

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**  
**KHOA CÔNG NGHỆ THÔNG TIN 2**  
---o0o---



**BÁO CÁO GIỮA KỲ & CUỐI KỲ**  
**TÀI LIỆU HƯỚNG DẪN THIẾT LẬP**  
**TRIỂN KHAI PHẦN MỀM**

**Môn học: Chuyên đề Công nghệ phần mềm**

**Giảng viên hướng dẫn: ThS. Lê Hà Thanh**

**Sinh viên thực hiện: Nguyễn Phan Nhựt Trường**

**MSSV: N20DCCN082**

**Lớp: D20CQCNPM01-N**

*TP.HCM, tháng 06 năm 2024*



## LỜI NÓI ĐẦU

Sau khoảng thời gian nghiên cứu và học tập một cách nghiêm túc, em đã hoàn thành xong chủ đề môn học của mình. Lời đầu tiên em xin gửi lời cảm ơn tới thầy **Lê Hà Thanh** đã dìu dắt và truyền đạt kiến thức để em hoàn thành tốt chủ đề này, cảm ơn những góp ý của thầy đã giúp nhóm em vấn đề còn tồn tại và kịp thời sửa chữa.

Dù đã rất cố gắng, song em cũng không thể tránh khỏi những sai sót do vốn kiến thức còn hạn hẹp. Em rất mong nhận được sự góp ý, giúp đỡ từ thầy và các bạn.

Em xin chân thành cảm ơn!

## NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

*Tp.HCM, ngày tháng năm 2024*

*(Ký và ghi rõ họ tên)*

## Mục lục

LỜI NÓI ĐẦU .....	2
NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN .....	3
I. Giới thiệu .....	5
1. Giới thiệu về tài liệu .....	5
2. Công nghệ sử dụng .....	5
3. Yêu cầu hệ thống .....	5
3.1. Yêu cầu phần cứng .....	5
3.2 Yêu cầu phần mềm .....	5
II. Triển khai phần mềm .....	6
1. IDE/Text Editor .....	6
2. Hệ quản trị phiên bản - Git .....	6
3. Docker Desktop .....	6
4. Pull sản phẩm về máy .....	6
5. Sử dụng Visual Studio Code để triển khai phần mềm .....	7
5.1. Mở Folder project trong Visual Studio Code .....	7
5.2. Thêm file biến môi trường: .....	8
5.3. Giải thích folder/file trong dự án .....	8
5.4. Khởi chạy dự án .....	14
6. Triển khai CMS Wordpress .....	17
7. Triển khai cài đặt E-Commerce (WooCommerce) .....	19
III. Triển khai CI/CD .....	22
IV. Triển khai Reverser Proxy bằng Nginx .....	28

## **I. Giới thiệu**

### **1. Giới thiệu về tài liệu**

- Tài liệu này dùng để:

- Triển khai cài đặt website sử dụng kỹ thuật **Reverse Proxy** bằng **Nginx**.
- Triển khai cài đặt **CMS (WordPress blogs)** và cài đặt **E-commerce (WooCommerce)**.
- Triển khai sử dụng **CI/CD** bằng **Github Actions**.

### **2. Công nghệ sử dụng**

- Hệ quản trị nội dung mã nguồn mở (CMS - web content management system): Wordpress.
- E-Commerce mã nguồn mở: WooCommerce.
- Cơ sở dữ liệu: MySQL
- Reverse Proxy Server: Nginx
- Run-time Enviroment: Docker Desktop.
- Git: Github
- CI/CD: GitHub Actions
- Text Editor: Visual Studio Code.

### **3. Yêu cầu hệ thống**

#### **3.1. Yêu cầu phần cứng**

- Hệ điều hành: Windows 10/11 64-bit.
- Dung lượng RAM: tối thiểu 4GB.
- Dung lượng ổ cứng: tối thiểu 10GB.
- Kích hoạt ảo hóa phần cứng (hardware virtualization) trong BIOS.
- Bật được tính năng WSL 2 trên Windows.

#### **3.2 Yêu cầu phần mềm**

- Text Editor: Visual Studio Code
- Cài đặt sẵn Docker Desktop.
- Hệ quản trị phiên bản: Git
- Trình duyệt: Microsoft Edge
- Hệ quản trị CSDL: MySQL Workbench hoặc phpMyAdmin (kèm theo Xampp)

## II. Triển khai phần mềm

### 1. IDE/Text Editor

- Để thuận tiện ta sử dụng Visual Studio Code để dễ dàng cài đặt.
- Tải Visual Studio Code: <https://code.visualstudio.com/download>

