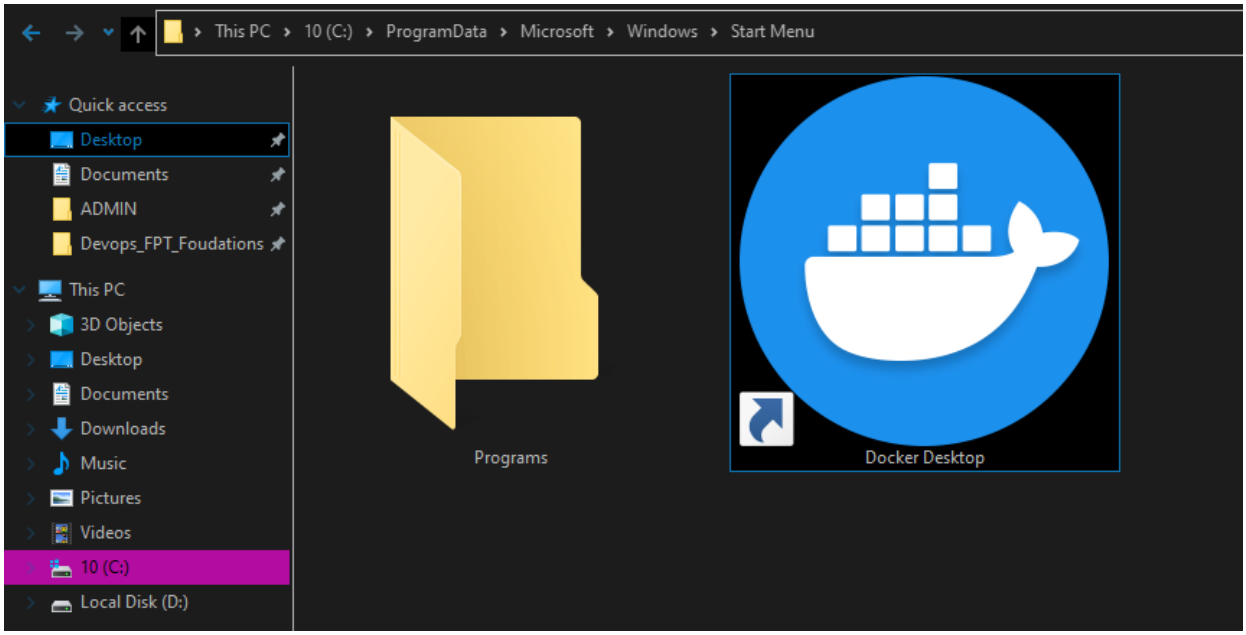
Nguyễn Thái Dương Day 1 docker install exerise:

1. Install Docker on your local machine.
2. Create a DockerHub account.

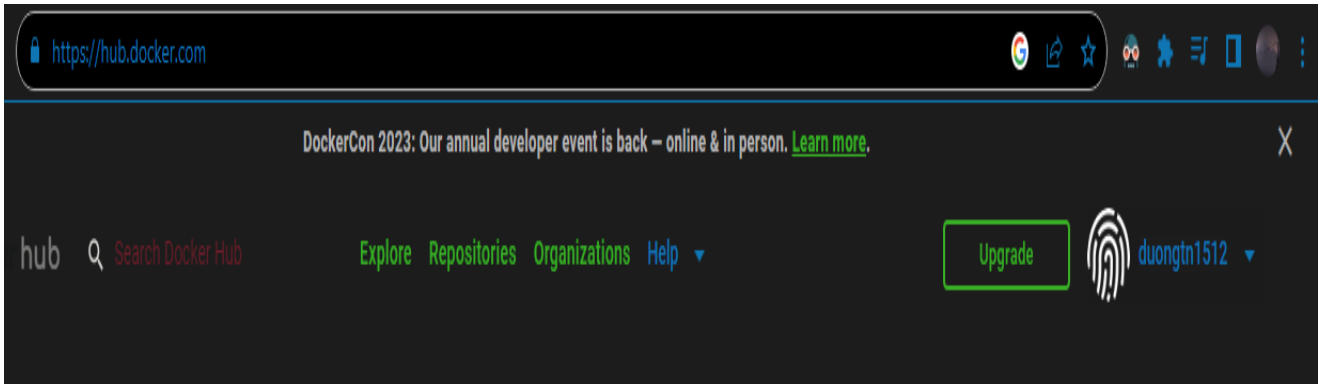
Task1: Install Docker on your local machine.




```
PS D:\Devops_FPT_Foudations> docker system df
TYPE            TOTAL     ACTIVE    SIZE      RECLAIMABLE
Images          3         3         234.1MB   186.9MB (79%)
Containers      3         0         2.415kB   2.415kB (100%)
Local Volumes   0         0         0B        0B
Build Cache     11        0         89B       89B
PS D:\Devops_FPT_Foudations> docker --version
Docker version 24.0.5, build ced0996
```

Task2: Create a DockerHub account.

```
PS D:\Devops_FPT_Foudations> docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
PS D:\Devops_FPT_Foudations>
```



Nguyễn Thái Dương Day 2 docker image exerise:

1. Create a Dockerfile for a Python application that prints a message. Build the image and run a container from it.

2. Build an image with multiple tags, such as "latest" and a specific version. Push the image to Docker Hub.

3. Create a multi-stage Dockerfile for a Node.js app. Build the app and copy only the necessary files to the final image.

4. Push an image you've created to your Docker Hub repository. Then, pull it to a different machine and run a container.

5. From an official Nginx image, build a custom Nginx image to output a customized homepage.

6. Run an Nginx container in detached mode, map a local port to the container's port, and access the Nginx welcome page.

7. Run an Ubuntu container interactively, access its shell, and execute basic commands.

8. Run a container with environment variables set using the -e flag and `ENV` instruction. Access these variables from within the container.

9. Build a Docker image using the Dockerfile in the `issue` directory and run a container using that image. What happened with the container? Troubleshoot so the container is able to run.

10. Run a container, stop it, and then remove it.

11. Build an image with a custom HTML file. Run a container from the image and copy a file from the host to the container.

Task 1: Create a Dockerfile for a Python application that prints a message.

Build the image and run a container from it.

Task 2: Build an image with multiple tags and push it to Docker Hub.

(I do fist 2 task in 1 run)

Task1.Dockerfile U × | task1.py U | Reslove.txt M | task1.ps1 U

docker > session-02 > Task1 > Task1.Dockerfile > ...

```
# Create a Dockerfile for a Python application that prints a message.
# Build the image and run a container from it.

# Use the official Python image with version 3.9
FROM python:3.9
# Set the working diretory for file store in app file
WORKDIR /app
# Copy the exiten python script to the diretory in image
COPY task1.py .
# Run the python script that we just Copy
CMD ["python", "task1.py"]
```

Task1.Dockerfile U | task1.py U | Reslove.txt M | task1.ps1 U

docker > session-02 > Task1 > task1.py > ...

```python
print("Hello world this is a Duong making task 1 and 2 ")
print("In docker container it cant read input from terminal")
a = int(1500)
b = str("To the moon")
print(f"The number a is {a}")
print(f"Slowly to the {b}")
```

Powershell script to auto muti tag, build, push, run and login Docker

Task1.Dockerfile U | task1.py U | task1.ps1 U | Reslove.txt M

docker > session-02 > Task1 > task1.ps1

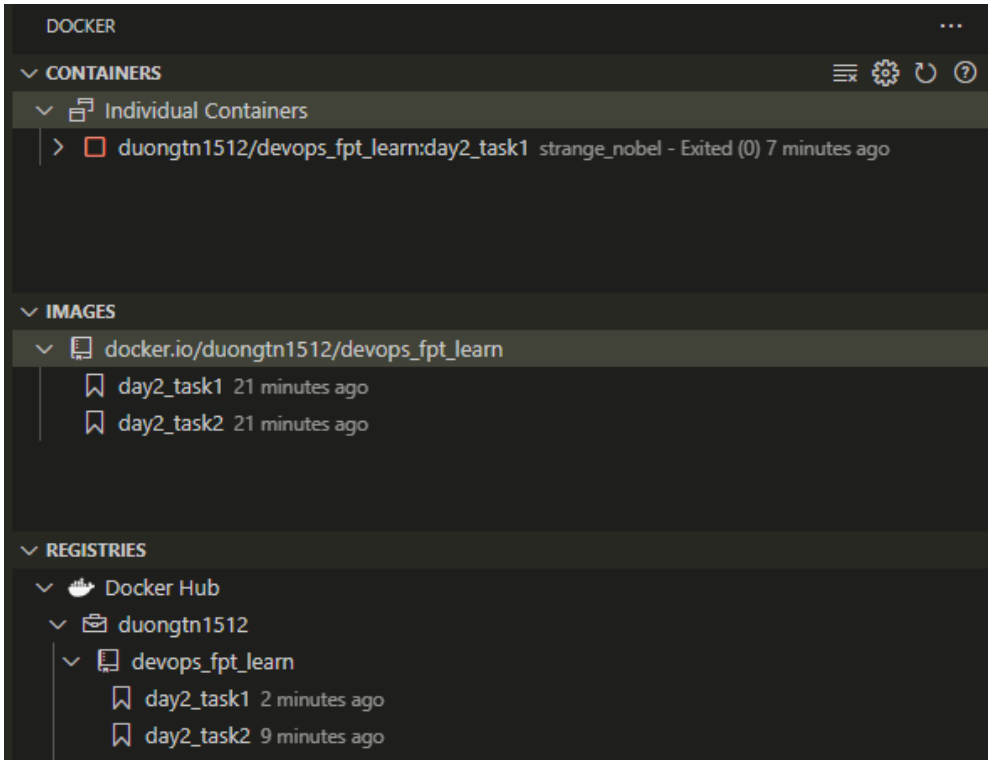
```python
# Build my application with tag my repo in docker hub and path in the file that have docker conten
docker build -t duongtn1512/devops_fpt_learn:day2_task1 -t duongtn1512/devops_fpt_learn:day2_task2 -f Task1.Dockerfile .
# Make sure we login in docker hub repo duongtn1512
docker login
# Push to our repo
docker push duongtn1512/devops_fpt_learn:day2_task1
docker push duongtn1512/devops_fpt_learn:day2_task2
# Run it again to make sure it pull from docker hub repo of our
docker run duongtn1512/devops_fpt_learn:day2_task1
```

Run the script and get result printed on terminal

```
PS D:\Devops_FPT_Foudations\docker\session-02\Task1> ./task1.ps1
[+] Building 8.9s (9/9) FINISHED
 => [internal] load build definition from Task1.Dockerfile
 => => transferring dockerfile: 449B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.9
 => [auth] library/python:pull token for registry-1.docker.io
 => [1/3] FROM docker.io/library/python:3.9@sha256:fdff20fe1b98766e020a4dd5ad4537e675
 => [internal] load build context
 => => transferring context: 30B
 => CACHED [2/3] WORKDIR /app
 => CACHED [3/3] COPY task1.py .
 => exporting to image
 => => exporting layers
 => => writing image sha256:3c40c7a5388e548a3cf0d116e15e4ef8fec219d2e87e4d7edc2b6f577
 => => naming to docker.io/duongtn1512/devops_fpt_learn:day2_task1
 => => naming to docker.io/duongtn1512/devops_fpt_learn:day2_task2

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
Authenticating with existing credentials...
Login Succeeded
```

```
day2_task1: digest: sha256:842036eaca6e22c790610fa84ac1b6a45e183f47843d8bda8ac11dd945c4086a size: 2421
The push refers to repository [docker.io/duongtn1512/devops_fpt_learn]
52fc3d18c46d: Layer already exists
3eba84947def: Layer already exists
efe0d4175e55: Layer already exists
c387ce904bb7: Layer already exists
cf3ce544b9f4: Layer already exists
c5f1d4dd95f0: Layer already exists
6a25221bdf24: Layer already exists
b578f477cd5d: Layer already exists
b298f9991a11: Layer already exists
c94dc8fa3d89: Pushing [=================>                                  ]  40.73MB/116.5MB
```

Success login docker hub, build many tag docker images, push it to repo and run a image



Log from container

```
Executing task: docker logs --tail 1000 -f 7b8331d0c99ecd4f8aec64
6aa031483d5341fb2ab7989ffa06e64cf09c0e5c4b

Hello world this is a Duong making task 1 and 2
In docker container it cant read input from terminal
The number a is 1500
Slowly to the To the moon
```

Output of python code I make it into a docker image then run it in a container day2_task1

```
PS D:\Devops_FPT_Foudations\docker\session-02\Task1> docker run --name a001  duongtn1512/devops_fpt_learn:day2_task1
Hello world this is a Duong making task 1 and 2
In docker container it cant read input from terminal
The number a is 1500
Slowly to the To the moon
PS D:\Devops_FPT_Foudations\docker\session-02\Task1>
```

Task 3: Create a multi-stage Dockerfile for a Node.js app. Build the app and copy only the necessary files to the final image.

```
task3
> dist
> pic
JS app.js                U
Dockerfile               U
nginx.conf               U
{} package.json          U
runtask3.ps1             U
```

Dockerfile U | app.js U | nginx.conf U | {} p

docker > Day_02 > Exerise_Day2 > task3 > Dockerfile > FROM

```
FROM node:16-alpine3.17 AS build
WORKDIR /usr/app
COPY package.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:1.22 AS deploy
WORKDIR /app
COPY --from=build /usr/app/dist /app/
COPY nginx.conf /etc/nginx/
CMD [ "nginx", "-g", "daemon off;" ]
```

Dockerfile U | app.js U | nginx.conf U | {} package.json U

docker > Day_02 > Exerise_Day2 > task3 > app.js > ...

```
const http = require('http');

const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello, this is a simple Node.js app!\n');
});

const PORT = process.env.PORT || 3000;
server.listen(PORT, () => {
    console.log(`Server is running on port ${PORT}`);
});
```

Dockerfile U | app.js U | nginx.conf U | {} packag

docker > Day_02 > Exerise_Day2 > task3 > nginx.conf

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;
    sendfile on;

    server {
        listen 80;
        server_name localhost;

        location / {
            root /app;
            index index.html;
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
            root /usr/share/nginx/html;
        }
    }
}
```

package.json U | Dockerfile U | app.js U | nginx.conf U

docker > Day_02 > Exerise_Day2 > task3 > package.json > scripts > build

```json
{
    "name": "simple-node-app",
    "version": "1.0.0",
    "description": "A simple Node.js app",
    "main": "app.js",
    "scripts": {
        "start": "node app.js",
        "build": "mkdir -p dist"
    },
    "dependencies": {}
}
```

runtask3.ps1 U | package.json U | Dockerfile U | app.js U | nginx.conf U | docker

docker > Day_02 > Exerise_Day2 > task3 > runtask3.ps1

```bash
# Build Docker image with specified tag
docker build -t duongtn1512/devops_fpt_learn:day2_task3 .


