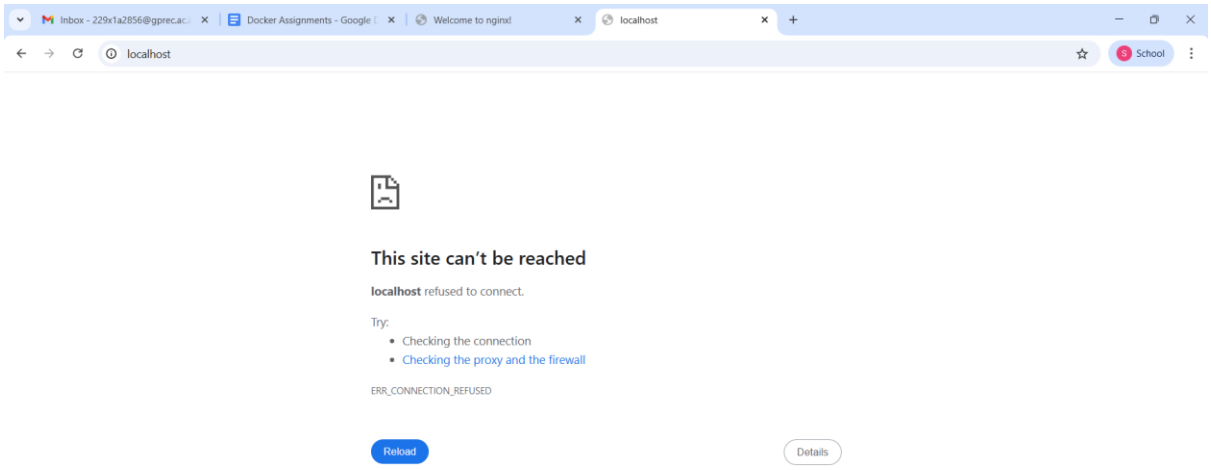
- None network (totally isolated)

2. Document what's different:
    - How isolated are they?
    - Do you need port mapping?
    - Which is faster?
3. Explain when you'd use each type

Show me: A comparison table and real-world examples

Answer:

```
uday2@UDAYxMVP: ~/assign
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\uday2> wsl
uday2@UDAYxMVP:/mnt/c/Users/uday2$ cd ~
uday2@UDAYxMVP:~$ rm -rf assignment-18-docker-network
uday2@UDAYxMVP:~$ mkdir assignment-18-docker
uday2@UDAYxMVP:~$ cd assignment-18-docker
uday2@UDAYxMVP:~/assignment-18-docker$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:fb01117203ff38c2f9af91db1a7409459182a37c87cced5cb442d1d8fcc66d19
Status: Image is up to date for nginx:latest
docker.io/library/nginx:latest
uday2@UDAYxMVP:~/assignment-18-docker$ docker run -d --name bridge-test -p 8080:80 nginx
docker: Error response from daemon: Conflict. The container name "/bridge-test" is already in use by container "d98feb7a37a69a89204f872f376cb80e08ae6dfce2add02e2bc96c38eb
5d8d9f". You have to remove (or rename) that container to be able to reuse that name.

Run 'docker run --help' for more information
uday2@UDAYxMVP:~/assignment-18-docker$ docker ps -a
CONTAINER ID   IMAGE                 COMMAND                  CREATED        STATUS                          PORTS                  NAMES
eb8488d7ca9e   nginx                 "/docker-entrypoint.…"   15 hours ago   Exited (255) About a minute ago                        host-test
d98feb7a37a6   nginx                 "/docker-entrypoint.…"   16 hours ago   Exited (255) About a minute ago   0.0.0.0:8088->80/tcp   bridge-test
b77cb22b8533   nginx:latest          "/docker-entrypoint.…"   16 hours ago   Exited (255) About a minute ago   0.0.0.0:8080->80/tcp   docker-assignment-17-nginx-1