### 2. Hệ quản trị phiên bản - Git

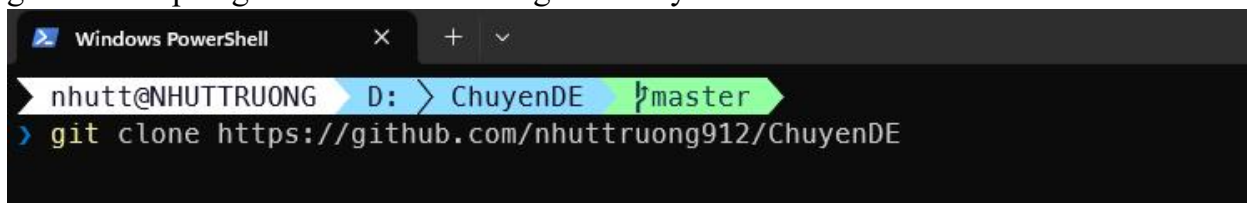
- Có thể sử dụng các câu lệnh của Git.
- Tải Git: <https://git-scm.com/downloads>

### 3. Docker Desktop

- Link hướng dẫn cài đặt Docker Desktop trên windows tham khảo tại: <https://viblo.asia/p/cai-dat-docker-tren-windows-10-3Q75w6gelWb>
- Cài đặt Docker Desktop: <https://docs.docker.com/desktop/install/windows-install/>

### 4. Pull sản phẩm về máy

- Chọn folder để clone project từ Github về máy.
- Tại folder vừa chọn mở WindowsPowerShell hoặc GitBash gõ lệnh:  
git clone https://github.com/nhuttruong912/ChuyenDE



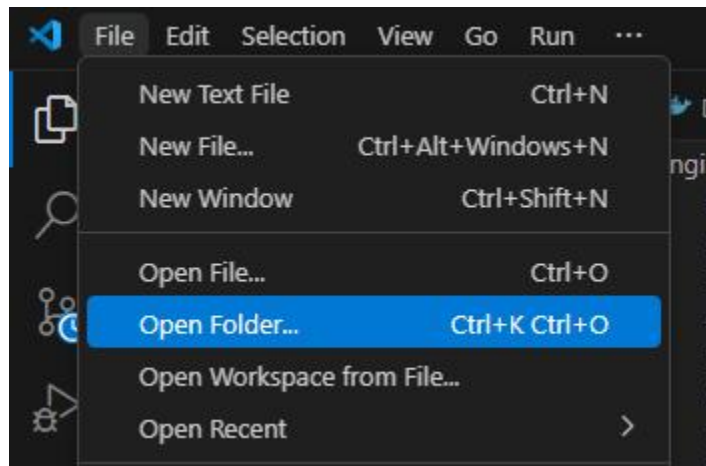
*Ảnh 1: Clone project từ Github*

## 5. Sử dụng Visual Studio Code để triển khai phần mềm

### 5.1. Mở Folder project trong Visual Studio Code

Cách 1:

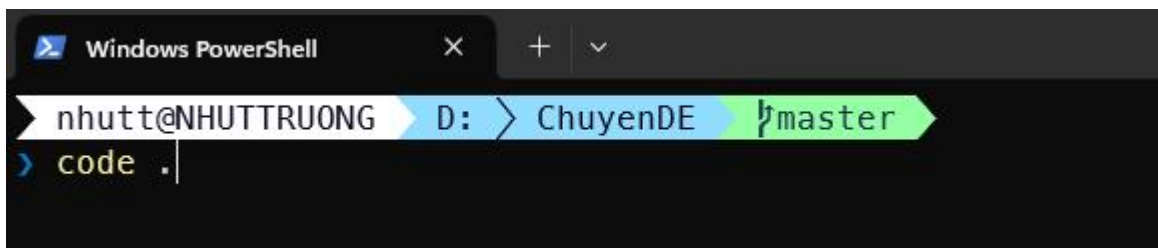
Bật Visual Studio Code, ấn File -> Open Folder rồi chọn Folder đã chọn trong máy



*Ảnh 2. Mở Folder dự án trong Visual Studio Code*

Cách 2:

Tại thư mục gốc của project, mở WindowsPowerShell, gõ lệnh: `code .`



*Ảnh 3. Mở Folder dự án bằng terminal.*



## 5.2. Thêm file biến môi trường:

Tại thư mục gốc của dự án, tạo một file mới có tên **.env** và nội dung bên trong gồm có:

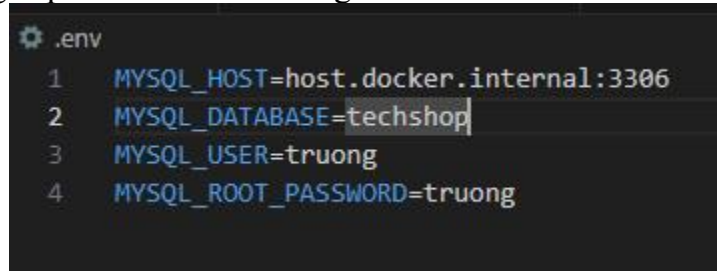
**MYSQL\_HOST**=host.docker.internal:3306

**MYSQL\_DATABASE**=techshop

**MYSQL\_USER**=truong

**MYSQL\_ROOT\_PASSWORD**=truong

File **.env** để cung cấp các biến môi trường để kết nối đến database.