# Push the image to Docker Hub with custom tag
docker push duongtn1512/devops_fpt_learn:day2_task3

# Remove any existing container with the same name
docker rm -f b1

# Run the Docker image in detached mode, mapping port 3000 and setting the container name
docker run -d -p --name b1 duongtn1512/devops_fpt_learn:day2_task3

```

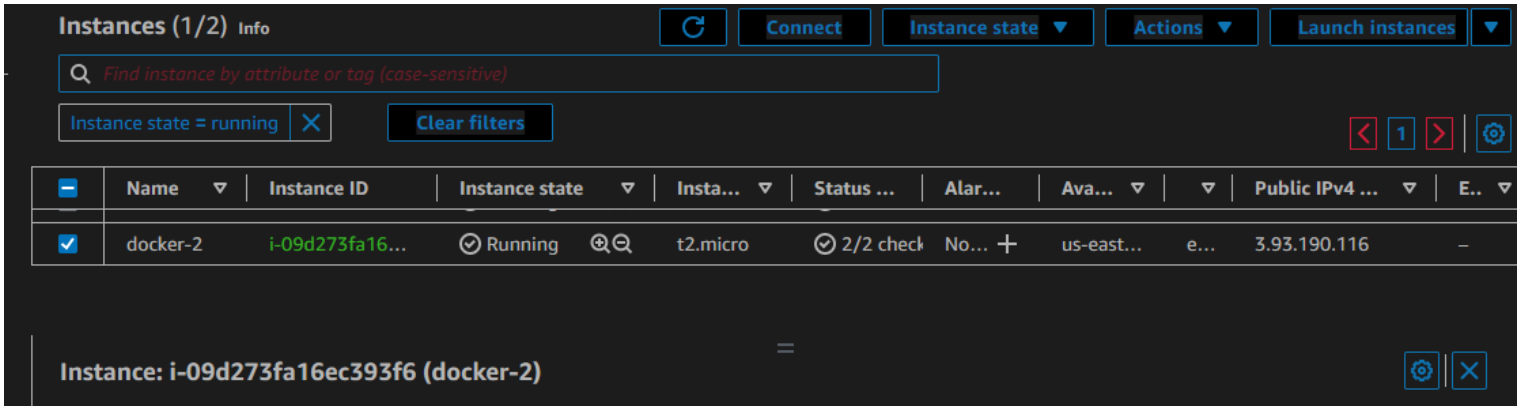
```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task3> ./runtask3.ps1
[+] Building 9.0s (17/17) FINISHED
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 308B
 => [internal] load metadata for docker.io/library/nginx:1.22
 => [internal] load metadata for docker.io/library/node:16-alpine3.17
 => [auth] library/nginx:pull token for registry-1.docker.io
 => [build 1/6] FROM docker.io/library/node:16-alpine3.17
 => [deploy 1/4] FROM docker.io/library/nginx:1.22@sha256:fc5f5fb7574755c306aaf88456ebfbe0b006420a184d52b923d2f0197108f6b7
 => [internal] load build context
 => => transferring context: 119.03kB
 => CACHED [build 2/6] WORKDIR /usr/app
 => CACHED [build 3/6] COPY package.json ./
 => CACHED [build 4/6] RUN npm install
 => [build 5/6] COPY . .
 => [build 6/6] RUN npm run build
 => CACHED [deploy 2/4] WORKDIR /app
 => [deploy 3/4] COPY --from=build /usr/app/dist /app/
 => [deploy 4/4] COPY nginx.conf /etc/nginx/
 => exporting to image
 => => exporting layers
 => => writing image sha256:59ab0b95b4947f563be9ad6ba683590402fe64a84205ca70b00575dca2645737
 => => naming to docker.io/duongtn1512/devops_fpt_learn:day2_task3
```

End result we enter the container, list the files in app and curl localhost:80

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task3> docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED        STATUS         PORTS     NAMES
d02752d3bfa8   duongtn1512/devops_fpt_learn:day2_task3   "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   80/tcp    b1
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task3> docker exec -it b1 bash
root@d02752d3bfa8:/app# ls
LICENSE  README.md  assets  bucket_url.txt  gameController.js  index.html  style.css
root@d02752d3bfa8:/app# curl localhost:80
<!DOCTYPE html>
<html lang="en">
<head>
```

Task 4: Push an image you've created to your Docker Hub repository. Then, pull it to a different machine and run a container.

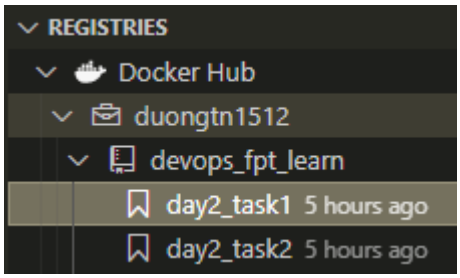
Create an ec2 aws instance



Connect to ec2 via SSH from vs code

```
Connection to 35.175.64.169 closed by remote host.
Connection to 35.175.64.169 closed.
PS D:\Devops_FPT_Foudations\AWS_key> ssh -i  D:\Devops_FPT_Foudations\AWS_key\Key_AWS_1.pem ec2-user@3.93.190.116
The authenticity of host '3.93.190.116 (3.93.190.116)' can't be established.
ECDSA key fingerprint is SHA256:glml0kxAV8guLSmpzH+0BOMj427xvH+uC19sG1Cra1E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.93.190.116' (ECDSA) to the list of known hosts.

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
```

Check images on docker hub repo



Install docker on cenos and check for status

```
sudo yum update -y
sudo amazon-linux-extras install docker
sudo service docker start
sudo systemctl enable docker
sudo usermod -a -G docker ec2-user
docker info
```

```
[ec2-user@ip-172-31-89-120 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2023-08-08 19:38:32 UTC; 28min ago
     Docs: https://docs.docker.com
 Main PID: 9507 (dockerd)
    Tasks: 10
   Memory: 478.8M
   CGroup: /system.slice/docker.service
           └─9507 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536
```

Docker login to docker hub repo

```
[ec2-user@ip-172-31-89-120 ~]$ docker login
Login with your Docker ID to push and pull i
reate one.
Username: duongtn1512
Password:
WARNING! Your password will be stored unencr
Configure a credential helper to remove this
https://docs.docker.com/engine/reference/com

Login Succeeded
```

Pull a image and run (Docker Image is Python base image with code wirted on Task 1)

```
[ec2-user@ip-172-31-89-120 ~]$ docker run --name a1 duongtn1512/devops_fpt_learn:day2_task1
Unable to find image 'duongtn1512/devops_fpt_learn:day2_task1' locally
day2_task1: Pulling from duongtn1512/devops_fpt_learn
785ef8b9b236: Pull complete
5a6dad8f55ae: Pull complete
bd36c7bfe5f4: Pull complete
4d207285f6d2: Pull complete
9402da1694b8: Pull complete
6fa59a7ce94b: Pull complete
cc429e3ed9d5: Pull complete
eb752ab36a04: Pull complete
6e29556ddf64: Pull complete
8eeaea3eba21: Pull complete
Digest: sha256:842036eaca6e22c790610fa84ac1b6a45e183f47843d8bda8ac11dd945c4086a
Status: Downloaded newer image for duongtn1512/devops_fpt_learn:day2_task1
Hello world this is a Duong making task 1 and 2
In docker container it cant read input from terminal
The number a is 1500
Slowly to the To the moon
```

Check docker contaner on this EC2 machine

```
[ec2-user@ip-172-31-89-120 ~]$ docker ps -a
CONTAINER ID   IMAGE                                  COMMAND            CREATED          STATUS                      PORTS     NAMES
01f38889c002   duongtn1512/devops_fpt_learn:day2_task1   "python task1.py"   25 minutes ago   Exited (0) 25 minutes ago           a1
[ec2-user@ip-172-31-89-120 ~]$
```

Task 5: From an official Nginx image, build a custom Nginx image to output a customized homepage.

Task 6: Run an Nginx container in detached mode, map a local port to the container's port, and access the Nginx welcome page. (2 Task 1 run)

Fist we make custom html file display what we want

index.html U × | Dockerfile U × | runtask5.ps1 U | powershell | exercises.md

docker > Day_02 > Exerise_Day2 > task5 > index.html > html > body > div.container > a.cta-button

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome to My Awesome Website</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f8f8f8;
            margin: 0;
            padding: 0;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }

        .container {
            text-align: center;
        }

        .heading {
            font-size: 2.5rem;
            color: #333;
            margin-bottom: 1rem;
        }
```

Dockerfile will put index.html file to the folder that will deploy our frontend web we custom build

index.html U | Dockerfile U × | runtask5.ps1 U | pov

docker > Day_02 > Exerise_Day2 > task5 > Dockerfile > ...

```dockerfile
FROM nginx

COPY index.html /usr/share/nginx/html/index.html

CMD ["nginx", "-g", "daemon off;"]
```

Auto run script that will run a container in detached mode map port 1512 to the container's port 80

index.html U | Dockerfile U | runtask5.ps1 U × | powershell | exercises.md

docker > Day_02 > Exerise_Day2 > task5 > runtask5.ps1

```powershell
docker build -t nginx_task5 .
docker run -d --name a555 -p 1512:80 nginx_task5
```

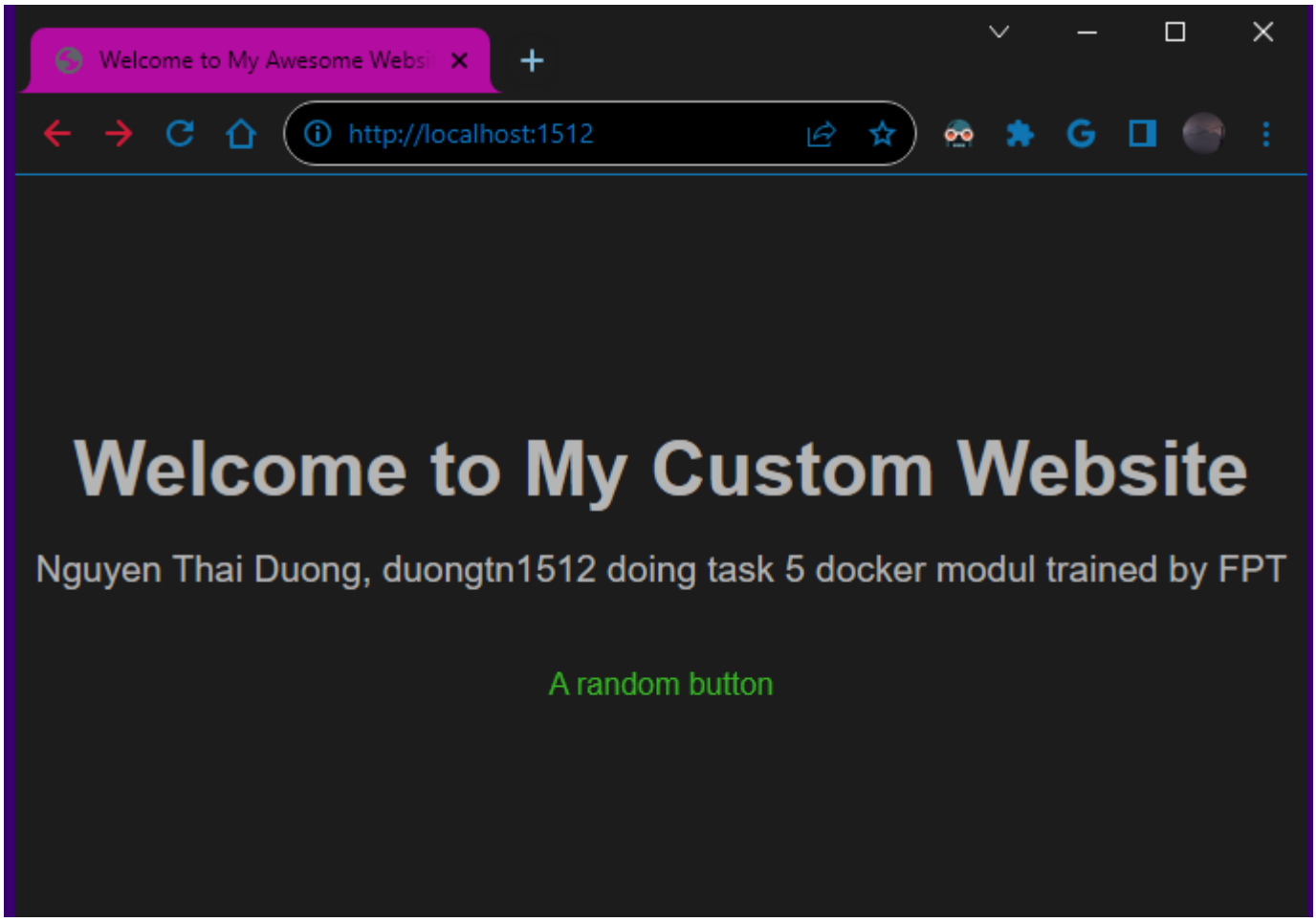
We start the script

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task5> ls


    Directory: D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task5


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         8/14/2023   7:51 AM            100 Dockerfile
-a----         8/14/2023   7:49 AM           1548 index.html
-a----         8/14/2023   7:54 AM             84 runtask5.ps1


PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task5> ./runtask5
[+] Building 0.6s (7/7) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.1s
 => => transferring dockerfile: 137B                                                                                          0.0s
 => [internal] load .dockerignore                                                                                             0.1s
 => => transferring context: 2B                                                                                               0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                                               0.0s
 => [internal] load build context                                                                                             0.1s
 => => transferring context: 1.59kB                                                                                           0.0s
 => CACHED [1/2] FROM docker.io/library/nginx                                                                                 0.0s
 => [2/2] COPY index.html /usr/share/nginx/html/index.html                                                                    0.1s
 => exporting to image                                                                                                        0.1s
 => => exporting layers                                                                                                       0.0s
 => => writing image sha256:65ed9e262b17fb125ec803e43e9982adc0370cc62bf108e02c34a7a6dac3a32f                                  0.0s
 => => naming to docker.io/library/nginx_task5                                                                                0.0s
```

Check for the result

	Name	Image	Status	Port(s)	Last started	C Actions
	a555 98d97e37d147	nginx_task5	Running	1512:80	23 minutes ago	

The homepage we just deploy



Task 7: Run an Ubuntu container interactively, access its shell, and execute basic commands.

runtask6.ps1 U × | exercises.md × | powershell

docker > Day_02 > Exerise_Day2 > task6 > runtask6.ps1

```
docker pull ubuntu
docker run -it --name ubt001 ubuntu
```

Run the script and do basic command

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task6> ./runtask6
Using default tag: latest
latest: Pulling from library/ubuntu
3153aa388d02: Pull complete
Digest: sha256:0bced47fffa3361afa981854fcabcd4577cd43cebbb808cea2b1f33a3dd7f508
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview ubuntu
root@797820cf2614:/# pwd
/
root@797820cf2614:/# ls -a
.  ..  .dockerenv  bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@797820cf2614:/# unname -a
bash: unname: command not found
root@797820cf2614:/# uname -a
Linux 797820cf2614 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
root@797820cf2614:/# exit
exit
```

Task 8: Run a container with environment variables set using the -e flag and `ENV` instruction. Access these variables from within the container.

We fist create an env file and assign the variables same with the path we make docker command

task8.env U ● | docker | exercises.md

docker > Day_02 > Exerise_Day2 > task8 > task8.env

```
update=apt-get update
upgrade=apt-get upgrade
```

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task8> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task8> ls


    Directory: D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task8


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         8/14/2023  12:34 PM             46 task8.env


PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\task8> docker run -it --env-file task8.env ubuntu
root@0762cfb14446:/# echo $update
apt-get update
root@0762cfb14446:/# echo $upgrade
apt-get upgrade
root@0762cfb14446:/# $update
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [832 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
```

Folow the result we run a ubuntu container with flag -env to task8.env file where we make our varible we then echo our varible and try one to see if they work

Task 9: Build a Docker image using the Dockerfile in the `issue` directory and run a container using that image. What happened with the container? Troubleshoot so the container is able to run.