9ca11bcbbd70   node:18               "docker-entrypoint.s…"   16 hours ago   Exited (255) About a minute ago                          docker-assignment-17-app-1
958bbadf4e53   redis:7               "docker-entrypoint.s…"   16 hours ago   Exited (255) About a minute ago   6379/tcp               docker-assignment-17-redis-1
d65d71b9e253   mongo:6               "docker-entrypoint.s…"   16 hours ago   Exited (255) About a minute ago   27017/tcp              docker-assignment-17-mongo-1
906a8c9b07c1   python:3.10           "bash"                   16 hours ago   Exited (127) 16 hours ago                                debug-container
823d7a08fa5a   broken-app            "python app.py"          16 hours ago   Exited (1) 16 hours ago                                  broken-container
1002a9ccad8c   nginx                 "/docker-entrypoint.…"   17 hours ago   Exited (0) 17 hours ago                                  docker-assignment-14-web-1
ed465f58801a   alexeiled/stress-ng   "/stress-ng --vm 1 -…"   17 hours ago   Exited (0) 17 hours ago                                  docker-assignment-14-stress-1
2e31a3347082   alexeiled/stress-ng   "/stress-ng --vm 1 -…"   17 hours ago   Exited (255) About a minute ago                          limited-container
uday2@UDAYxMVP:~/assignment-18-docker$ docker stop bridge-test
bridge-test
uday2@UDAYxMVP:~/assignment-18-docker$ docker rm bridge-test
bridge-test
uday2@UDAYxMVP:~/assignment-18-docker$ docker run -d --name bridge-test -p 8080:80 nginx
80a232b1f8e4dddc0874d5057661ffc900cd490a1fc6bb5580fcc9056e26d4bd
uday2@UDAYxMVP:~/assignment-18-docker$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                      NAMES
80a232b1f8e4   nginx     "/docker-entrypoint.…"   4 seconds ago   Up 3 seconds   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   bridge-test
uday2@UDAYxMVP:~/assignment-18-docker$ docker rm -f bridge-test
bridge-test
```