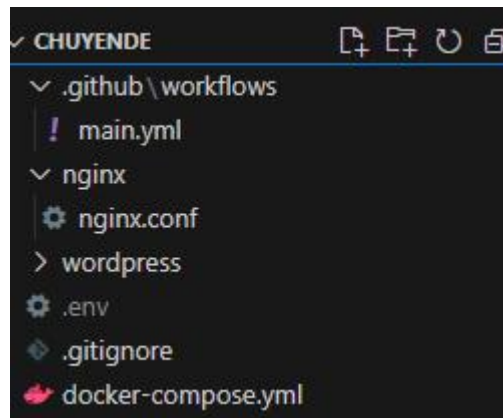


```
.env
1  MYSQL_HOST=host.docker.internal:3306
2  MYSQL_DATABASE=techshop
3  MYSQL_USER=truong
4  MYSQL_ROOT_PASSWORD=truong
```

*Ảnh 4. Nội dung file .env*

## 5.3. Giải thích folder/file trong dự án

- Cấu trúc thư mục dự án:



*Ảnh 5. Cấu trúc thư mục dự án*

- Thư mục **.github\workflows** chứa file **main.yml** dùng để cấu hình CI/CD trên *Github actions*.
- Thư mục **nginx** chứa file **nginx.conf** để cấu hình reverse proxy server.

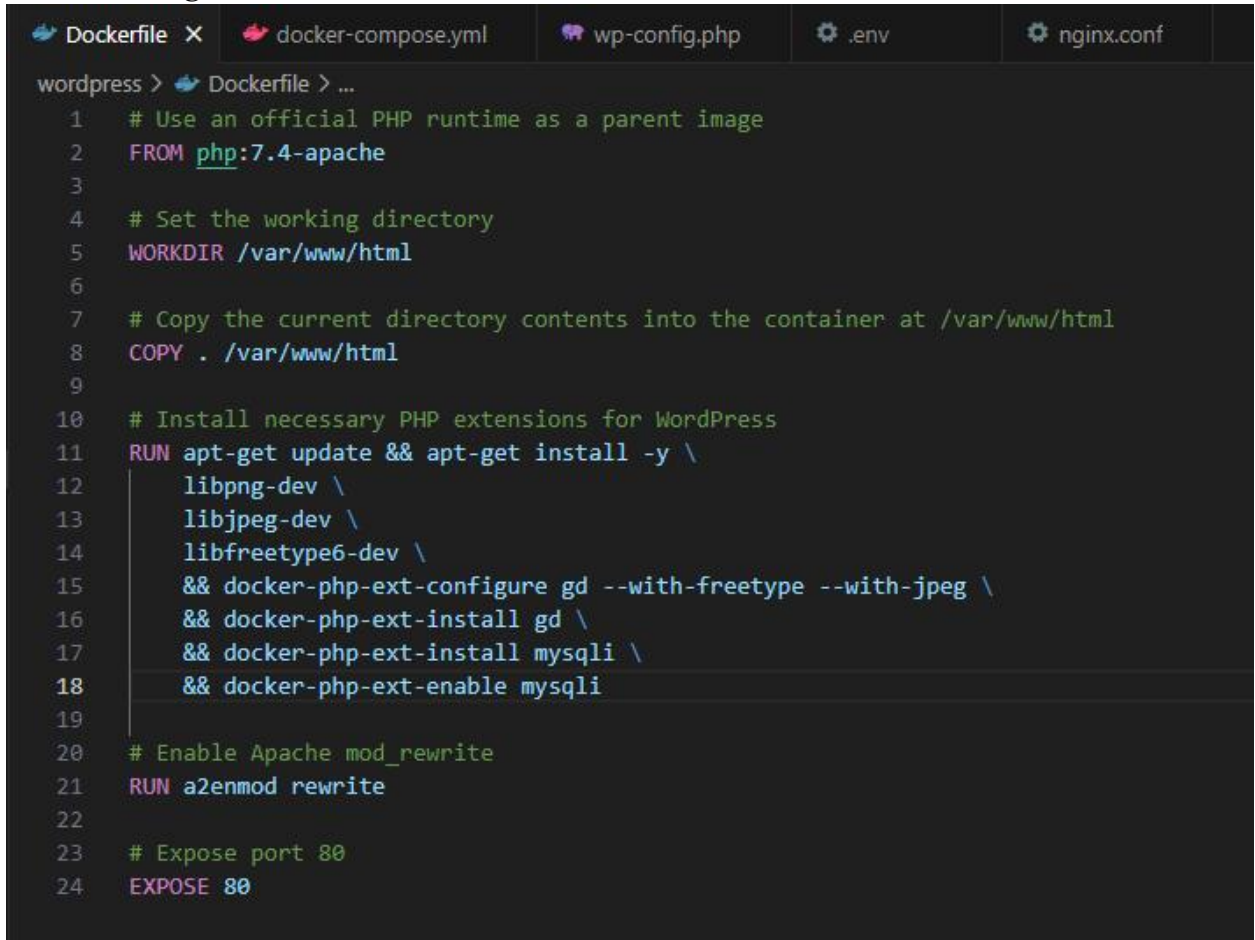
```

nginx > nginx.conf
1  events {
2      worker_connections 1024;
3  }
4
5  http {
6      include mime.types;
7      default_type application/octet-stream;
8
9      sendfile on;
10     keepalive_timeout 65;
11     server {
12         # Xác định cổng (port)
13         listen 80;
14         # Xác định tên miền hoặc địa chỉ IP của máy chủ web
15         server_name localhost; # yourdomain.com
16
17         # Xác định thư mục chứa mã nguồn của website
18         location / {
19             # Khi nginx nhận yêu cầu, nó sẽ chuyển yêu cầu đến dịch vụ "localhost trên cổng 8080
20             proxy_pass http://localhost:8080;
21
22             #
23             proxy_set_header Host $host;
24
25             # Xác định địa chỉ IP của người dùng
26             proxy_set_header X-Real-IP $remote_addr;
27
28             #
29             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
30         }
31     }
32 }