Fist we make script stage 1 to build and run stage 2 to use different command to inspect container

runtask9.ps1 U | Dockerfile U | index.html U | powershell | exercises.md M

docker > Day_02 > Exerise_Day2 > issue > runtask9.ps1

```
# Build and run nginx container
docker build -t nginx_task9 .
docker run -d --name nginx09 -p 1999:80 nginx_task9
# Troubleshoot command line
docker logs nginx09
docker inspect nginx09
docker ps -a
docker history nginx_task9 .
```

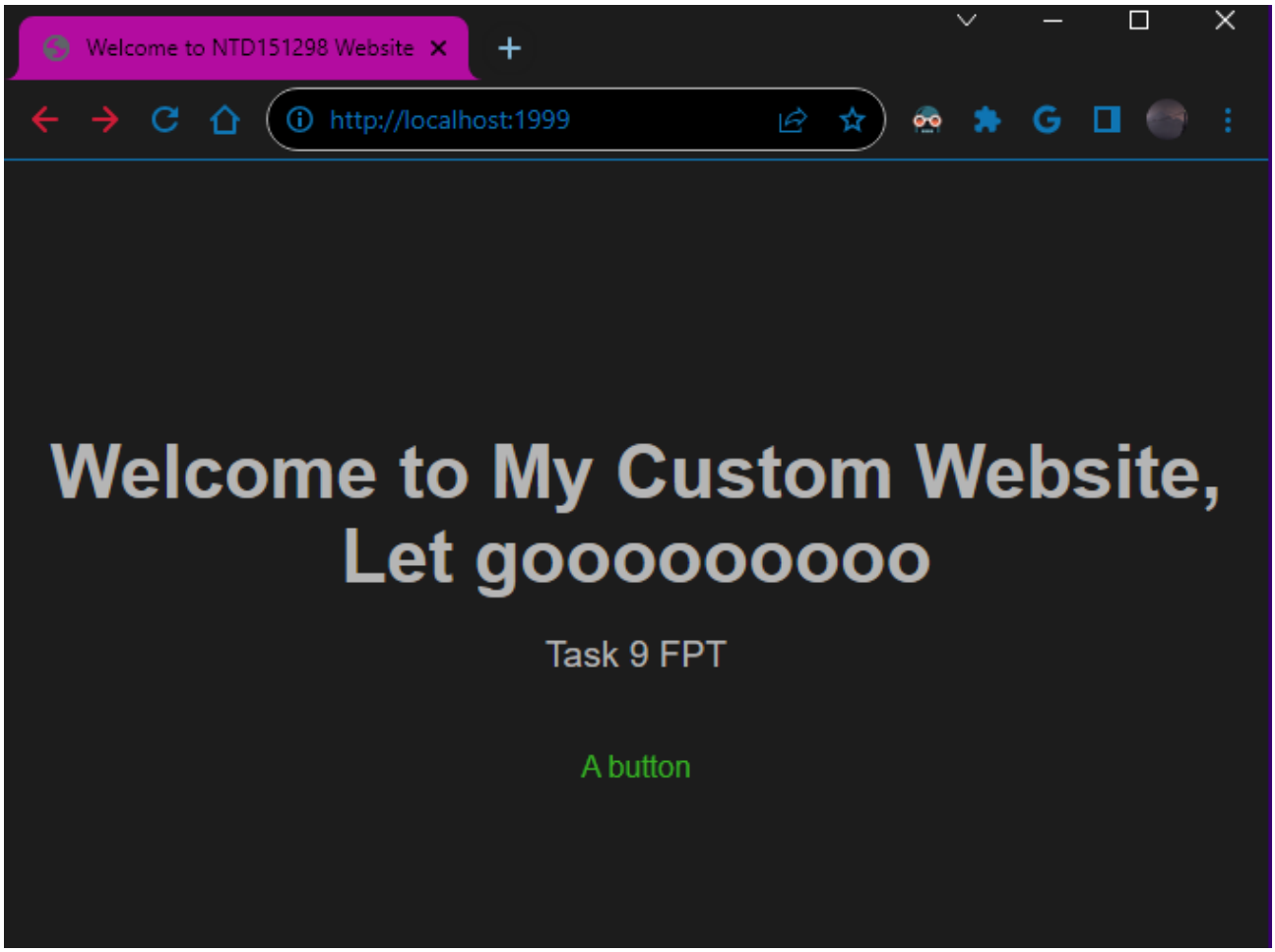
Result of build stage

runtask9.ps1 U | Dockerfile U | index.html U | powershell | exercises.md M

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> ./runtask9
[+] Building 0.6s (7/7) FINISHED                                                    docker:default
 => [internal] load build definition from Dockerfile                                          0.1s
 => => transferring dockerfile: 137B                                                          0.0s
 => [internal] load .dockerignore                                                             0.1s
 => => transferring context: 2B                                                               0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                               0.0s
 => [internal] load build context                                                             0.1s
 => => transferring context: 1.54kB                                                           0.0s
 => CACHED [1/2] FROM docker.io/library/nginx                                                 0.0s
 => [2/2] COPY index.html /usr/share/nginx/html/index.html                                    0.0s
 => exporting to image                                                                        0.1s
 => => exporting layers                                                                       0.1s
 => => writing image sha256:55d1b8c9b7e63cbcc89ae65c2ffa9ec0f8ea672f01f3e1c9632bf1849631f8cb  0.0s
 => => naming to docker.io/library/nginx_task9                                                0.0s
```

Log and deep inspect

```
dbd0f536f766138fb66e6544e16482c629419ef725a256aaef2ea02cb5ee771a
/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
[
    {
        "Id": "dbd0f536f766138fb66e6544e16482c629419ef725a256aaef2ea02cb5ee771a",
        "Created": "2023-08-14T06:08:31.5026841Z",
        "Path": "/docker-entrypoint.sh",
        "Args": [
            "nginx",
            "-g",
            "daemon off;"
        ],
        "State": {
            "Status": "running",
```



It appear that container run smotly and we can curl to it port public at localhost:1999

Task 10: Run a container, stop it, and then remove it.

```
runtask9.ps1 U   Dockerfile U   index.html U   powershell ×   exercises.md M

PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
nginx        latest    89da1fb6dcb9   2 weeks ago   187MB
ubuntu       latest    5a81c4b8502e   6 weeks ago   77.8MB
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker run -d --name a1 nginx
3352c8820d0eccb2756fcd54e967551519a00a30204e89d86709fb81a648ce43
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
3352c8820d0e   nginx     "/docker-entrypoint.…"   5 seconds ago   Up 4 seconds   80/tcp    a1
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker stop a1
a1
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                     PORTS     NAMES
3352c8820d0e   nginx     "/docker-entrypoint.…"   13 seconds ago   Exited (0) 4 seconds ago             a1
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker rm a1
a1
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\issue>
```

Task 11: Build an image with a custom HTML file. Run a container from the image and copy a file from the host to the container.

The custom HTML file:

index.html U × | powershell | runtask11.ps1 U | Dockerfile U | text.txt U

docker > Day_02 > Exerise_Day2 > final_task > index.html > html

```html
</head>

<body>
    <div class="container">
        <h1 class="heading">Welcome to My Custom Website</h1>
        <p class="description">Task 11 final task bro it not hard. but take a lot of time</p>
        <button class="cta-button">Display Text</button>
        <div id="displayText"></div>
    </div>

    <script>
        const ctaButton = document.querySelector('.cta-button');
        const displayTextDiv = document.querySelector('#displayText');

        ctaButton.addEventListener('click', async () => {
            try {
                const response = await fetch('text.txt');
                if (response.ok) {
                    const text = await response.text();
                    displayTextDiv.textContent = text;
                } else {
                    displayTextDiv.textContent = 'No text available.';
                }
            } catch (error) {
                displayTextDiv.textContent = 'Error loading text.';
            }
        });
    </script>
</body>

</html>
```

Dockerfile

index.html U × | powershell | runtask11.ps1 U | Dockerfile U

docker > Day_02 > Exerise_Day2 > final_task > Dockerfile > ...

```dockerfile
FROM nginx

COPY index.html /usr/share/nginx/html/index.html

CMD ["nginx", "-g", "daemon off;"]
```

Auto build and run script

index.html U × | powershell | runtask11.ps1 U | Dockerfile U

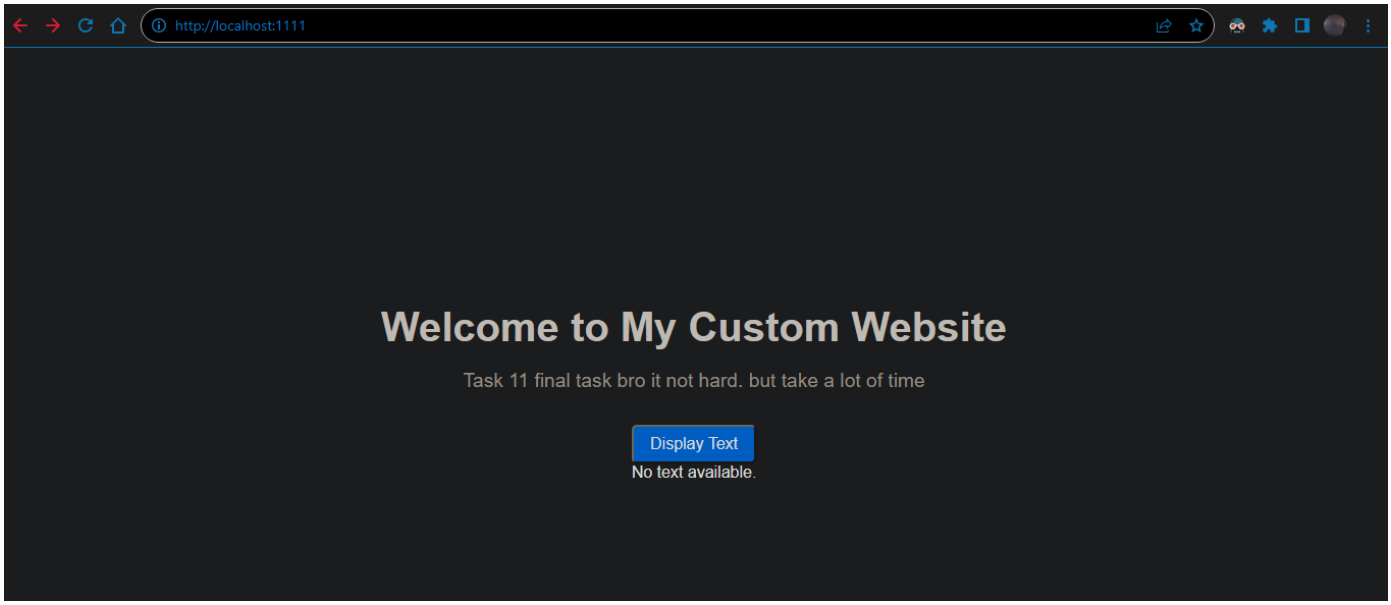
docker > Day_02 > Exerise_Day2 > final_task > runtask11.ps1

```powershell
docker build -t nginx_task11 .
docker run -d --name nginx11 -p 1111:80 nginx_task11
```

Run script

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\final_task> ./runtask11
[+] Building 27.9s (8/8) FINISHED                                                                                docker:default
 => [internal] load build definition from Dockerfile                                                                      0.1s
 => => transferring dockerfile: 139B                                                                                      0.0s
 => [internal] load .dockerignore                                                                                         0.1s
 => => transferring context: 2B                                                                                           0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                                           3.7s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                              0.0s
 => [internal] load build context                                                                                         0.1s
 => => transferring context: 2.31kB                                                                                       0.0s
 => [1/2] FROM docker.io/library/nginx@sha256:104c7c5c54f2685f0f46f3be607ce60da7085da3eaa5ad22d3d9f01594295e9c           22.7s
 => => resolve docker.io/library/nginx@sha256:104c7c5c54f2685f0f46f3be607ce60da7085da3eaa5ad22d3d9f01594295e9c            0.1s
 => => sha256:104c7c5c54f2685f0f46f3be607ce60da7085da3eaa5ad22d3d9f01594295e9c 1.86kB / 1.86kB                            0.0s
 => => sha256:48a84a0728cab8ac558f48796f901f6d31d287101bc8b317683678125e0d2d35 1.78kB / 1.78kB                            0.0s
 => => sha256:eea7b3dcba7ee47c0d16a60cc85d2b977d166be3960541991f3e6294d795ed24 8.15kB / 8.15kB                            0.0s
 => => sha256:fd9f026c631046113bd492f69761c3ba6042c791c35a60e7c7f3b8f254592daa 41.34MB / 41.34MB                         11.0s
 => => sha256:055fa98b43638b67d10c58d41094d99c8696cc34b7a960c7a0cc5d9d152d12b3 628B / 628B                                0.7s
 => => sha256:52d2b7f179e32b4cbd579ee3c4958027988f9a8274850ab0c7c24661e3adaac5 29.12MB / 29.12MB                          13.6s
```

Check the localhost:1111



Check container file

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\final_task> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
c8fcb0469aee   nginx_task11   "/docker-entrypoint.…"   59 seconds ago   Up 57 seconds   0.0.0.0:1111->80/tcp     nginx11
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\final_task> docker exec -it nginx11 bash
root@c8fcb0469aee:/# ls
bin   dev                  docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                   lib   lib64  media   opt  root  sbin  sys  usr
root@c8fcb0469aee:/# cd /usr/share/nginx/html
root@c8fcb0469aee:/usr/share/nginx/html# ls
50x.html  index.html
root@c8fcb0469aee:/usr/share/nginx/html#
```

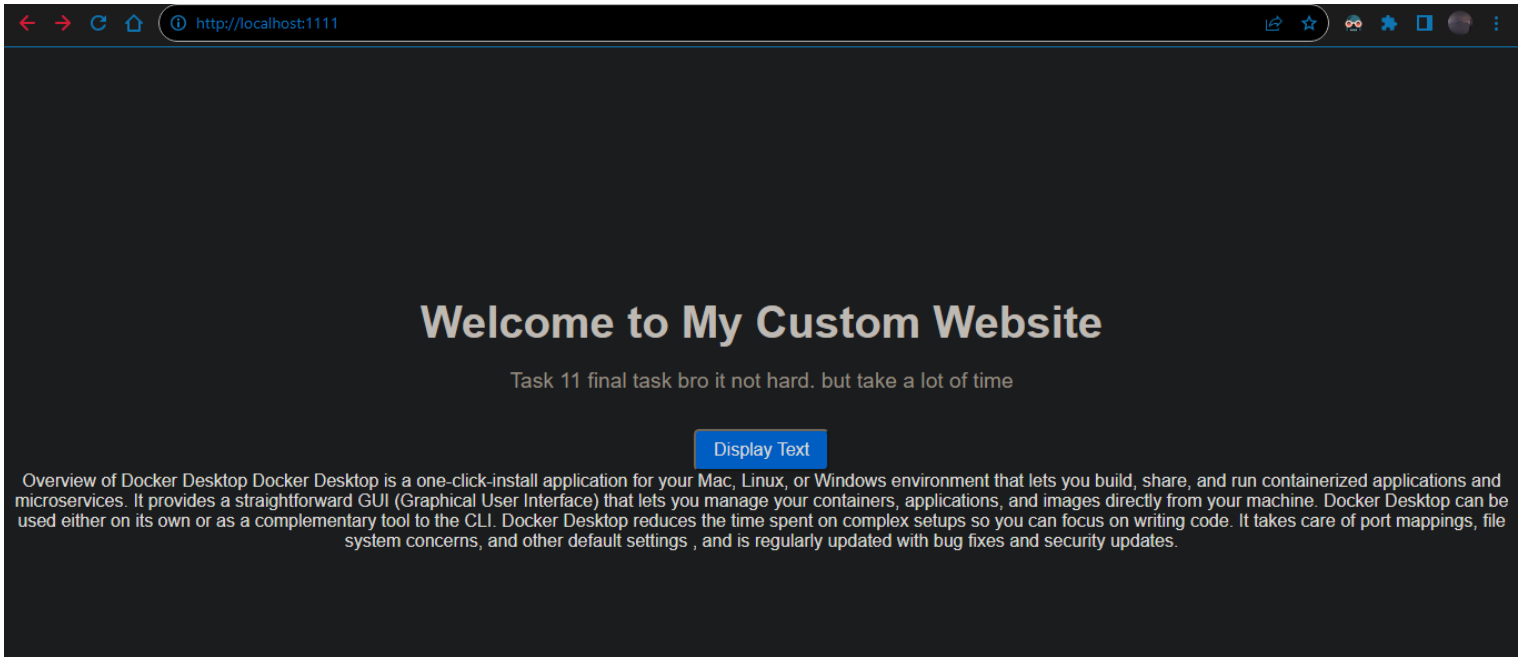
Create file text.txt on the same dir terminal at

```
index.html U   runtask11.ps1 U   Dockerfile U   text.txt U   docker
docker > Day_02 > Exerise_Day2 > final_task > text.txt
1   Overview of Docker Desktop
2   Docker Desktop is a one-click-install application for your Mac, Linux, or Windows
3   environment that lets you build, share, and run containerized applications and microservices.
4
5   It provides a straightforward GUI (Graphical User Interface) that lets you manage your
6   containers, applications, and images directly from your machine. Docker Desktop can be
7   used either on its own or as a complementary tool to the CLI.
8
9   Docker Desktop reduces the time spent on complex setups so you can focus on writing
10  code. It takes care of port mappings, file system concerns, and other default settings
11  , and is regularly updated with bug fixes and security updates.
12
```

Docker copy the text file to docker container at the same dir that container are display web code

```
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\final_task> docker cp text.txt nginx11:/usr/share/nginx/html/
Successfully copied 2.56kB to nginx11:/usr/share/nginx/html/
PS D:\Devops_FPT_Foudations\docker\Day_02\Exerise_Day2\final_task> docker exec -it nginx11 bash
root@c8fcb0469aee:/# ls
bin   dev                  docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                    lib   lib64  media   opt  root  sbin  sys  usr
root@c8fcb0469aee:/# cd /usr/share/nginx/html
root@c8fcb0469aee:/usr/share/nginx/html# ls
50x.html  index.html  text.txt
root@c8fcb0469aee:/usr/share/nginx/html# cat text.txt
Overview of Docker Desktop
Docker Desktop is a one-click-install application for your Mac, Linux, or Windows
environment that lets you build, share, and run containerized applications and microservices.