Not Worked for second case I was using Docker Desktop + WSL (But the command was working can be seen in snippet)

Observation (Host Network)

On Windows using Docker Desktop with WSL, containers running in host network mode are attached to the Linux VM's network, not the Windows host network. Therefore, accessing the container via localhost from Windows browser is not possible. However, absence of port mapping confirms host networking behavior.

```
uday2@UDAYxMVP:~/assignment-18-docker$ docker run -d --name host-test --network host nginx
docker: Error response from daemon: Conflict. The container name "/host-test" is already in use by container "eb8488d7ca9eba04cbae682f8ccdf92346697639f9b3d53e07a739f405a3
22ce". You have to remove (or rename) that container to be able to reuse that name.

Run 'docker run --help' for more information
uday2@UDAYxMVP:~/assignment-18-docker$ docker rm -f host-test
host-test
uday2@UDAYxMVP:~/assignment-18-docker$ docker run -d --name host-test --network host nginx
9b06109aca0fb98e512d804e131af288fc6438683a0f84334c79cbfe04bc5477
uday2@UDAYxMVP:~/assignment-18-docker$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
9b06109aca0f   nginx     "/docker-entrypoint.…"   6 seconds ago   Up 5 seconds             host-test
uday2@UDAYxMVP:~/assignment-18-docker$ docker rm -f host-test
host-test
uday2@UDAYxMVP:~/assignment-18-docker$ docker run -d --name none-test --network none nginx
71e6f8959e04f7f70b447668895745f7fdec1bc25f2751e7d165303a218ca26e
uday2@UDAYxMVP:~/assignment-18-docker$ docker inspect none-test | grep -i network -A 5
            "NetworkMode": "none",
            "PortBindings": {},
            "RestartPolicy": {
                "Name": "no",
                "MaximumRetryCount": 0
            },
--
        "NetworkSettings": {
            "Bridge": "",
            "SandboxID": "4b699e0e7bad20c9a79acaca5848cad527279a6bf3a70ded68000b5ed03e2818",
            "SandboxKey": "/var/run/docker/netns/4b699e0e7bad",
            "Ports": {},
            "HairpinMode": false,
--
            "Networks": {
                "none": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "MacAddress": "",
--
                    "NetworkID": "56ea782f6bcdae4f5b2d51a6af95062cb6513b8d922141ad99a19b846dbaa666",
                    "EndpointID": "ad78d524e6cb3ccd5429dfb96e5cc4686f7a8761809830677e44fadf072766b8",
                    "Gateway": "",
                    "IPAddress": "",
                    "IPPrefixLen": 0,
                    "IPv6Gateway": ""
uday2@UDAYxMVP:~/assignment-18-docker$
```

Notes:

Why the error occurred:

Docker container names must be unique. Even stopped containers reserve names until removed.