```

Ảnh 6. File *nginx.conf* dùng để cấu hình reverse proxy server

- Thư mục **wordpress** có chứa các mã nguồn của WordPress và **Dockerfile** hỗ trợ *build image trên docker*.



```
wordpress > Dockerfile > ...
1  # Use an official PHP runtime as a parent image
2  FROM php:7.4-apache
3
4  # Set the working directory
5  WORKDIR /var/www/html
6
7  # Copy the current directory contents into the container at /var/www/html
8  COPY . /var/www/html
9
10 # Install necessary PHP extensions for WordPress
11 RUN apt-get update && apt-get install -y \
12     libpng-dev \
13     libjpeg-dev \
14     libfreetype6-dev \
15     && docker-php-ext-configure gd --with-freetype --with-jpeg \
16     && docker-php-ext-install gd \
17     && docker-php-ext-install mysqli \
18     && docker-php-ext-enable mysqli
19
20 # Enable Apache mod_rewrite
21 RUN a2enmod rewrite
22
23 # Expose port 80
24 EXPOSE 80
```

*Ảnh 7: File Dockerfile dùng để hỗ trợ build image trên Docker*

- *FROM php:7.4-apache*: Dòng này chỉ định rằng chúng ta sẽ sử dụng image chính thức của PHP phiên bản 7.4 kèm với máy chủ Apache làm image gốc.
- *WORKDIR /var/www/html*: Dòng này thiết lập thư mục làm việc mặc định bên trong container là /var/www/html. Mọi lệnh COPY, ADD, RUN, v.v., tiếp theo sẽ được thực hiện trong thư mục này.
- *COPY . /var/www/html*: Dòng này copy toàn bộ nội dung của thư mục hiện tại (nơi Dockerfile nằm) trên máy host vào thư mục /var/www/html bên trong container.
- *\*\*RUN apt-get update && apt-get install -y \*\**: Dòng này cập nhật danh sách package và cài đặt các thư viện cần thiết.
- *libpng-dev, libjpeg-dev, libfreetype6-dev*: Các thư viện cần thiết cho phần mở rộng gd để xử lý hình ảnh.

- ***\*\*docker-php-ext-configure gd --with-freetype --with-jpeg\*\****: Cấu hình phần mở rộng gd để hỗ trợ định dạng FreeType và JPEG.
- ***\*\*docker-php-ext-install gd\*\****: Cài đặt phần mở rộng gd.
- ***\*\*docker-php-ext-install mysqli\*\****: Cài đặt phần mở rộng mysqli, cần thiết cho WordPress để kết nối với MySQL.
- ***docker-php-ext-enable mysqli***: Kích hoạt phần mở rộng mysqli.
- ***RUN a2enmod rewrite***: Dòng này kích hoạt module mod\_rewrite của Apache, cần thiết để hỗ trợ các URL đẹp và cấu hình .htaccess của WordPress.
- ***EXPOSE 80***: Dòng này chỉ định rằng container sẽ lắng nghe trên cổng 80. Đây là cổng mặc định cho HTTP, cho phép các kết nối HTTP tới container