It provides a straightforward GUI (Graphical User Interface) that lets you manage your
containers, applications, and images directly from your machine. Docker Desktop can be
used either on its own or as a complementary tool to the CLI.

Docker Desktop reduces the time spent on complex setups so you can focus on writing
code. It takes care of port mappings, file system concerns, and other default settings
, and is regularly updated with bug fixes and security updates.root@c8fcb0469aee:/usr/share/nginx/html#
```

We also check the file data after copy finish and check web brower that code are deploy at



Folow the logic in html file code the button Display Text will display data from file text.txt

```html
<script>
    const ctaButton = document.querySelector('.cta-button');
    const displayTextDiv = document.querySelector('#displayText');

    ctaButton.addEventListener('click', async () => {
        try {
            const response = await fetch('text.txt');
            if (response.ok) {
                const text = await response.text();
                displayTextDiv.textContent = text;
            } else {
                displayTextDiv.textContent = 'No text available.';
            }
        } catch (error) {
            displayTextDiv.textContent = 'Error loading text.';
        }
    });
</script>
</body>
```

Nguyễn Thái Dương Day 3 docker network exerise:

Task: create a docker compose file that run 2 container in a same network then try to check connection within that network using container dns

Here is that file docker compose and file env that docker compose will use to run

docker > docker-compose.yaml U

docker > Day_03 > docker-compose.yaml

```yaml
version: '3.4'

networks:
  test_day3:
    driver: bridge

services:
  nginx1:
    container_name: ${con1}
    image: nginx
    networks:
      - ${net}
    ports:
      - "${port1}"

  nginx2:
    container_name: ${con2}
    image: nginx
    networks:
      - ${net}
    ports:
      - "${port2}"
```

.env U

docker > Day_03 > .env

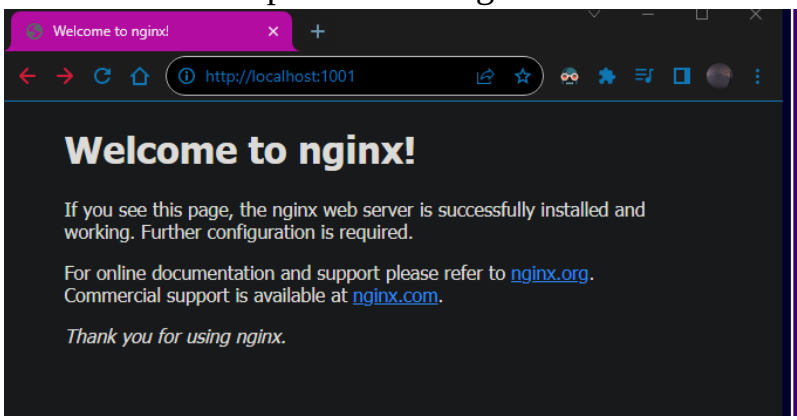
```env
con1=web1
con2=web2
net=test_day3
port1=127.0.0.1:1001:80
port2=127.0.0.1:1002:80
```

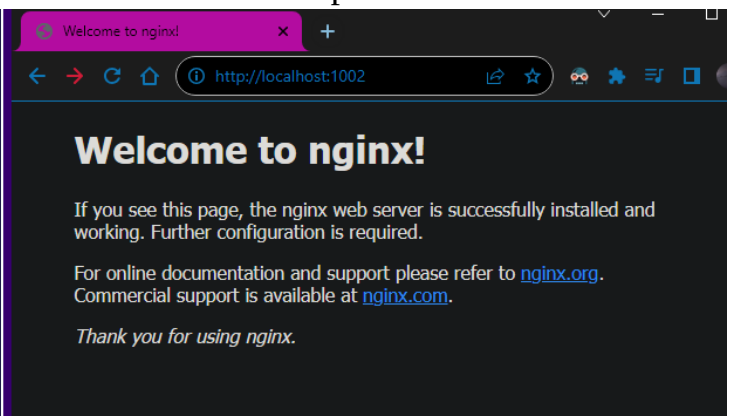
This is terminal result after we enter command “docker compose up -d” at wor dir

```
 ✓ 52d2b7f179e3 Pull complete                                                    33.7s
 ✓ fd9f026c6310 Pull complete                                                    37.1s
 ✓ 055fa98b4363 Pull complete                                                    37.1s
 ✓ 96576293dd29 Pull complete                                                    37.1s
 ✓ a7c4092be904 Pull complete                                                    37.2s
 ✓ e3b6889c8954 Pull complete                                                    37.2s
 ✓ da761d9a302b Pull complete                                                    37.2s
✓ nginx2 Pulled                                                                  43.6s
[+] Running 1/3
 ✓ Network day_03_test_day3  Created                                              0.7s
 - Container web1            Starting                                             [+] R
[+] Running 3/3
 ✓ Network day_03_test_day3  Created                                              0.7s
 ✓ Container web1            Started                                             24.3s
 ✓ Container web2            Started                                             24.3s
PS D:\Devops_FPT_Foudations\docker\Day_03> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                    NAMES
4c8a71a4802e   nginx     "/docker-entrypoint.…"   51 seconds ago   Up 27 seconds   127.0.0.1:1001->80/tcp   web1
d331a6f3b229   nginx     "/docker-entrypoint.…"   51 seconds ago   Up 27 seconds   127.0.0.1:1002->80/tcp   web2
PS D:\Devops_FPT_Foudations\docker\Day_03>
```

Folow the docker ps we run nginx with name web1 and web2 on localhost:1001 and localhost:1002

We have success pull and run nginx as container name web1 and web2 on port 1001 and 1002





Now we check for web1 and web2 network

```
PS D:\Devops_FPT_Foudations\docker\Day_03> docker network ls
NETWORK ID     NAME               DRIVER    SCOPE
4c2270f871c3   bridge             bridge    local
12ccc6bb4f4e   day_03_test_day3   bridge    local
a2cec830f010   docker_gwbridge    bridge    local
eeb939a64378   host               host      local
24ml2wzapgd7   ingress            overlay   swarm
fb9c40f5fca2   none               null      local
82eb06763a3e   test_demo_net_1    bridge    local
PS D:\Devops_FPT_Foudations\docker\Day_03> docker network inspect day_03_test_day3
[
    {
        "Name": "day_03_test_day3",
        "Id": "12ccc6bb4f4e667f658d8efa6b157cda1f4e5d2f1aaae290693f21f23f9e8e15",
        "Created": "2023-08-17T19:43:59.8234576Z",
        "Scope": "local",
        "Driver": "bridge",
```

Our network named day_03_test_day3

```
"Containers": {
    "4c8a71a4802ea5eadf931f23fe5ecc9b5497daf23a9eec22704ce11f2ff357aa": {
        "Name": "web1",
        "EndpointID": "2e29da2a9a9890043a5eab21aa529d71cea7dc64c96db54e0496c9ba4671776c",
        "MacAddress": "02:42:ac:14:00:02",
        "IPv4Address": "172.20.0.2/16",
        "IPv6Address": ""
    },
    "d331a6f3b2291811a63ca4b53522fd05c7b9bff589f5dc089ec3c97036075204": {
        "Name": "web2",
        "EndpointID": "9dae88d569fe2fa23b6821327a90cf1177bc2e5d16853ca3f188325530d371c3",
        "MacAddress": "02:42:ac:14:00:03",
        "IPv4Address": "172.20.0.3/16",
        "IPv6Address": ""
```

With docker network inspect we find out ip of web1 within docker network is 172.20.0.2 and web2 ip is 172.20.0.3

We enter container web1 by using “docker exec -it web1 bash”

```
PS D:\Devops_FPT_Foudations\docker\Day_03> docker exec -it web1 bash
root@4c8a71a4802e:/# ls
bin   dev                docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  docker-entrypoint.d  etc                  lib   lib64  media   opt  root  sbin  sys  usr
root@4c8a71a4802e:/# w
```

We then try to make sure connection between web1 and web2 is good by using curl and domaname

```
root@4c8a71a4802e:/# curl web2
<!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>
root@4c8a71a4802e:/#
```

Nguyễn Thái Dương Day 4 docker volume exerise:

1. Create a named volume called "data_volume" and mount it to a container. Verify that data is persisted even after the container is removed.

2. Mount a directory from your host machine to a container using a bind mount. Modify files from both the host and the container to observe changes.

3. Use the docker volume inspect command to view metadata and configuration details of a volume.

4. Identify and remove volumes that are no longer in use to free up storage space.

5. Backup the contents of a volume to your local machine and then restore it to a new volume.

6. Create a container with a tmpfs mount to store temporary data in memory. Observe how the data is lost when the container stops.

7. Write a docker-compose.yml file that defines a service using volumes, then launch multiple containers to share data.

8. Launch a container that uses multiple volumes for different parts of its filesystem.

9. Create two containers, migrate data from one to the other using volumes, and ensure minimal downtime.

10. Launch multiple instances of a container and share data using the same volume between them.

Task 1: Create a named volume called "data_volume" and mount it to a container. Verify that data is persisted even after the container is removed.

Dockerfile U × | powershell | compose.yaml ...\Nginx_t

docker > Day_04 > task1 > Dockerfile > ...

```dockerfile
FROM node:16-alpine3.17 AS build
WORKDIR /usr/app
COPY package.json ./
RUN npm install
COPY . .
RUN npm run build

FROM nginx:1.22 AS deploy
WORKDIR /app
COPY --from=build /usr/app/dist /app/
COPY nginx.conf /etc/nginx/nginx.conf
CMD [ "nginx", "-g", "daemon off;" ]
```

```
PS D:\Devops_FPT_Foudations> docker volume ls
PS D:\Devops_FPT_Foudations> docker volume ls
PS D:\Devops_FPT_Foudations> docker volume ls
DRIVER    VOLUME NAME
local     a001
local     a002
local     a003
PS D:\Devops_FPT_Foudations> docker volume inspect a001
[
    {
        "CreatedAt": "2023-08-11T12:40:28Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/a001/_data",
        "Name": "a001",
        "Options": null,
        "Scope": "local"
    }
]
PS D:\Devops_FPT_Foudations>
```

Using dockerfile we create image duongtn1512/devops_fpt_learn:day2_task3 and use volume a001

From docker compose we mount a001 to new image we created and name container bb1

The file in dist folder will get put into folder app in container bb1 folow docker file in deploy stage

compose.yaml U × | Dockerfile U | powershell | pow

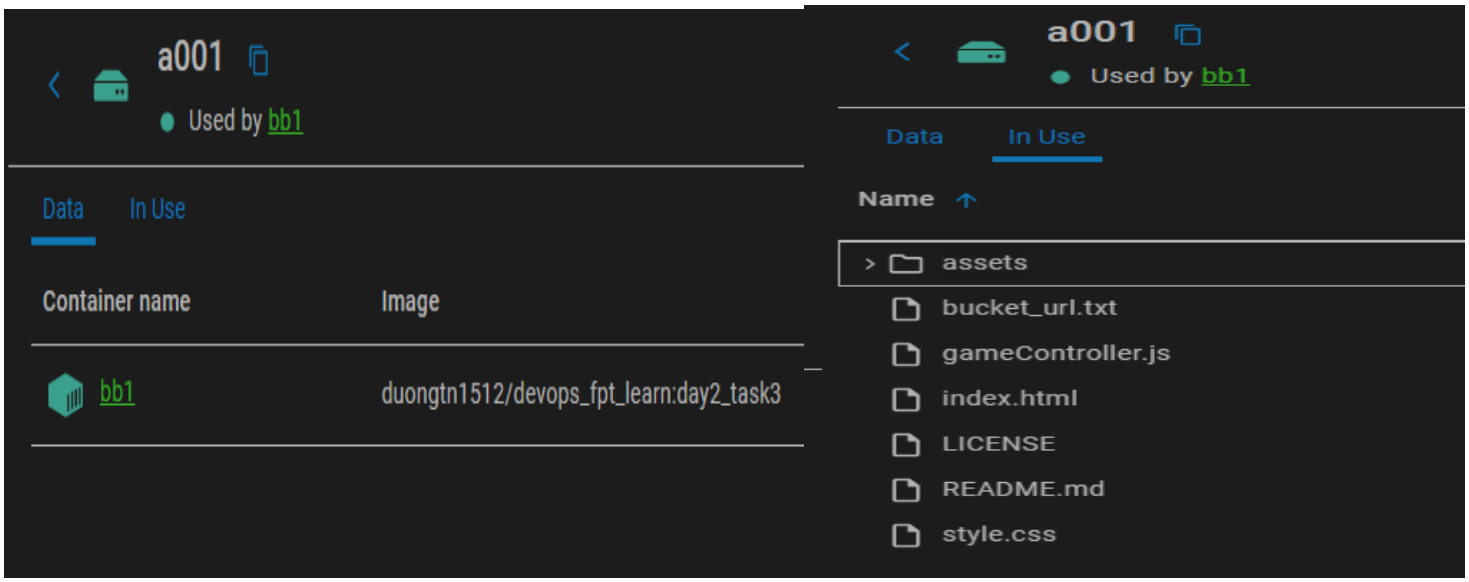
docker > Day_04 > task1 > compose.yaml

```yaml
version: '3'
services:

  chess:
    image: duongtn1512/devops_fpt_learn:day2_task3
    container_name: bb1
    volumes:
      - a001:/app

volumes:
  a001:
    external: true
```

- task1
  - dist
    - assets
    - bucket_url.txt U
    - gameController.js U
    - index.html U
    - LICENSE U
    - README.md U
    - style.css U

```
PS D:\Devops_FPT_Foudations\docker\Day_04\task1> docker compose up -d
[+] Running 2/2
 ✓ Network task1_default  Created
 ✓ Container bb1          Started
PS D:\Devops_FPT_Foudations\docker\Day_04\task1>
```

The file in app folder will get mount to storage a001



We delete container bb1

```
PS D:\Devops_FPT_Foudations\docker\Day_04\task1> docker compose up -d
[+] Running 2/2
 ✓ Network task1_default  Created
 ✓ Container bb1          Started
PS D:\Devops_FPT_Foudations\docker\Day_04\task1> docker compose down
[+] Running 2/2
 ✓ Container bb1          Removed
 ✓ Network task1_default  Removed
PS D:\Devops_FPT_Foudations\docker\Day_04\task1>
```

And then we check data in storage a001 when not in use by any docker containers

Its still here.

Task 2: Mount a directory from your host machine to a container using a bind mount. Modify files from both the host and the container to observe changes.

Prepare Directory and Files:

With host folder is task2_host_floder and host file is host_file.txt

```
Day_04
  > Nginx_test
  > others
  > task1
  > task2_host_floder
    host_file.txt  U
  > teacher_doc
```

```
    Directory: D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----        8/11/2023  10:42 PM             87 host_file.txt
```

Data in file host_file.txt

```
docker    host_file.txt U ×
docker > Day_04 > task2_host_floder > host_file.txt
1   This is a 1512 demo pls folowing the guide to compelish task 2 of docker of some what ?
```

Create an contaner name cc1 with bind mount task2_host_floder to it contaner path /app:

```
docker run -it --name cc1 -v D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder:/app
nginx bash
```

We then enter container and check for the file host_file.txt

```
root@8940c4f8a4f1:/# ls
app  boot  docker-entrypoint.d   etc   lib    lib64   media  opt   root  sbin  sys  usr
bin  dev   docker-entrypoint.sh  home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@8940c4f8a4f1:/# ls app
host_file.txt
root@8940c4f8a4f1:/# cd app
root@8940c4f8a4f1:/app# cat host_file.txt
This is a 1512 demo pls folowing the guide to compelish task 2 of docker of some what ?r
```

We had success bring file from windows host to docker nginx container

Modify from Host

```
docker    host_file.txt U ×
docker > Day_04 > task2_host_floder > host_file.txt
1   Hello Bro
```

And then we enter container to see if it change the txt file

```
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker exec -it cc1 bash
root@8940c4f8a4f1:/# ls
app  boot  docker-entrypoint.d   etc   lib    lib64   media  opt   root  sbin  sys  usr
bin  dev   docker-entrypoint.sh  home  lib32  libx32  mnt    proc  run   srv   tmp  var
root@8940c4f8a4f1:/# cd app
root@8940c4f8a4f1:/app# ls
host_file.txt
root@8940c4f8a4f1:/app# cat host_file.txt
Hello Broroot@8940c4f8a4f1:/app#
```

Yes it dose change the data in host_file.txt

Modify from Container

```
root@8940c4f8a4f1:/app# echo "DUONG THIS IS SPARTA !" > host_file.txt
root@8940c4f8a4f1:/app# cat host_file.txt
DUONG THIS IS SPARTA !
root@8940c4f8a4f1:/app#
```

And in the host is changed

```
docker        host_file.txt U ×
docker > Day_04 > task2_host_floder > host_file.txt
1   DUONG THIS IS SPARTA !
2
```

So both ways are ok and can make change to the file its bind mouth

Task 3: Use the docker volume inspect command to view metadata and configuration details of a volume.