How it was resolved:

Existing containers were stopped and removed using docker rm -f.

Bridge Network (Default)

- Isolation: Medium (containers have private IP)
- Port mapping: Required (-p host_port:container_port)
- Speed: Moderate
- Internet access: Yes

Use when:

- Normal applications
- Multiple containers communicating
- Development and testing

Example:

- Web app with database
- Backend API with Redis

Host Network

- Isolation: None (shares host network)
- Port mapping: Not required
- Speed: Fastest
- Internet access: Yes

Use when:

- High performance required
- Low latency networking

Example:

- Monitoring tools
- Game servers

Note:

- On Windows with Docker Desktop, browser access via localhost may not work

None Network

- Isolation: Full (no network interface)
- Port mapping: Not possible
- Speed: Not applicable
- Internet access: No

Use when:

- Maximum security required
- Offline processing

Example:

- Batch jobs
- Encryption or file processing containers

Comparison Table

Network	Isolation	Port Mapping	Speed	Use Case
Bridge	Medium	Yes	Moderate	Normal apps
Host	None	No	Fastest	Performance-critical apps
None	Full	No	N/A	Secure or offline tasks

Conclusion:

Bridge is suitable for most applications, Host provides maximum performance with no isolation, and None is used for complete network isolation.

Assignment 19: Make Flexible Build Configs

What you'll learn: Using build arguments for different builds

What to do:

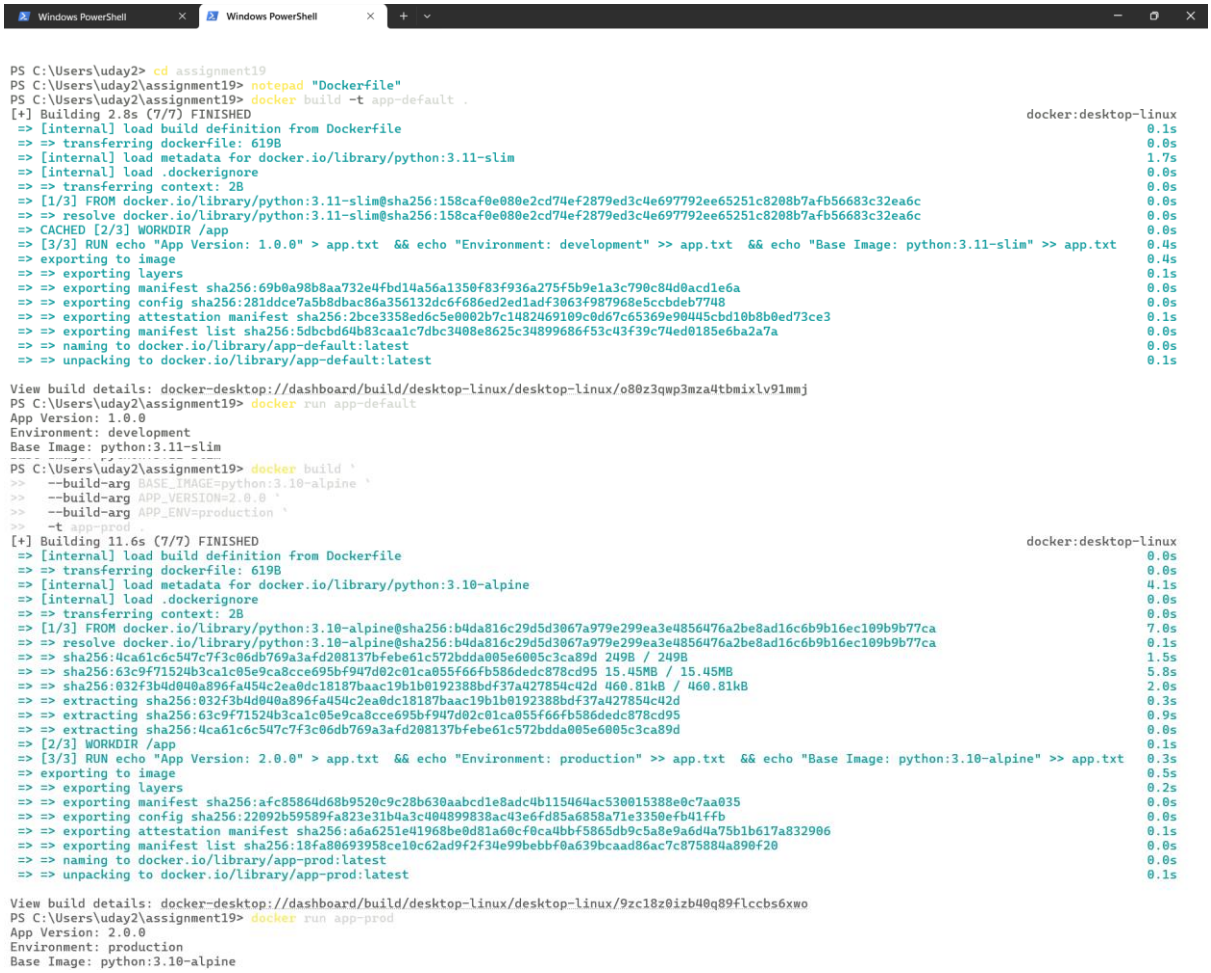
1. Write a Dockerfile that accepts build arguments
2. Use arguments to:
    - Pick different base images
    - Pass in config values at build time