- File **docker-compose.yml** hỗ trợ *quản lý cấu trúc ứng dụng và triển khai lên docker*.

```
nginx:
  image: nginx:latest
  restart: always
  ports:
    - "80:80"
  volumes:
    - ./nginx:/etc/nginx/conf.d
    - ./wordpress:/var/www/html
  networks:
    - app-network
```

*Ảnh 8. Cấu hình nginx server trong file docker-compose.yml để chạy trên docker container*

- *nginx*: Đây là tên của dịch vụ trong docker-compose.yml. Bạn có thể dùng bất kỳ tên nào bạn muốn.
- *image: nginx*  
: Sử dụng image chính thức của Nginx với tag latest từ Docker Hub. Image này sẽ được kéo từ Docker Hub nếu nó chưa tồn tại trên máy.
- *restart: always*: Chỉ định chính sách khởi động lại cho container. always có nghĩa là container sẽ luôn được khởi động lại nếu nó dừng lại, bất kể nguyên nhân nào.
- *ports*: Định nghĩa ánh xạ cổng giữa máy host và container.
- *"80:80"*: Cổng 80 trên máy host được ánh xạ tới cổng 80 trên container. Điều này có nghĩa là bạn có thể truy cập dịch vụ Nginx thông qua cổng 80 trên máy host.
- *volumes*: Định nghĩa các volume để gắn kết thư mục trên máy host vào container.
- *./nginx:/etc/nginx/conf.d*: Thư mục ./nginx trên máy host được gắn vào thư mục /etc/nginx/conf.d trong container. Thư mục này chứa các file cấu hình Nginx.
- *./wordpress:/var/www/html*: Thư mục ./wordpress trên máy host được gắn vào thư mục /var/www/html trong container. Thư mục này chứa mã nguồn của ứng dụng WordPress.
- *networks*: Định nghĩa các mạng mà dịch vụ sẽ tham gia.

- *app-network*: Dịch vụ này sẽ tham gia vào mạng tên là app-network. Bạn cần định nghĩa mạng này trong file docker-compose.yml hoặc Docker sẽ tự động tạo ra mạng này nếu nó chưa tồn tại.

```
wordpress:
# Trường hợp muốn build image từ local
# build:
#   context: ./wordpress

# Trường hợp muốn lấy image từ docker repository
image: nhuttruong912/demo_wordpress_cicd:latest
container_name: wordpress
restart: always
env_file: .env
ports:
- "8080:80"
environment:
  WORDPRESS_DB_HOST: ${MYSQL_HOST}
  WORDPRESS_DB_USER: ${MYSQL_USER}
  WORDPRESS_DB_PASSWORD: ${MYSQL_ROOT_PASSWORD}
  WORDPRESS_DB_NAME: ${MYSQL_DATABASE}
volumes:
- ./wordpress:/var/www/html
networks:
- app-network
```

Ảnh 9. Cấu hình chạy wordpress trong file docker-compose.yml để chạy trên docker container

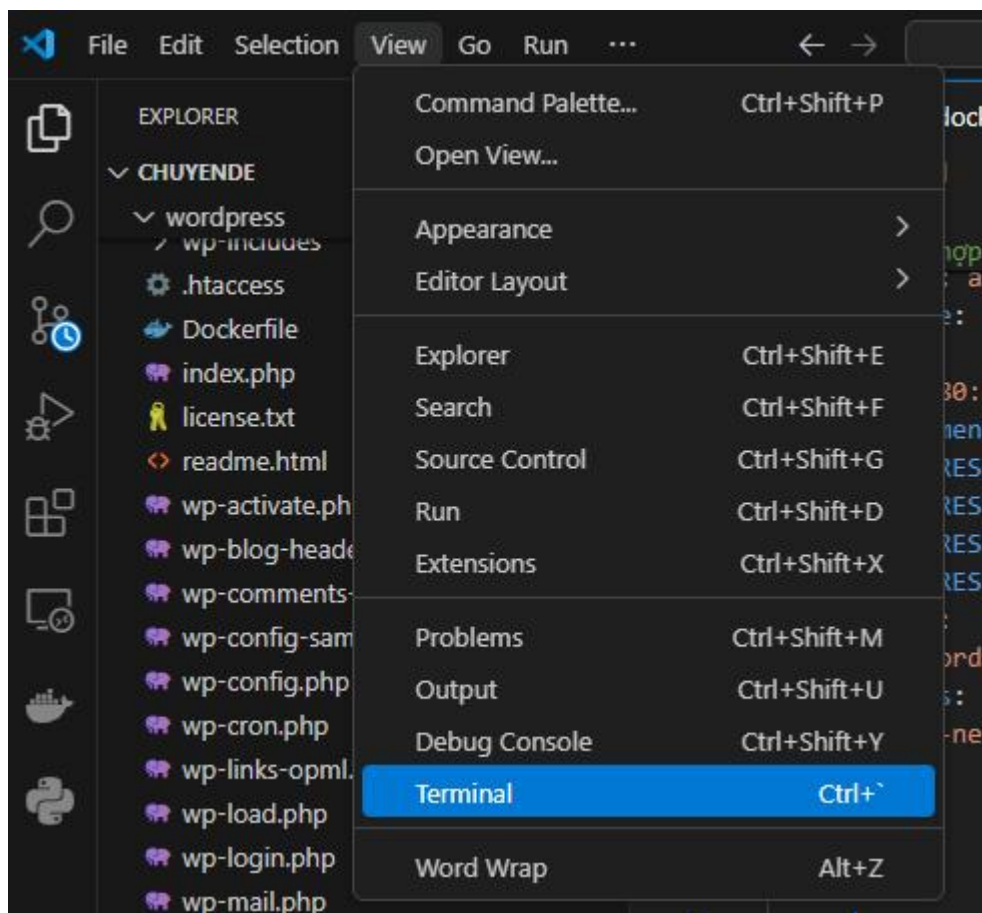
- *wordpress*: Đây là tên của dịch vụ trong docker-compose.yml. Bạn có thể dùng bất kỳ tên nào bạn muốn.
- *build*: Thay vì sử dụng một image có sẵn, bạn có thể build image từ mã nguồn local. Dòng này được comment lại, nhưng nếu bạn muốn sử dụng, bạn có thể uncomment.
- *context: ./wordpress*: Chỉ định thư mục context là ./wordpress. Docker sẽ sử dụng Dockerfile trong thư mục này để build image.
- *image*: Sử dụng image từ Docker Hub hoặc một Docker registry khác.
- *nhuttruong912/demo\_wordpress\_cicd*: Đây là tên của image và tag mà bạn muốn sử dụng.