```
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker volume ls
DRIVER    VOLUME NAME
local     a001
local     a002
local     a003
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker volume inspect a001
[
    {
        "CreatedAt": "2023-08-11T12:40:28Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/a001/_data",
        "Name": "a001",
        "Options": null,
        "Scope": "local"
    }
]
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder>
```

Task 4: Identify and remove volumes that are no longer in use to free up storage space.

```
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker ps -a
CONTAINER ID   IMAGE                                   COMMAND                  CREATED              STATUS              PORTS     NAMES
a555a88d1b26   duongtn1512/devops_fpt_learn:day2_task3   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    bb1
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker volume ls
DRIVER    VOLUME NAME
local     a001
local     a002
local     a003
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker inspect bb1 | Select-String "Mounts" -Context 3

            "CpuPercent": 0,
            "IOMaximumIOps": 0,
            "IOMaximumBandwidth": 0,
>           "Mounts": [
                {
                    "Type": "volume",
                    "Source": "a001",
            },
            "Name": "overlay2"
        },
>       "Mounts": [
            {
                "Type": "volume",
                "Name": "a001",


PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder>
```

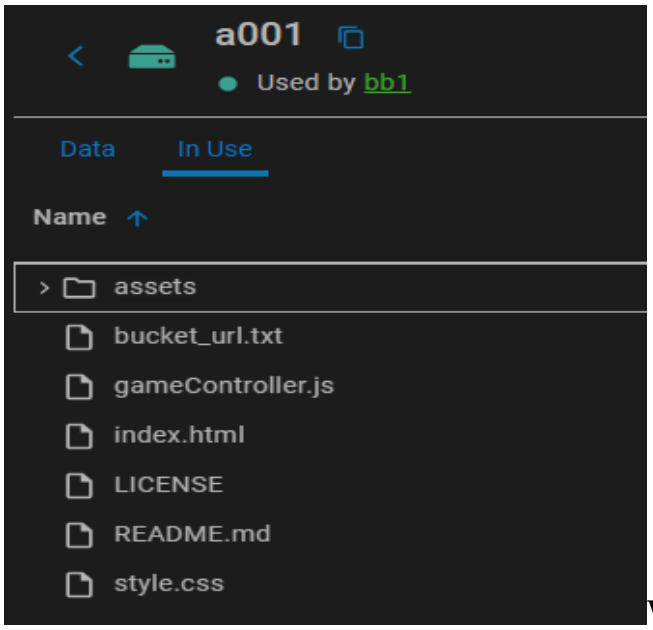
So from the image I get, I belive that only volume a001 in used with container bb1

We rm volume a002 and a003 because it not in use

```
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker volume rm a002 a003
a002
a003
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder> docker volume ls
DRIVER    VOLUME NAME
local     a001
PS D:\Devops_FPT_Foudations\docker\Day_04\task2_host_floder>
```

Task 5: Backup the contents of a volume to your local machine and then restore it to a new volume.

Fist we use volume a001 created before in task 1



We use this comand too backup at host windows

```
docker run --rm -v a001:/data -v C:\local\backup\path:/backup nginx tar cvf /backup/backup.tar /data
```

Check path we backup data C:\local\backup\path using ls command

```
PS C:\local\backup\path> docker run --rm -v a001:/data -v C:\local\backup\path:/backup nginx tar cvf /backup/backup.tar /data
tar: Removing leading `/' from member names
/data/
/data/LICENSE
/data/README.md
/data/assets/
/data/assets/white_king.png
/data/assets/black_rook.png
/data/assets/white_pawn.png
/data/assets/black_bishop.png
/data/assets/black_knight.png
/data/assets/black_king.png
/data/assets/white_bishop.png
/data/assets/white_rook.png
/data/assets/black_pawn.png
/data/assets/black_queen.png
/data/assets/white_knight.png
/data/assets/white_queen.png
/data/style.css
/data/gameController.js
/data/bucket_url.txt
/data/index.html
PS C:\local\backup\path> ls


    Directory: C:\local\backup\path


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         8/12/2023   1:29 AM          61440 backup.tar
```

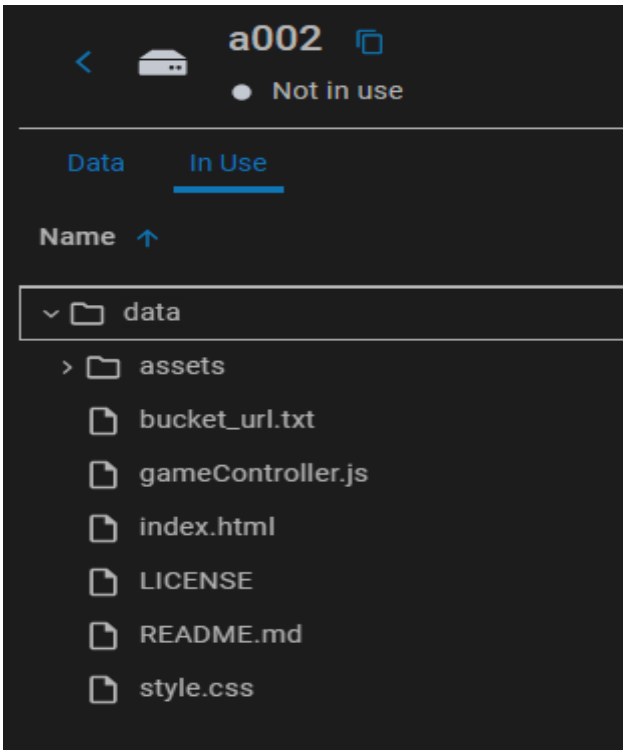
As we can see we have store volume a001 data to file backup.tar at C:\local\backup\path

For the restore the data to another volume, we create a new volume and bring data to that volume

```
docker volume create a002

docker run --rm -v a002:/data -v C:\local\backup\path:/backup nginx tar xvf /backup/backup.tar -C /data
```

We inspect the data from docker desktop

```
PS C:\local\backup\path> docker run --rm -v a002:/data
data/
data/LICENSE
data/README.md
data/assets/
data/assets/white_king.png
data/assets/black_rook.png
data/assets/white_pawn.png
data/assets/black_bishop.png
data/assets/black_knight.png
data/assets/black_king.png
data/assets/white_bishop.png
data/assets/white_rook.png
data/assets/black_pawn.png
data/assets/black_queen.png
data/assets/white_knight.png
data/assets/white_queen.png
data/style.css
data/gameController.js
data/bucket_url.txt
data/index.html
```



<u>Task 6: Create a container with a tmpfs mount to store temporary data in memory. Observe how the data is lost when the container stops.</u>

We create a nginx container temporary mount it at path /data inside container using this command

```
docker run  -itd --mount type=tmpfs,target=/data nginx
```

We make a test.txt file echo in it “Hello, temporary data!” and save file within peaceful_kalam

```
PS D:\Devops_FPT_Foudations> docker run  -itd --mount type=tmpfs,target=/data nginx
a69c2d3f1843e337c04a63f22d094c9964c0ebb5e14d4b0580873ab04e174373
PS D:\Devops_FPT_Foudations> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
a69c2d3f1843   nginx     "/docker-entrypoint.…"   58 seconds ago   Up 57 seconds   80/tcp    peaceful_kalam
PS D:\Devops_FPT_Foudations> docker exec -it peaceful_kalam bash
root@a69c2d3f1843:/# ls
bin   data  docker-entrypoint.d   etc   lib    lib64   media  opt   root  sbin sys  usr
boot  dev   docker-entrypoint.sh  home  lib32  libx32  mnt    proc  run   srv  tmp  var
root@a69c2d3f1843:/# cd data
root@a69c2d3f1843:/data# ls
root@a69c2d3f1843:/data# echo "Hello, temporary data!" > test.txt
root@a69c2d3f1843:/data# cat test.txt
Hello, temporary data!
root@a69c2d3f1843:/data#
```

Using same command we create another container with docker name it thirsty_nightingale

```
PS D:\Devops_FPT_Foudations> docker run  -itd --mount type=tmpfs,target=/data nginx
c7210a3007f01d2e55248a2ced429f7a5f680f6beb65bbe381da2a70431ac8aa
PS D:\Devops_FPT_Foudations> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
c7210a3007f0   nginx     "/docker-entrypoint.…"   7 seconds ago   Up 6 seconds   80/tcp    thirsty_nightingale
a69c2d3f1843   nginx     "/docker-entrypoint.…"   7 minutes ago   Up 7 minutes   80/tcp    peaceful_kalam
PS D:\Devops_FPT_Foudations> docker exec -it thirsty_nightingale bash
root@c7210a3007f0:/# ls
bin   data  docker-entrypoint.d   etc   lib    lib64   media  opt   root  sbin sys  usr
boot  dev   docker-entrypoint.sh  home  lib32  libx32  mnt    proc  run   srv  tmp  var
root@c7210a3007f0:/# ls data
root@c7210a3007f0:/#
```

As you see we enter container with same type of tmpfs mount and in data folder there is no test.txt

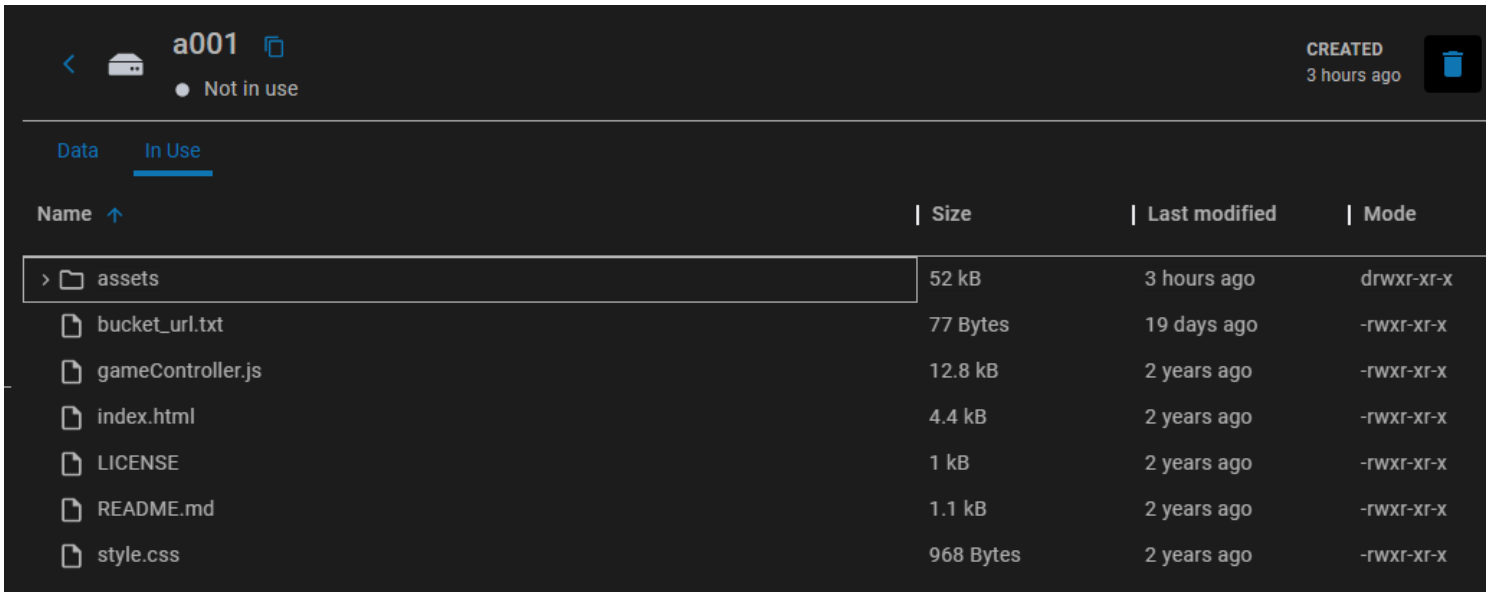
We stop container peaceful_kalam and restart it again

```
PS D:\Devops_FPT_Foudations> docker stop peacefull_kalam
Error response from daemon: No such container: peacefull_kalam
PS D:\Devops_FPT_Foudations> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS          PORTS     NAMES
c7210a3007f0   nginx     "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes    80/tcp    thirsty_nightingale
a69c2d3f1843   nginx     "/docker-entrypoint.…"   12 minutes ago  Up 12 minutes   80/tcp    peaceful_kalam
PS D:\Devops_FPT_Foudations> docker stop peaceful_kalam
peaceful_kalam
PS D:\Devops_FPT_Foudations> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS                      PORTS     NAMES
c7210a3007f0   nginx     "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes                80/tcp    thirsty_nightingale
a69c2d3f1843   nginx     "/docker-entrypoint.…"   12 minutes ago  Exited (0) 17 seconds ago             peaceful_kalam
PS D:\Devops_FPT_Foudations> docker start peaceful_kalam
peaceful_kalam
PS D:\Devops_FPT_Foudations> docker exec -it peaceful_kalam bash
root@a69c2d3f1843:/# ls
bin   data  docker-entrypoint.d   etc   lib    lib64   media  opt   root  sbin sys  usr
boot  dev   docker-entrypoint.sh  home  lib32  libx32  mnt    proc  run   srv  tmp  var
root@a69c2d3f1843:/# ls data
root@a69c2d3f1843:/#
```

And then enter container, the file test.txt we make has alredy gone. This is how tmpfs mount work

<u>Task 7: Write a docker-compose.yml file that defines a service using volumes, then launch multiple containers to share data.</u>

Fist off we will use volume a001 to use with multiple container that will share eachother same data



Here is a docker compose file that will make 3 nginx container with it data will be take from volume a001 and then store in /usr/share/nginx/html folder inside container to make an frontend web

docker > Day_04 > Nginx_test > compose.yaml