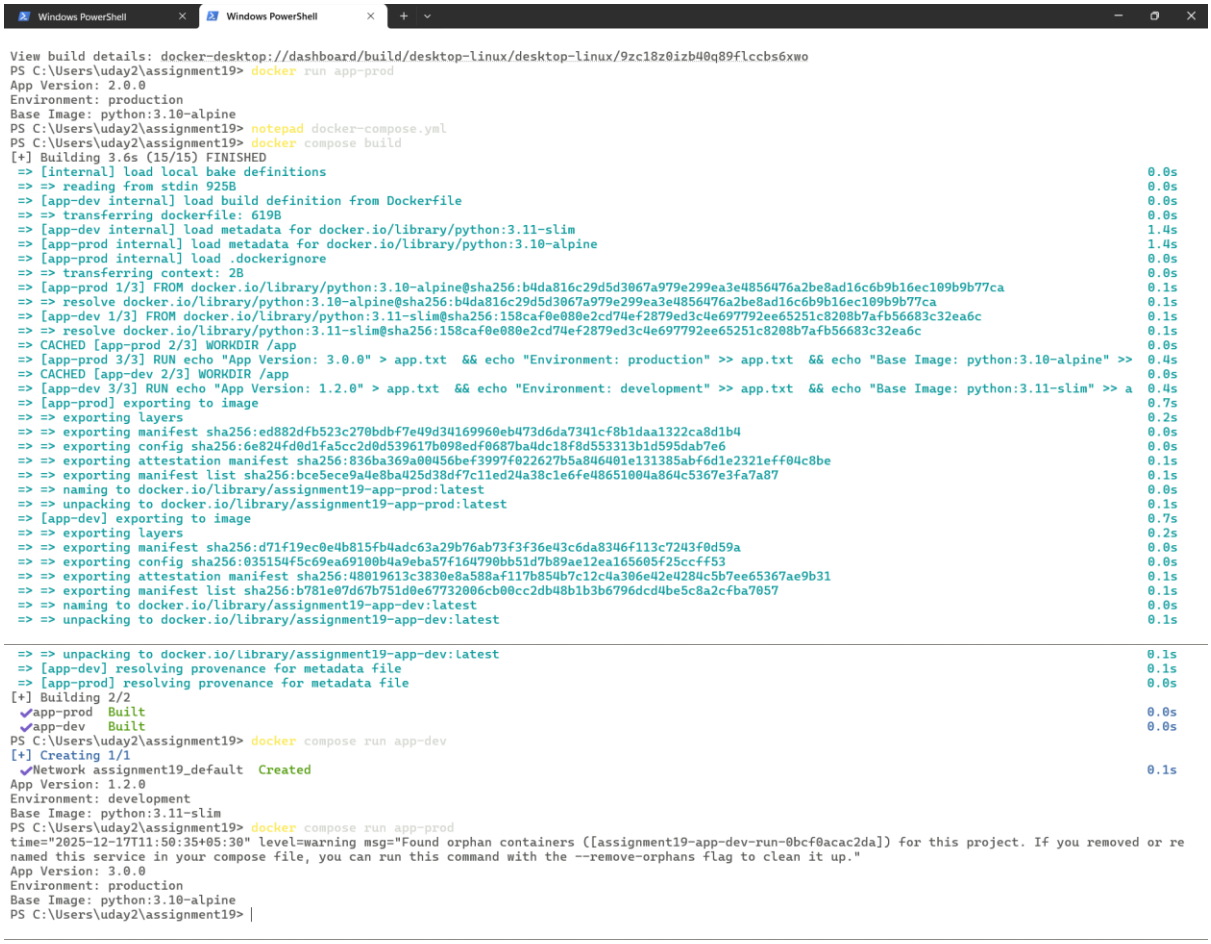
- Set the app version

3. Build the same Dockerfile multiple ways with different arguments
4. Add build arguments to a docker-compose.yml too

Show me: Your Dockerfile with ARG instructions and different builds

Answer:





Assignment 20: Deploy a Full Application

What you'll learn: Everything together in one project!

What to do: Build a complete three-tier app:

1. **Frontend**: React or Angular app
2. **Backend**: An API (Node.js, Python, or Java - your choice)
3. **Database**: PostgreSQL or MongoDB

You need:

- Dockerfiles for each part
- A docker-compose.yml that runs everything
- Volumes so data sticks around
- Networks connecting the pieces
- Environment variables for settings
- Health checks
- Resource limits

- Full documentation
- Step-by-step setup instructions

Show me: Everything - all files, a README, an architecture diagram, and proof it works (screenshots or video)

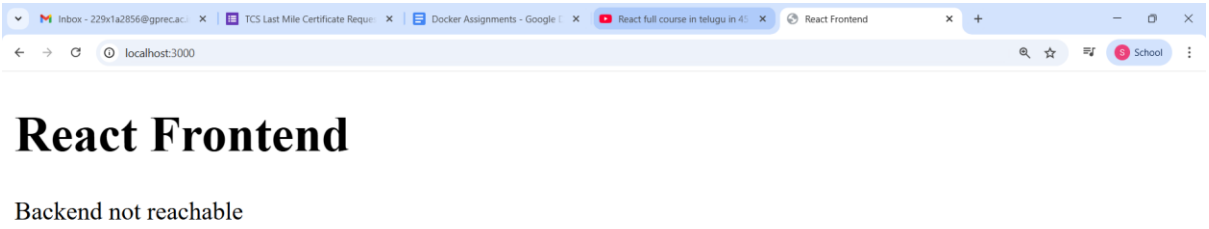
Answer:

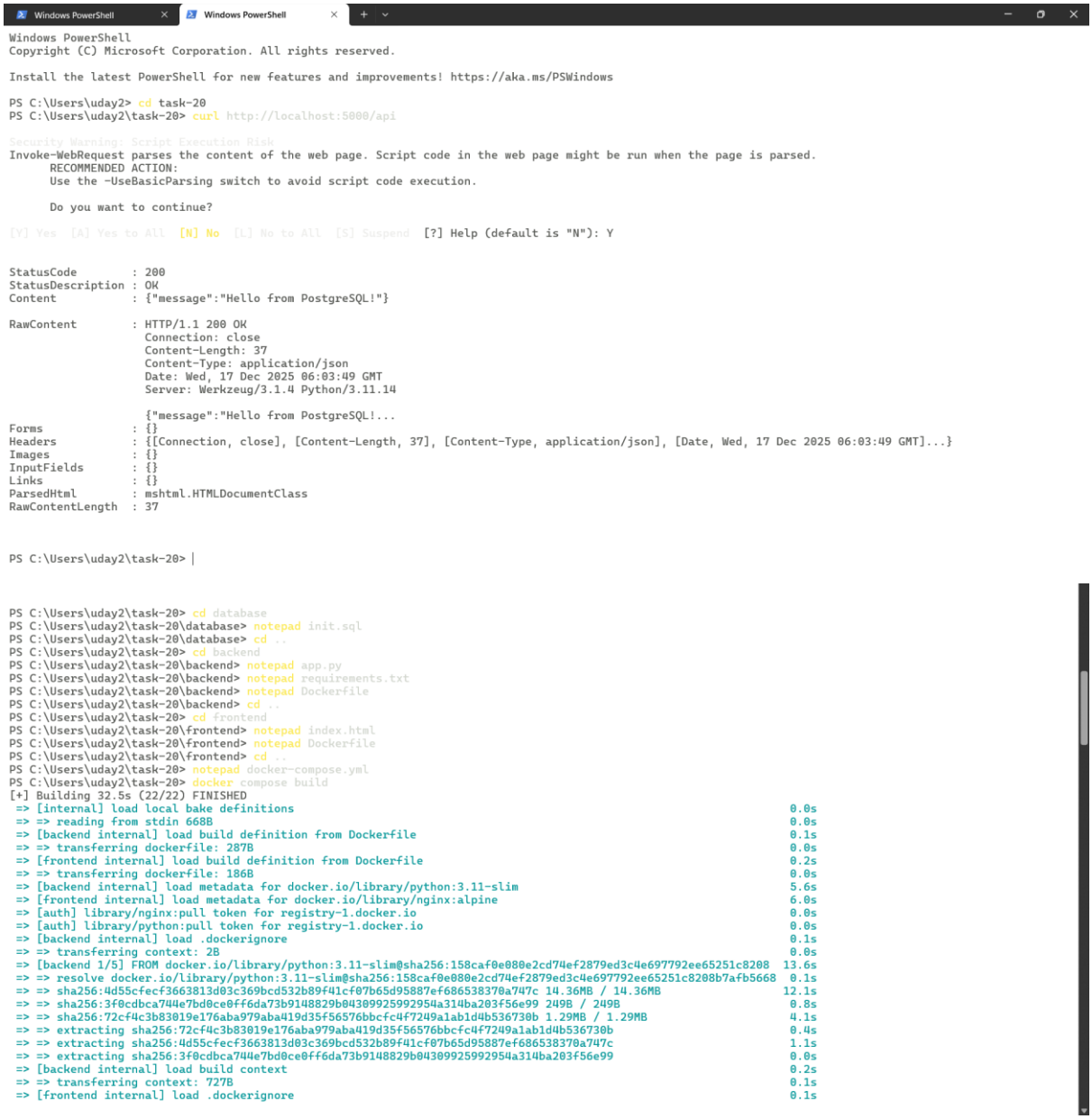
Choosing:

1. **Frontend**: React
2. **Backend**: An API (Python → Flask)
3. **Database**: PostgreSQL

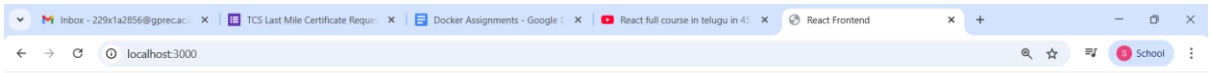
Architecture:

Browser → Frontend (Nginx) → Backend (Flask API) → PostgreSQL (Volume: pgdata)