- *container\_name*: Đặt tên cho container là wordpress. Điều này giúp dễ dàng nhận diện container khi bạn sử dụng các lệnh Docker khác nhau.
- *restart: always*: Chỉ định chính sách khởi động lại cho container. *always* có nghĩa là container sẽ luôn được khởi động lại nếu nó dừng lại, bất kể nguyên nhân nào.
- *env\_file*: Chỉ định file *.env* chứa các biến môi trường cần thiết cho dịch vụ. Các biến môi trường này sẽ được Docker Compose đọc và sử dụng.
- *ports*: Định nghĩa ánh xạ cổng giữa máy host và container.
- *"8080:80"*: Cổng 8080 trên máy host được ánh xạ tới cổng 80 trên container. Điều này có nghĩa là bạn có thể truy cập dịch vụ WordPress thông qua cổng 8080 trên máy host.
- *environment*: Định nghĩa các biến môi trường cần thiết cho dịch vụ WordPress.
- *WORDPRESS\_DB\_HOST*: Sử dụng biến môi trường *{MYSQL\_HOST}* từ file *.env*.
- *WORDPRESS\_DB\_USER*: Sử dụng biến môi trường *{MYSQL\_USER}* từ file *.env*.
- *WORDPRESS\_DB\_PASSWORD*: Sử dụng biến môi trường *{MYSQL\_ROOT\_PASSWORD}* từ file *.env*.
- *WORDPRESS\_DB\_NAME*: Sử dụng biến môi trường *{MYSQL\_DATABASE}* từ file *.env*.
- *volumes*: Định nghĩa các volume để gắn kết thư mục trên máy host vào container.
- *./wordpress:/var/www/html*: Thư mục *./wordpress* trên máy host được gắn vào thư mục */var/www/html* trong container. Thư mục này chứa mã nguồn của ứng dụng WordPress.
- *networks*: Định nghĩa các mạng mà dịch vụ sẽ tham gia.
- *app-network*: Dịch vụ này sẽ tham gia vào mạng tên là *app-network*. Bạn cần định nghĩa mạng này trong file *docker-compose.yml* hoặc Docker sẽ tự động tạo ra mạng này nếu nó chưa tồn tại.

## 5.4. Khởi chạy dự án

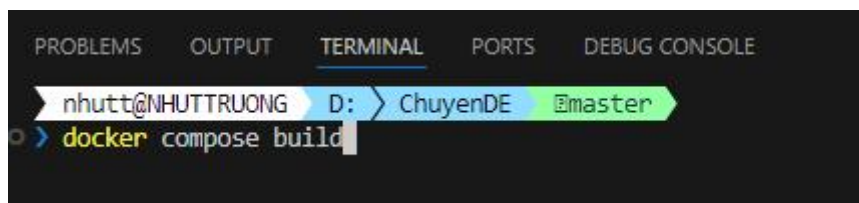
- Khởi động Docker Desktop.
- Mở bảng CMD bằng tổ hợp phím “Ctrl + `” hoặc vào View -> Terminal





*Ảnh 10. Khởi chạy Terminal*

- Lệnh “docker-compose build” được sử dụng để xây dựng (build) các image Docker dựa trên các định nghĩa trong file docker-compose.yml



*Ảnh 11. Build image docker*



- Sau đó nhập “**docker-compose up -d**” cho phép khởi động các containers.