```yaml
version: '3'
services:
  nginx1:
    image: nginx:latest
    container_name: nginx-1
    ports:
      - "1000:80"
    volumes:
      - a001:/usr/share/nginx/html
    command: [ "nginx", "-g", "daemon off;" ]
  nginx2:
    image: nginx:latest
    container_name: nginx-2
    ports:
      - "1001:80"
    volumes:
      - a001:/usr/share/nginx/html
    command: [ "nginx", "-g", "daemon off;" ]
  nginx3:
    image: nginx:latest
    container_name: nginx-3
    ports:
      - "1002:80"
    volumes:
      - a001:/usr/share/nginx/html
    command: [ "nginx", "-g", "daemon off;" ]
volumes:
  a001:
    external: true
```

We docker compose up -d to make them run and curl each of nginx port to see result

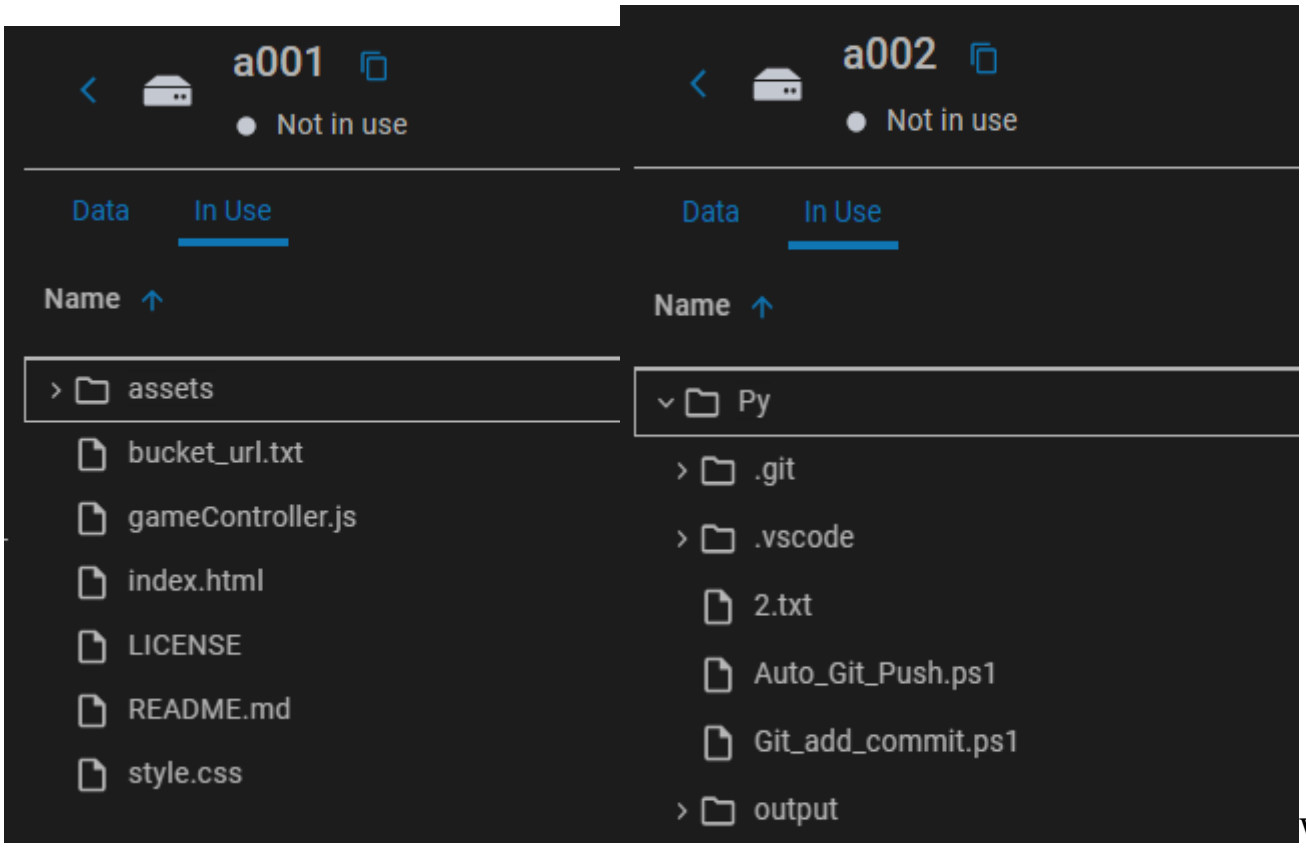
```
PS D:\Devops_FPT_Foudations\docker\Day_04\Nginx_test> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_04\Nginx_test> docker compose up -d
[+] Running 4/4
 ✓ Network nginx_test_default  Created
 ✓ Container nginx-1           Started
 ✓ Container nginx-2           Started
 ✓ Container nginx-3           Started
PS D:\Devops_FPT_Foudations\docker\Day_04\Nginx_test> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                   NAMES
b19c7cc7a8e7   nginx:latest   "/docker-entrypoint.…"   8 seconds ago   Up 5 seconds   0.0.0.0:1002->80/tcp    nginx-3
8c841ea23d64   nginx:latest   "/docker-entrypoint.…"   8 seconds ago   Up 5 seconds   0.0.0.0:1001->80/tcp    nginx-2
19a554d1386e   nginx:latest   "/docker-entrypoint.…"   8 seconds ago   Up 5 seconds   0.0.0.0:1000->80/tcp    nginx-1
PS D:\Devops_FPT_Foudations\docker\Day_04\Nginx_test>
```

So all 3 port take same resource from volume a001 to make web chess game and expose to port

Localhost: 1000, 1001, 1002

Task 8: Launch a container that uses multiple volumes for different parts of its filesystem.

We will use 2 volume a001 and a002 with them own distinctive data



We then run

```
docker run -d --name multi_nginx -v a001:/app1 -v a002:/app2 nginx
```

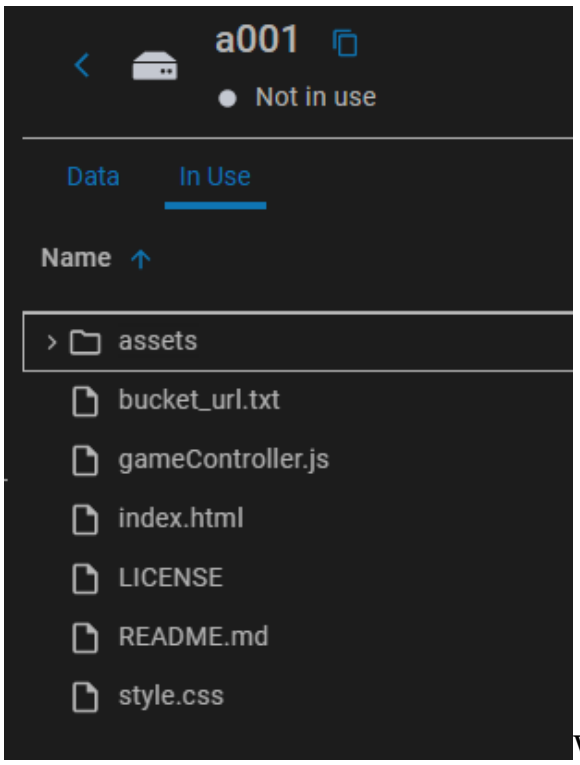
This command line will fist pull and run image nginx:latest to a container name multi_nginx and attach data from volume a001 to app1 folder in container, so on with data from a002 to app2 floder

We then enter container and check if the data from volume a001 and a002 has successfully deploy inside container multi_nginx folder app1, app2.

```
PS D:\Devops_FPT_Foudations> docker run -d --name multi_nginx -v a001:/app1 -v a002:/app2 nginx
40ee3e34b37940bfbdee3bc498b359f485f83c6ed7fc0047302a94868de7bdbf
PS D:\Devops_FPT_Foudations> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
40ee3e34b379   nginx     "/docker-entrypoint.…"   9 seconds ago   Up 7 seconds   80/tcp    multi_nginx
PS D:\Devops_FPT_Foudations> docker exec -it multi_nginx bash
root@40ee3e34b379:/# ls
app1  bin   dev                  docker-entrypoint.sh  home  lib32  libx32  mnt  proc  run   srv  tmp  var
app2  boot  docker-entrypoint.d  etc                   lib   lib64  media   opt  root  sbin  sys  usr
root@40ee3e34b379:/# ls app1
LICENSE  README.md  assets  bucket_url.txt  gameController.js  index.html  style.css
root@40ee3e34b379:/# cd app2
root@40ee3e34b379:/app2# ls Py
2.txt               Py_blank.py        Python_Learn_Day2_3_4         Self-trace-history.txt  output
Auto_Git_Push.ps1   Python_Fast        Python_Learn_Day5_and_Beyone  Self-trace.txt
Git_add_commit.ps1  Python_Learn_Day1  README.md                     User_add_to_Git.ps1
root@40ee3e34b379:/app2#
```

Task 9: Create two containers, migrate data from one to the other using volumes, and ensure minimal downtime.

We fist will using volume a001 mount it to new container we will create nginx01



We then using volume-from to let new container take data

```
runtask9.ps1 M ×   powershell   compose.yaml
docker > Day_04 > task9 > runtask9.ps1
# Start the source container nginx01 with volume a001 mounted to /data
docker run -d --name nginx01 -v a001:/data nginx
# Create the destination container nginx02 using the volume from nginx01
docker run -d --name nginx02 --volumes-from=nginx01 nginx
```

We name our new container nginx02 using volume from nginx01 the path to data will be the same when we create nginx01 and mount it to a001

```
runtask9.ps1 M    powershell ×    compose.yaml

PS D:\Devops_FPT_Foudations\docker\Day_04\task9> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_04\task9> ./runtask9
f3506d40b281393e8f62c8f52c1fd43b064e8eae828f0ad220f8a7f526565baf
290d03dd809390a8d362584b75ea06310911fabafd5e63a42a8f872e257eb410
PS D:\Devops_FPT_Foudations\docker\Day_04\task9> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS     NAMES
290d03dd8093   nginx     "/docker-entrypoint.…"   4 seconds ago   Up 3 seconds   80/tcp    nginx02
f3506d40b281   nginx     "/docker-entrypoint.…"   5 seconds ago   Up 4 seconds   80/tcp    nginx01
PS D:\Devops_FPT_Foudations\docker\Day_04\task9> docker exec -it nginx02 bash
root@290d03dd8093:/# ls
bin   data  docker-entrypoint.d   etc   lib    lib64   media  opt   root  sbin
ys  usr
boot  dev   docker-entrypoint.sh  home  lib32  libx32  mnt    proc  run   srv
mp  var
root@290d03dd8093:/# ls data
LICENSE  README.md  assets  bucket_url.txt  gameController.js  index.html  style.css
root@290d03dd8093:/#
exit
```

Folow the result we have success transfer data floder from nginx01 mount a001 to nginx02

Task 10: Launch multiple instances of a container and share data using the same volume between them.

```
runtask10.ps1 U ×    powershell    exercises.md U    runtask9.ps1 U    Dockerfile U

docker > Day_04 > task10 > runtask10.ps1
# Create volume that will get shared
docker volume create a004
# Create 2 nginx container mount to volume a004
docker run -d --name nginx01 -v a004:/shared-data nginx
docker run -d --name nginx02 -v a004:/shared-data nginx
# Wirte text to volume a004 from nginx01 and check result at nginx02
docker exec -it nginx01 sh -c "echo 'Hello from Nginx01 Bro u got chose' > /shared-data/demo.txt"
docker exec -it nginx02 cat /shared-data/demo.txt
```

We make volume a004 and shared it with 2 continer nginx01 and nginx02

We run the script to auto report us if we have success share data using the volume a004

```
runtask10.ps1 U    powershell ×    exercises.md U    runtask9.ps1 U

PS D:\Devops_FPT_Foudations\docker\Day_04\task10> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_04\task10> ls


    Directory: D:\Devops_FPT_Foudations\docker\Day_04\task10


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----         8/13/2023   6:58 PM            446 runtask10.ps1


PS D:\Devops_FPT_Foudations\docker\Day_04\task10> ./runtask10
a004
8a10c394f4221cb8c07bed863878aeb9e658b20ba74731e24beac03179e7cea2
3722ede5310013beb90aa212d8c6752959b4c59f658bea5166ff2fe951861e9b
Hello from Nginx01 Bro u got chose
PS D:\Devops_FPT_Foudations\docker\Day_04\task10>
```

As nginx02 cat out text we make from nginx01 to volume a004 I said we has finish task 10

Nguyễn Thái Dương Day 5 docker deploy exerise:

1. Deploy a working application:

- Clone ReactJS frontend repo: https://github.com/thai-nm/sample-webapp-reactjs.git

- Clone NodeJS backend repo: https://github.com/uet-app-distributor/sample-nodejs-webapp.git

- Change directory to the frontend repo and build your own image with the pre-provided Dockerfile. You should check for the Dockerfile and its build workflow to understand and review about multi-stage and image build process.

- Change directory to the backend repo and build your own image without seeing the pro-provided Dockerfile. After trying to build your own image or being get stuck, you can use the Dockerfile in the repository as a reference.

- Upload those new images to your own DockerHub account.

- Develop your own `compose.yml` file for Docker Compose to do the following:

  - Create a new network named `deployment-practice-network`

  - Create 2 services:

    - Service `backend` will create a container name `sample-webapp-nodejs` and refer to the backend image you just pushed above. This service requires environment variables: `USER=practioner-be` and `ENV=dev`. This service will only be exposed at port `13000` of the `loopback` interface and mapped to port `3000` of the container.

    - Service `frontend` will create a container name `sample-webapp-reactjs` and refer to the frontend image you just pushed above. This service requires environment variables: `USER=practioner-fe` and `ENV=dev`. This service will be exposed at port `18080` of the `0.0.0.0` interface and mapped to port `80` of the container.

  - Deploy with Docker Compose.

  - To check if the containers are working or not, open your browser and go to http://localhost:18080. Then click `Give me a quote`. If there is a quote sent to you, everything is working well!

Task : 1. Deploy a working application:

- Clone ReactJS frontend repo: https://github.com/thai-nm/sample-webapp-reactjs.git

```
PS D:\Devops_FPT_Foudations\docker\Day_05> git clone https://github.com/thai-nm/sample-webapp-reactjs.git
Cloning into 'sample-webapp-reactjs'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (27/27), done.
```

- Clone NodeJS backend repo: https://github.com/uet-app-distributor/sample-nodejs-webapp.git

```
PS D:\Devops_FPT_Foudations\docker\Day_05> git clone  https://github.com/uet-app-distributor/sample-nodejs-webapp.git
Cloning into 'sample-nodejs-webapp'...
remote: Enumerating objects: 70, done.
remote: Counting objects: 100% (70/70), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 70 (delta 28), reused 59 (delta 19), pack-reused 0
```

Check the 2 file:

```
PS D:\Devops_FPT_Foudations\docker\Day_05> ls


    Directory: D:\Devops_FPT_Foudations\docker\Day_05


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/14/2023   1:25 AM                others
d-----         8/19/2023   4:53 PM                sample-nodejs-webapp
d-----         8/19/2023   4:53 PM                sample-webapp-reactjs
-a----         8/15/2023  10:20 PM            320 .env
-a----         8/16/2023   8:28 PM            558 compose.yaml
------         8/14/2023   1:25 AM           1767 exercises.md
-a----          8/8/2023   3:15 PM          22528 exerise.doc
------         8/14/2023   1:25 AM              0 notes.md


PS D:\Devops_FPT_Foudations\docker\Day_05>
```

- Change directory to the frontend repo and build your own image with the pre-provided Dockerfile. You should check for the Dockerfile and its build workflow to understand and review about multi-stage and image build process. (I change folder name to FE and build image with this docker file)

```
EXPLORER: DEVOPS_F...
FE
  public
  src
  .env
  .eslintrc.cjs
  .gitignore
  Dockerfile
  index.html
  nginx.conf
  package.json
  README.md
  vite.config.js
```

Dockerfile | powershell | exercises.md

docker > Day_05 > FE > Dockerfile > ...