Windows PowerShell
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\uday2> cd task-20
PS C:\Users\uday2\task-20> curl http://localhost:5000/api
Security Warning: Script Execution Risk
Invoke-WebRequest parses the content of the web page. Script code in the web page might be run when the page is parsed.
RECOMMENDED ACTION:
Use the -UseBasicParsing switch to avoid script code execution.
Do you want to continue?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
StatusCode : 200
StatusDescription : OK
Content : {"message":"Hello from PostgreSQL!"}
RawContent : HTTP/1.1 200 OK
Connection: close
Content-Length: 37
Content-Type: application/json
Date: Wed, 17 Dec 2025 06:03:49 GMT
Server: Werkzeug/3.1.4 Python/3.11.14
{"message":"Hello from PostgreSQL!...
Forms : {}
Headers : {[Connection, close], [Content-Length, 37], [Content-Type, application/json], [Date, Wed, 17 Dec 2025 06:03:49 GMT]...}
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 37
PS C:\Users\uday2\task-20>
PS C:\Users\uday2\task-20> cd database
PS C:\Users\uday2\task-20\database> notepad init.sql
PS C:\Users\uday2\task-20\database> cd ..
PS C:\Users\uday2\task-20> cd backend
PS C:\Users\uday2\task-20\backend> notepad app.py
PS C:\Users\uday2\task-20\backend> notepad requirements.txt
PS C:\Users\uday2\task-20\backend> notepad Dockerfile
PS C:\Users\uday2\task-20\backend> cd ..
PS C:\Users\uday2\task-20> cd frontend
PS C:\Users\uday2\task-20\frontend> notepad index.html
PS C:\Users\uday2\task-20\frontend> notepad Dockerfile
PS C:\Users\uday2\task-20\frontend> cd ..
PS C:\Users\uday2\task-20> notepad docker-compose.yml
PS C:\Users\uday2\task-20> docker compose build
[+] Building 32.5s (22/22) FINISHED
=> [internal] load local bake definitions 0.0s
=> => reading from stdin 668B 0.0s
=> [backend internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 287B 0.0s
=> [frontend internal] load build definition from Dockerfile 0.2s
=> => transferring dockerfile: 186B 0.0s
=> [backend internal] load metadata for docker.io/library/python:3.11-slim 5.6s
=> [frontend internal] load metadata for docker.io/library/nginx:alpine 6.0s
=> [auth] library/nginx:pull token for registry-1.docker.io 0.0s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [backend internal] load .dockerignore 0.1s
=> => transferring context: 2B 0.0s
=> [backend 1/5] FROM docker.io/library/python:3.11-slim@sha256:158caf0e080e2cd74ef2879ed3c4e697792ee65251c8208 13.6s
=> => resolve docker.io/library/python:3.11-slim@sha256:158caf0e080e2cd74ef2879ed3c4e697792ee65251c8208b7afb5668 0.1s
=> => sha256:4d55cfecf3663813d03c369bcd532b89f41cf07b65d95887ef686538370a747c 14.36MB / 14.36MB 12.1s
=> => sha256:3f0cdbca744e7bd0ce0ff6da73b9148829b04309925992954a314ba203f56e99 249B / 249B 0.8s
=> => sha256:72cf4c3b83019e176aba979aba419d35f56576bbcfc4f7249a1ab1d4b536730b 1.29MB / 1.29MB 4.1s
=> => extracting sha256:72cf4c3b83019e176aba979aba419d35f56576bbcfc4f7249a1ab1d4b536730b 0.4s
=> => extracting sha256:4d55cfecf3663813d03c369bcd532b89f41cf07b65d95887ef686538370a747c 1.1s
=> => extracting sha256:3f0cdbca744e7bd0ce0ff6da73b9148829b04309925992954a314ba203f56e99 0.0s
=> [backend internal] load build context 0.2s
=> => transferring context: 727B 0.1s
=> [frontend internal] load .dockerignore 0.1s

Finally,



React Frontend

Hello from PostgreSQL!

The reason for using /api instead of local host!

The browser cannot resolve Docker service names. I used Nginx as a reverse proxy so the frontend communicates with the backend through the Docker network internally, which is the correct production approach