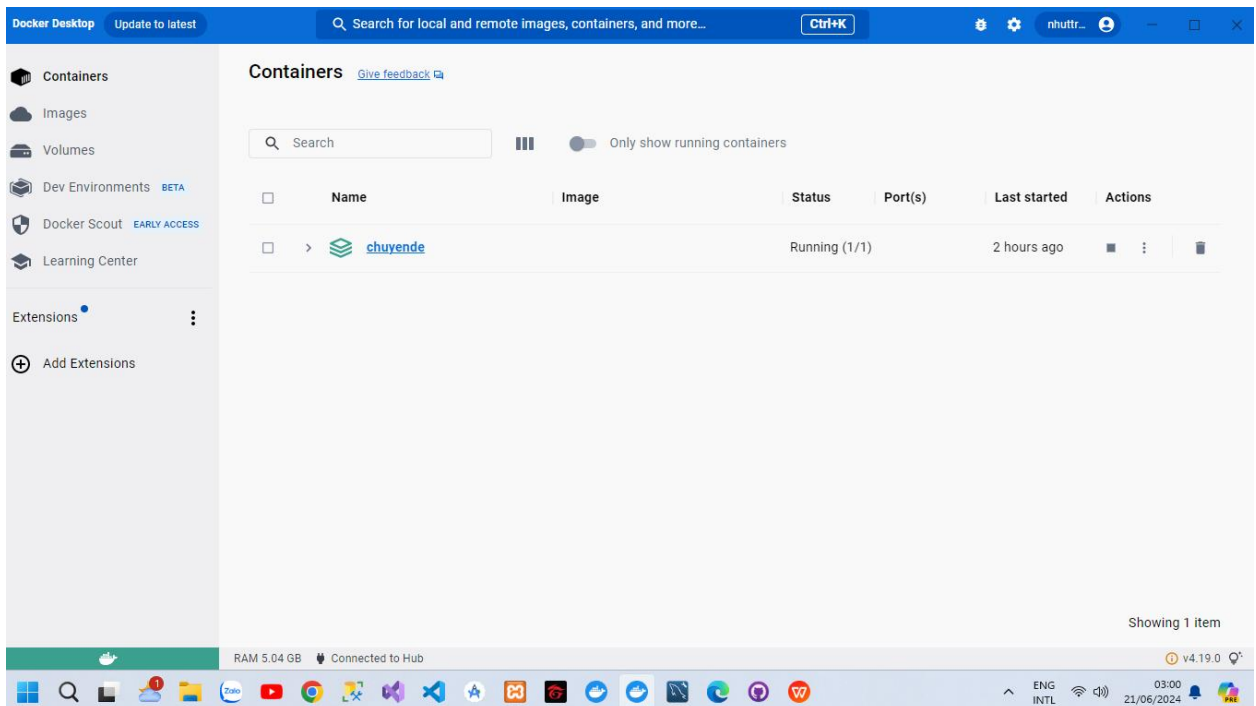


*Ảnh 12. Thực thi docker-compose up*

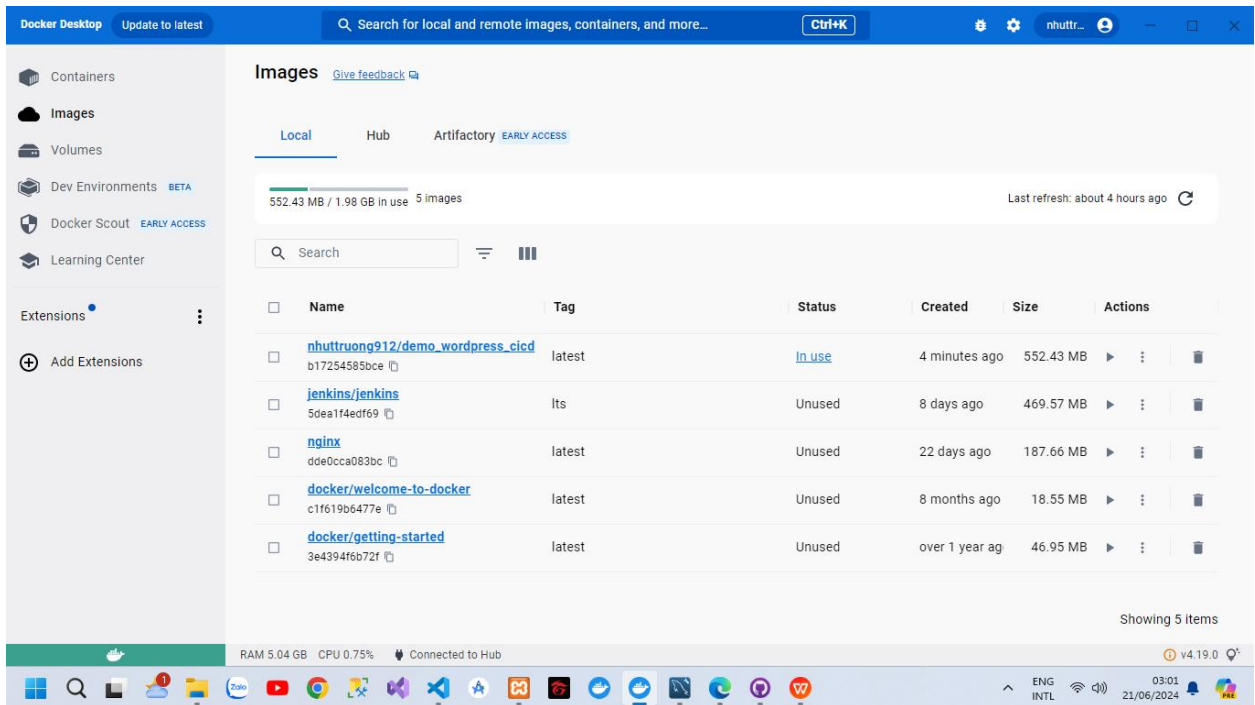


*Ảnh 13. Các Containers và Images đã được khởi chạy*

- Vào Docker Desktop để xem các Containers và Images.