```
FROM node:18 AS build
COPY package.json .
RUN npm install
COPY . .
RUN npm run build


FROM nginx:1.22 AS deploy
WORKDIR /app
COPY --from=build dist /app/
COPY nginx.conf /etc/nginx/nginx.conf
CMD [ "nginx", "-g", "daemon off;" ]
```

Fist I change the .env file to match API public port that I will public on “127.0.0.1:13000”

```
FE
  public
  src
  .env
```

```
# VITE_API_SERVER=http://172.17.0.1:13000
VITE_API_SERVER=http://127.0.0.1:13000
```

I then build fronend image with tag duongtn1512/docker_day5:fe_homework

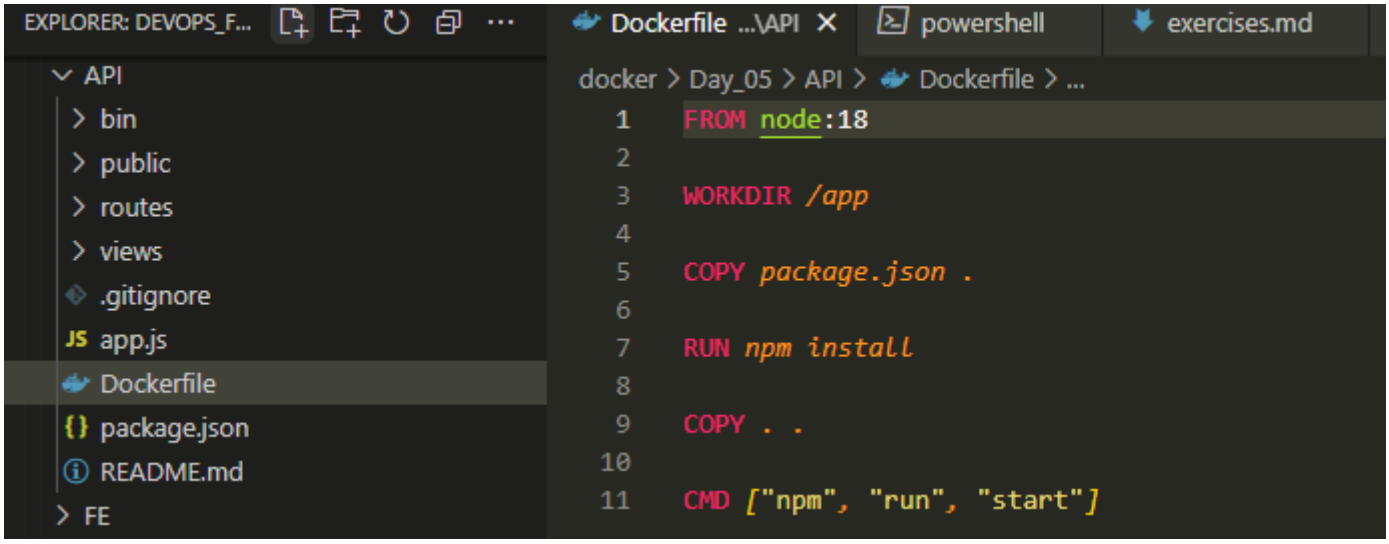
```
PS D:\Devops_FPT_Foudations\docker\Day_05\FE> docker build . -t duongtn1512/docker_day5:fe_homework
[+] Building 203.7s (17/17) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                            0.0s
 => => transferring dockerfile: 281B                                                                            0.0s
 => [internal] load .dockerignore                                                                               0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [internal] load metadata for docker.io/library/nginx:1.22                                                  11.5s
 => [internal] load metadata for docker.io/library/node:18                                                     11.0s
 => [auth] library/node:pull token for registry-1.docker.io                                                     0.0s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                    0.0s
 => [internal] load build context                                                                               0.4s
 => => transferring context: 180.18kB                                                                           0.4s
 => [deploy 1/4] FROM docker.io/library/nginx:1.22@sha256:fc5f5fb7574755c306aaf88456ebfbe0b006420a184d52b923d2f0197108f6b7   0.0s
 => [build 1/5] FROM docker.io/library/node:18@sha256:11e9c297fc51f6f65f7d0c7c8a8581e5721f2f16de43ceff1a199fd3ef609f95   123.2s
```

```
=> => extracting sha256:466ffc744f413d60969cb1ad4000afd4d9833edeb1e7b74d429a01efdbce27a3          0.0s
=> [build 2/5] COPY package.json .                                                                5.8s
=> [build 3/5] RUN npm install                                                                   44.0s
=> [build 4/5] COPY . .                                                                           0.2s
=> [build 5/5] RUN npm run build                                                                  9.6s
=> CACHED [deploy 2/4] WORKDIR /app                                                               0.0s
=> [deploy 3/4] COPY --from=build dist /app/                                                      1.7s
=> [deploy 4/4] COPY nginx.conf /etc/nginx/nginx.conf                                             0.1s
=> exporting to image                                                                             0.1s
=> => exporting layers                                                                            0.1s
=> => writing image sha256:fd690658c5edcd0f911a31adde013cb119d24a2e24e3fb5e2a818325ef41381d       0.0s
=> => naming to docker.io/duongtn1512/docker_day5:fe_homework                                     0.0s
```

Finish the build successfully

- Change directory to the backend repo and build your own image without seeing the pro-provided Dockerfile. After trying to build your own image or being get stuck, you can use the Dockerfile in the repository as a reference.

I change folder hold node application to API and use docker file within the foldor to build



Then build it with tag duongtn1512/docker_day5:api_homework

Here terminal print result

```
PS D:\Devops_FPT_Foudations\docker\Day_05\API> docker build . -t duongtn1512/docker_day5:api_homework
[+] Building 24.7s (11/11) FINISHED                                                    docker:default
 => [internal] load build definition from Dockerfile                                            0.1s
 => => transferring dockerfile: 150B                                                            0.0s
 => [internal] load .dockerignore                                                               0.1s
 => => transferring context: 2B                                                                 0.0s
 => [internal] load metadata for docker.io/library/node:18                                      2.4s
 => [auth] library/node:pull token for registry-1.docker.io                                     0.0s
 => CACHED [1/5] FROM docker.io/library/node:18@sha256:11e9c297fc51f6f65f7d0c7c8a8581e5721f2f16de43ceff1a199fd3ef609f95   0.0s
 => [internal] load build context                                                               0.4s
 => => transferring context: 52.85kB                                                            0.3s
 => [2/5] WORKDIR /app                                                                          0.3s
 => [3/5] COPY package.json .                                                                   0.0s
 => [4/5] RUN npm install                                                                      20.2s
 => [5/5] COPY . .                                                                              0.1s
 => exporting to image                                                                          0.7s
 => => exporting layers                                                                         0.7s
 => => writing image sha256:358ae2d7b10dbadc80757db1a49ac878af63b4e69ea4286c3568bae0df6eb6a6    0.0s
 => => naming to docker.io/duongtn1512/docker_day5:api_homework                                 0.1s

What's Next?
  View summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\Devops_FPT_Foudations\docker\Day_05\API>
```

- Upload those new images to your own DockerHub account.

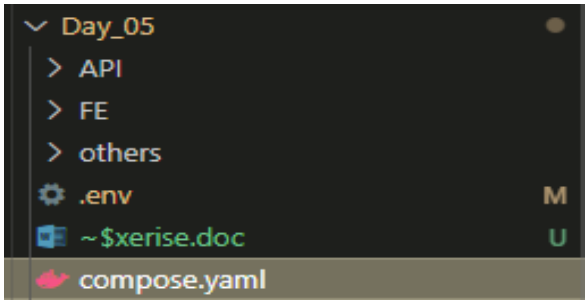
Check for images we have

```
PS D:\Devops_FPT_Foudations\docker\Day_05> docker image ls
REPOSITORY                TAG            IMAGE ID       CREATED              SIZE
duongtn1512/docker_day5   api_homework   358ae2d7b10d   About a minute ago   1.11GB
duongtn1512/docker_day5   fe_homework    fd690658c5ed   10 minutes ago       142MB
```

Push to docker hub

```
PS D:\Devops_FPT_Foudations\docker\Day_05> docker push duongtn1512/docker_day5:api_homework
The push refers to repository [docker.io/duongtn1512/docker_day5]
d9948f82221d: Pushed
769f0e3f79b7: Pushed
896da3650205: Pushed
9e2383af60f1: Pushed
01d1c0599755: Mounted from library/node
18e7f95f640c: Mounted from library/node
092fa1c2d20c: Mounted from library/node
59c677849659: Mounted from library/node
b485c6cd33a6: Mounted from library/node
6aa872026017: Mounted from library/node
43ba18a5eaf8: Mounted from library/node
ff61a9b258e5: Mounted from library/node
api_homework: digest: sha256:e7434fa66d066cb64c41328b7c0674ff189ff25ee1129a219d4e9f5e0f90d5bf size: 2838
PS D:\Devops_FPT_Foudations\docker\Day_05> docker push duongtn1512/docker_day5:fe_homework
The push refers to repository [docker.io/duongtn1512/docker_day5]
ad80b9c6ffb3: Pushed
2c1ac122c7c2: Pushed
9b5558b00a00: Layer already exists
9543dec06aa8: Layer already exists
ccf4f419ba49: Layer already exists
21f8452ebfb1: Layer already exists
25bbf4633bb3: Layer already exists
a4f34e6fb432: Layer already exists
3af14c9a24c9: Layer already exists
fe_homework: digest: sha256:192b3f7c85fb587b91fa1630a3b7ee5a345dc79de5806363f14777028eafe34f size: 2193
PS D:\Devops_FPT_Foudations\docker\Day_05>
```

- Develop your own `compose.yml` file for Docker Compose to do the following:



We wirte our compose file at same folder with env file

- Create a new network named `deployment-practice-network`

```yaml
networks:
  deployment-practice-network:
    driver: bridge
```

- Create 2 services:

  - Service `backend` will create a container name `sample-webapp-nodejs` and refer to the backend image you just pushed above. This service requires environment variables: `USER=practioner-be` and `ENV=dev`. This service will only be exposed at port `13000` of the `loopback` interface and mapped to port `3000` of the container.

```yaml
backend:
  environment:
    - USER=${user_api}
    - env=${env}
  container_name: ${container_api}
  image: ${image_api}
  networks:
    - ${net_day5}
  ports:
    - "${port_api}"
```

```env
env=dev

container_api=API
image_api=duongtn1512/docker_day5:api_homework
user_api=practioner-be
port_api=127.0.0.1:13000:3000

net_day5=deployment-practice-network
```

- Service `frontend` will create a container name `sample-webapp-reactjs` and refer to the frontend image you just pushed above. This service requires environment variables: `USER=practioner-fe` and `ENV=dev`. This service will be exposed at port `18080` of the `0.0.0.0` interface and mapped to port `80` of the container.

```yaml
frontend:
  environment:
    - USER=${user_fe}
    - env=${env}
    - VITE_API_SERVER=${api_for_conect}
  container_name: ${container_fe}
  image: ${image_fe}
  networks:
    - ${net_day5}
  ports:
    - "${port_fe}"
```

```env
env=dev

container_fe=FE
image_fe=duongtn1512/docker_day5:fe_homework
user_fe=practioner-fe
port_fe=127.0.0.1:18080:80

net_day5=deployment-practice-network
```

- Deploy with Docker Compose.

docker > Day_05 > compose.yaml

```yaml
version: '3.4'

networks:
  deployment-practice-network:
    driver: bridge

services:

  frontend:
    environment:
      - USER=${user_fe}
      - env=${env}
      - VITE_API_SERVER=${api_for_conect}
    container_name: ${container_fe}
    image: ${image_fe}
    networks:
      - ${net_day5}
    ports:
      - "${port_fe}"

  backend:
    environment:
      - USER=${user_api}
      - env=${env}
    container_name: ${container_api}
    image: ${image_api}
    networks:
      - ${net_day5}
    ports:
      - "${port_api}"
```

docker > Day_05 > .env

```env
env=dev

container_fe=FE
image_fe=duongtn1512/docker_day5:fe_homework
user_fe=practioner-fe
port_fe=127.0.0.1:18080:80

container_api=API
image_api=duongtn1512/docker_day5:api_homework
user_api=practioner-be
port_api=127.0.0.1:13000:3000

net_day5=deployment-practice-network


api_for_conect=http://127.0.0.1:13000/quote
```

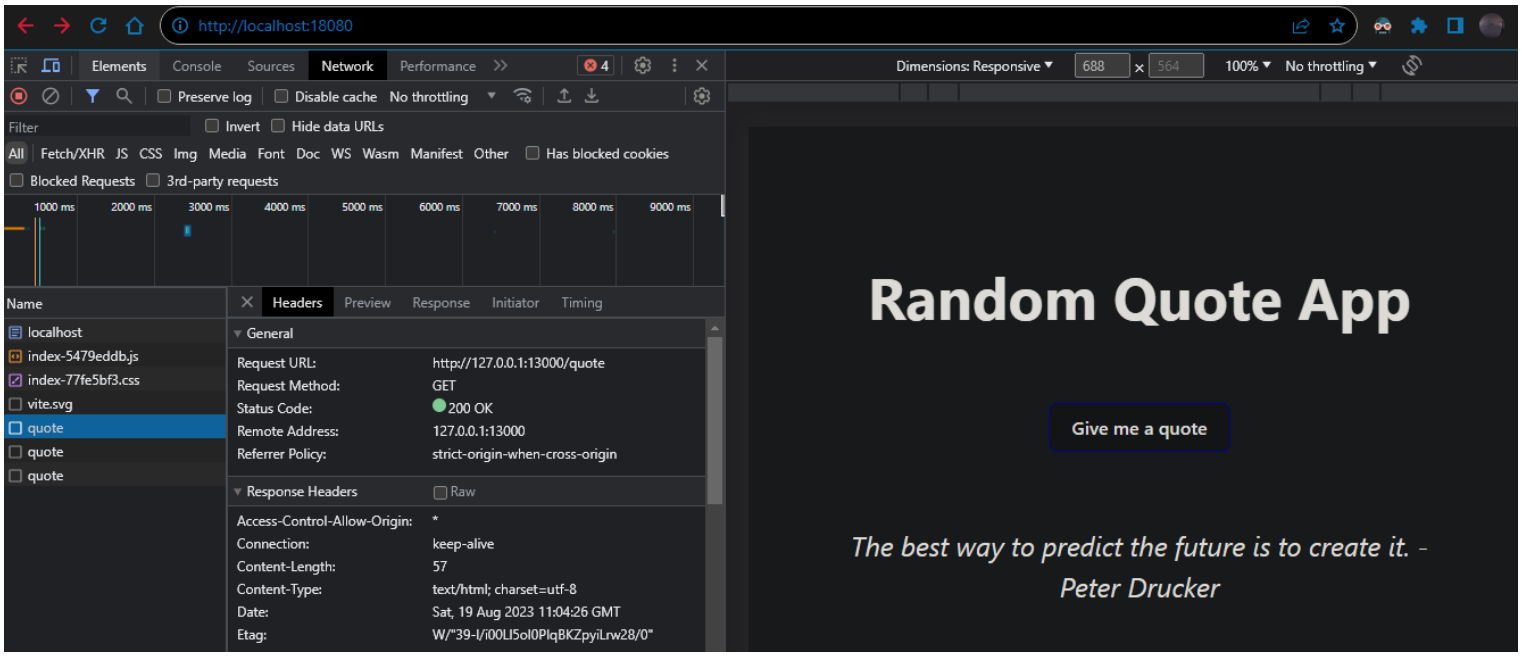
powershell × | Dockerfile | exercises.md | .env M | compose.yaml

```
PS D:\Devops_FPT_Foudations\docker\Day_05> docker compose up -d
[+] Running 3/3
 ✓ Network day_05_deployment-practice-network  Created                    0.7s
 ✓ Container FE                                Started                    2.1s
 ✓ Container API                               Started                    2.1s
PS D:\Devops_FPT_Foudations\docker\Day_05> docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS         PORTS                      NAMES
df401c147d98   duongtn1512/docker_day5:fe_homework  "/docker-entrypoint.…"   8 seconds ago  Up 5 seconds   127.0.0.1:18080->80/tcp    FE
32ba7465dda0   duongtn1512/docker_day5:api_homework "docker-entrypoint.s…"  8 seconds ago  Up 5 seconds   127.0.0.1:13000->3000/tcp  API
PS D:\Devops_FPT_Foudations\docker\Day_05>
```

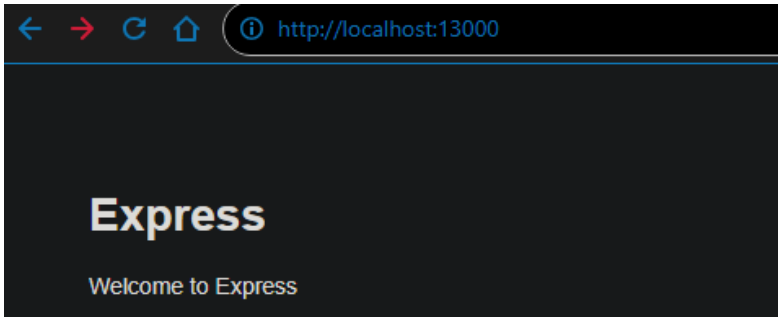
We docker compose up and success run 2 container and a network

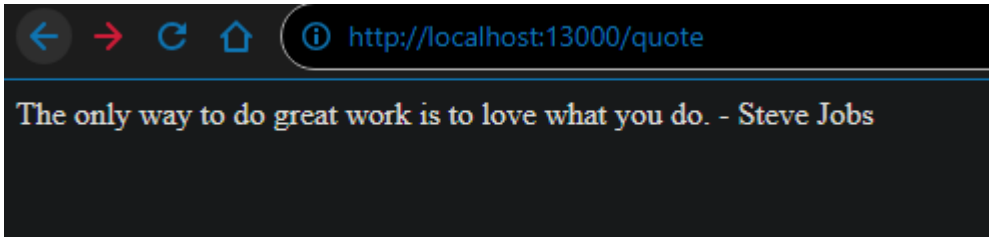
- To check if the containers are working or not, open your browser and go to http://localhost:18080. Then click `Give me a quote`. If there is a quote sent to you, everything is working well!

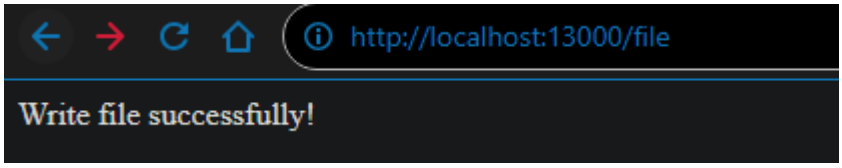
Frontend with return network status code 200 and can get the quote



Back end api result







Nguyễn Thái Dương Day 6 docker compose exerise:

Task: Viết và run 1 docker-compose file chạy 2 service là wordpress và database bất kỳ. Check web bằng cách tương tác thử với app

Chúng ta sẽ viết 1 file docker compose bao gồm 3 service : mysql, phpmyadmin, wordpress

Chúng ta sẽ public cổng 1001 cho php và 1002 cho wordpress, try cập vào trình duyệt web và tương tác với ứng dụng chạy trên container với config theo docker-compose sau:

Database config

Phpmyadmin config

```yaml
version: '3.8'

services:
  # Data base config
  database:
    container_name: ${db_container}
    image: ${db_image}
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${db_root_pass}
      MYSQL_DATABASE: ${db_database}
      MYSQL_USER: ${db_user}
      MYSQL_PASSWORD: ${db_pass}
    expose:
      - 3306
    volumes:
      - ${db_volume}:/var/lib/mysql
    networks:
      - ${net}
```

```yaml
  # I thinks this will database interface
  phpmyadmin:
    depends_on:
      - database
    image: ${php_image}
    container_name: ${php_con}
    restart: always
    ports:
      - "${php_port}"
    environment:
      PMA_HOST: database
      MYSQL_ROOT_PASSWORD: ${db_root_pass}
      MYSQL_USER: ${db_user}
      MYSQL_PASSWORD: ${db_pass}
    networks:
      - ${net}
```

Wordpress config. Storage and network

```yaml
  # FE container that will be user interface
  wordpress:
    depends_on:
      - database
    container_name: ${fe_container}
    image: ${fe_image}
    ports:
      - "${fe_port}"
    restart: always
    environment:
      WORDPRESS_DB_HOST: database
      WORDPRESS_DB_USER: ${db_user}
      WORDPRESS_DB_PASSWORD: ${db_pass}
      WORDPRESS_DB_NAME: ${db_database}
      APACHE_SERVER_NAME: ${apache_server_name}
    volumes:
      - ${fe_volume}:/var/www/html
    networks:
      - ${net}

networks:
  a01:
volumes:
  wordpress_data:
  db_data:
```

File env chứa biến môi trường của 3 service

.env M

docker > Day_06 > .env

```env
db_container=DB
db_image=mysql:latest
db_port=1000:3306
db_root_pass=rootpassword
db_database=mysql
db_user=mysql
db_pass=mysql
db_volume=db_data

php_image=phpmyadmin:latest
php_port=1001:80
php_con=PHP

fe_container=FE
fe_image=wordpress:latest
fe_port=1002:80
fe_volume=wordpress_data
apache_server_name=localhost

net=a01
```

Kiểm tra xem có container nào đang chạy không và chắc chắn rằng thư mục compose.yaml và thư mục .env nằm trên cùng đường dẫn

```
powershell ×   compose.yaml U   .env M   docker-compose.yml

PS D:\Devops_FPT_Foudations\docker\Day_06> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_06> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
PS D:\Devops_FPT_Foudations\docker\Day_06> ls


    Directory: D:\Devops_FPT_Foudations\docker\Day_06


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         8/16/2023   8:03 PM                wordpress.conf
d-----         6/22/2021   8:05 PM                WP-MYSQL
-a----         8/19/2023   3:54 PM            344 .env
-a----         8/19/2023   3:04 PM           1383 compose.yaml
-a----          8/8/2023   3:15 PM          22528 exerise.doc
-a----         8/16/2023   6:38 PM           4685 user_aws_display_fpt.txt


PS D:\Devops_FPT_Foudations\docker\Day_06>
```

Kiểm tra docker image, network và volume:

```
PS D:\Devops_FPT_Foudations\docker\Day_06> docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
phpmyadmin   latest    00d1bd49dd01   33 hours ago   562MB
wordpress    latest    ed7281630c77   2 days ago     666MB
mysql        latest    99afc808f15b   8 days ago     577MB
mysql        8.0.27    3218b38490ce   20 months ago  516MB
PS D:\Devops_FPT_Foudations\docker\Day_06> docker volume ls
DRIVER    VOLUME NAME
local     a001
local     a004
PS D:\Devops_FPT_Foudations\docker\Day_06> docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
ad0a78ca2068   bridge            bridge    local
a2cec830f010   docker_gwbridge   bridge    local
eeb939a64378   host              host      local
24ml2wzapgd7   ingress           overlay   swarm
fb9c40f5fca2   none              null      local
82eb06763a3e   test_demo_net_1   bridge    local
PS D:\Devops_FPT_Foudations\docker\Day_06>
```

Chúng ta thấy chúng ta đã kéo 3 image base của 3 service cần chạy, không có network và volume tương ứng với tên của volume chúng sẽ tạo ra khi chạy file docker compose

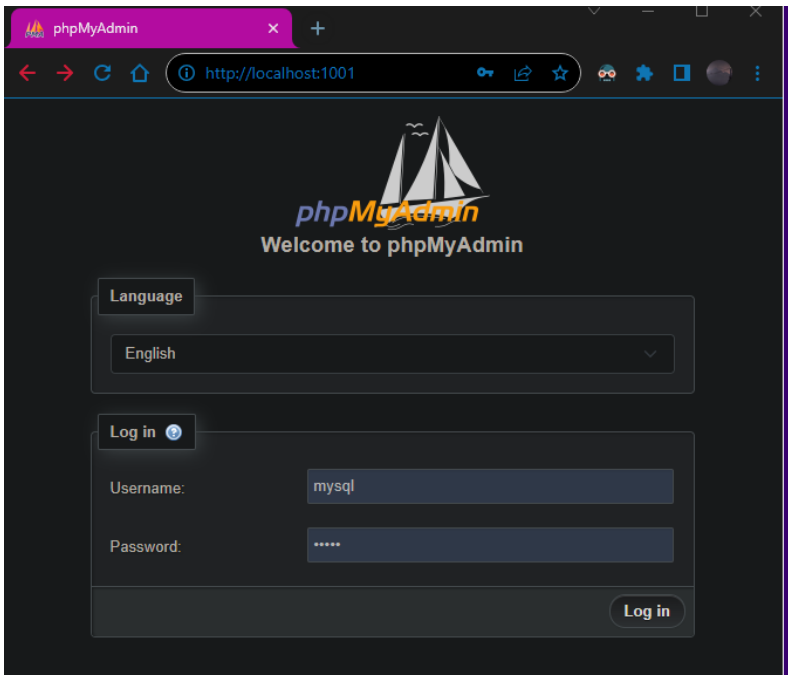
Chạy file docker compose bằng lệnh docker compose up -d (để docker không in log lên terminal)

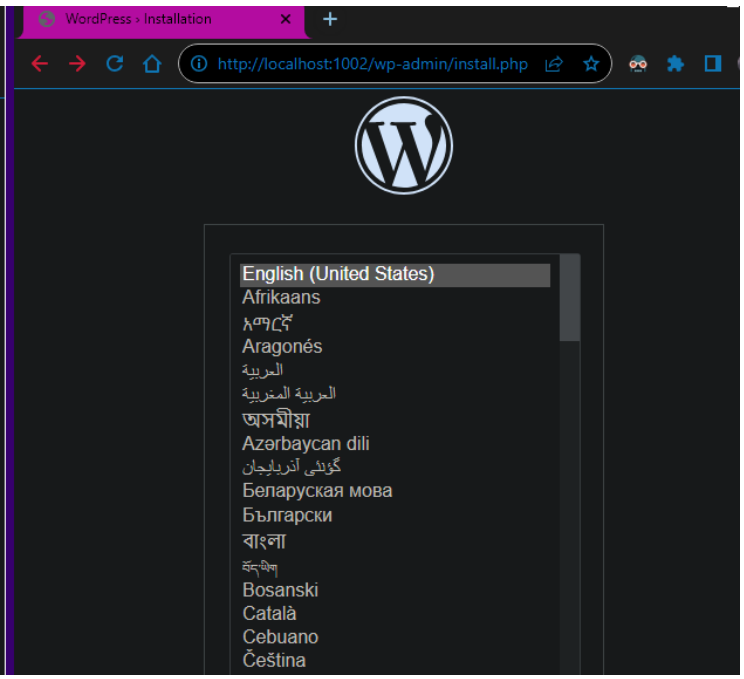
```
PS D:\Devops_FPT_Foudations\docker\Day_06> docker compose up -d
[+] Running 6/6
 ✓ Network day_06_a01            Created                                    0.8s
 ✓ Volume "day_06_db_data"       Created                                    0.0s
 ✓ Volume "day_06_wordpress_data" Created                                   0.0s
 ✓ Container DB                  Started                                    1.4s
 ✓ Container FE                  Started                                    3.1s
 ✓ Container PHP                 Started                                    2.9s
PS D:\Devops_FPT_Foudations\docker\Day_06>
```

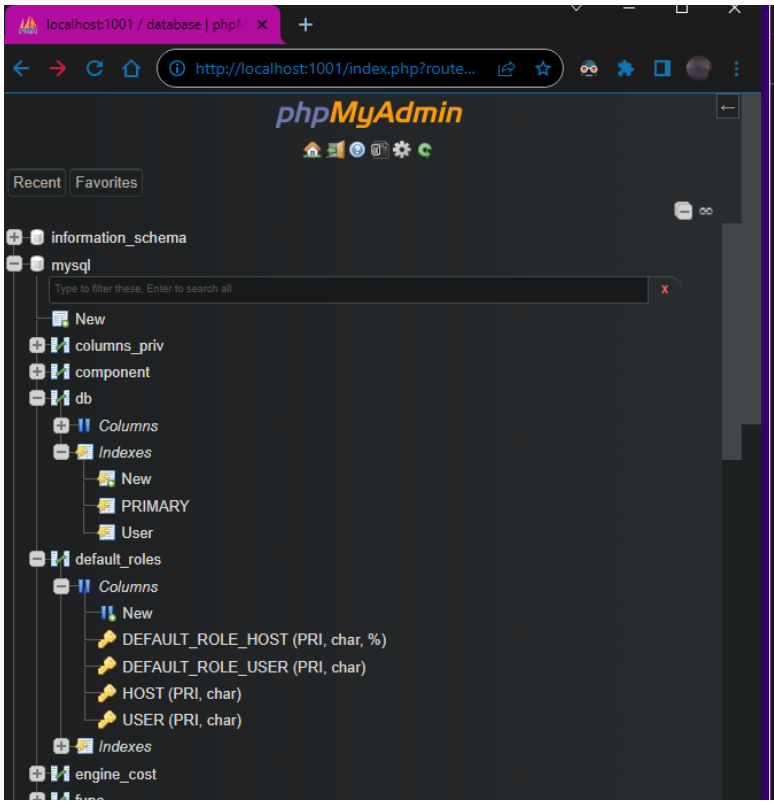
Theo kết quả in chúng ta đã dựng lên thành công 3 container tương ứng với 3 service cần chạy và tạo được 2 volume dùng cho workpress, mysql và 1 network dùng chung của 3 service đó

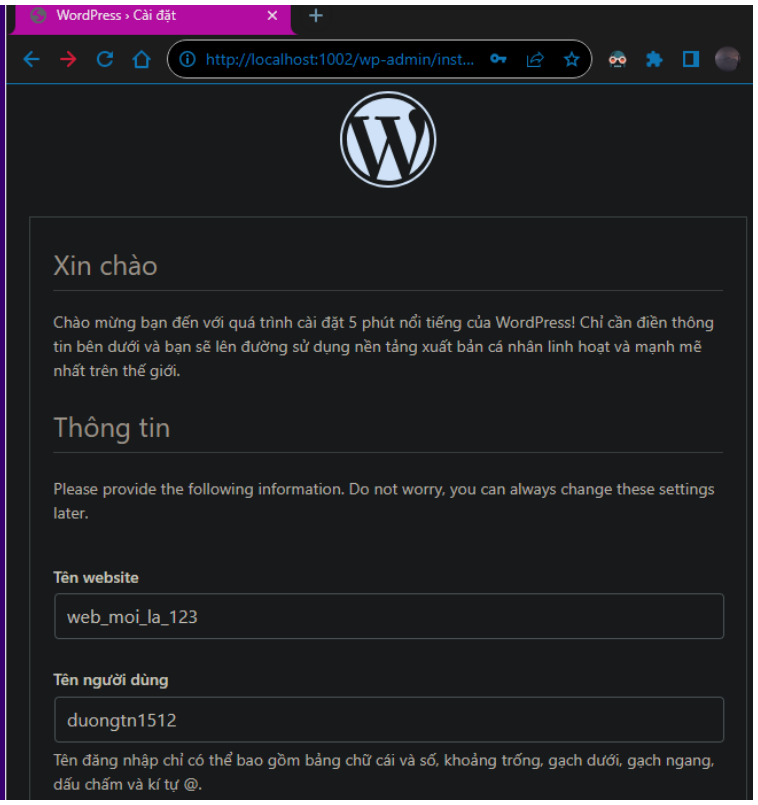
```
PS D:\Devops_FPT_Foudations\docker\Day_06> docker ps -a
CONTAINER ID   IMAGE              COMMAND                  CREATED        STATUS         PORTS                    NAMES
7e94afe77d19   phpmyadmin:latest  "/docker-entrypoint.…"   2 minutes ago  Up 2 minutes   0.0.0.0:1001->80/tcp     PHP
31a5765cb98d   wordpress:latest   "docker-entrypoint.s…"   2 minutes ago  Up 2 minutes   0.0.0.0:1002->80/tcp     FE
39036b7b49a1   mysql:latest       "docker-entrypoint.s…"   2 minutes ago  Up 2 minutes   3306/tcp, 33060/tcp      DB
PS D:\Devops_FPT_Foudations\docker\Day_06>
```

Chúng ta thấy 3 conatainer đang chạy và truy cập vào 2 web site chúng ta đã public cổng 1001 1002









Theo như kết quả hiển thị vậy chúng ta đã hoàn thành xong Task việc còn lại là của backend và frontend dev