*Ảnh 14. Docker Containers*

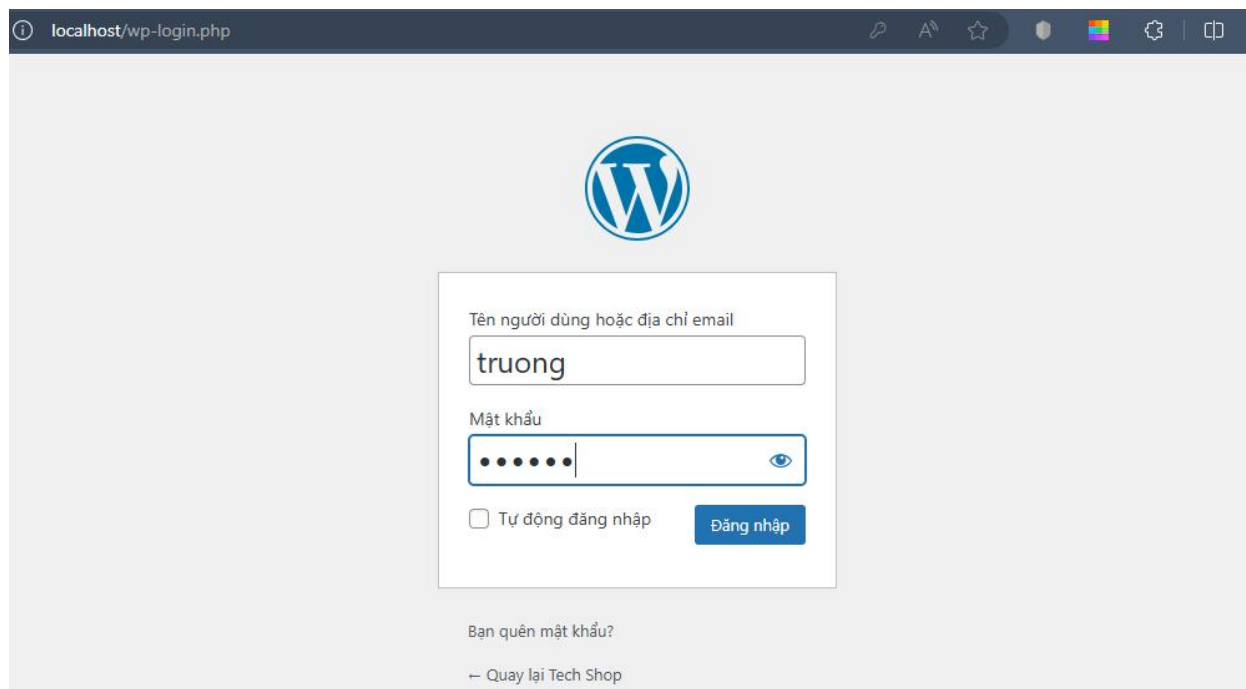


Ảnh 15. Các Docker Images

- Địa chỉ truy cập trang web: <http://localhost:80>
- Địa chỉ truy cập trang web cho quản trị viên: <http://localhost/wp-admin/>

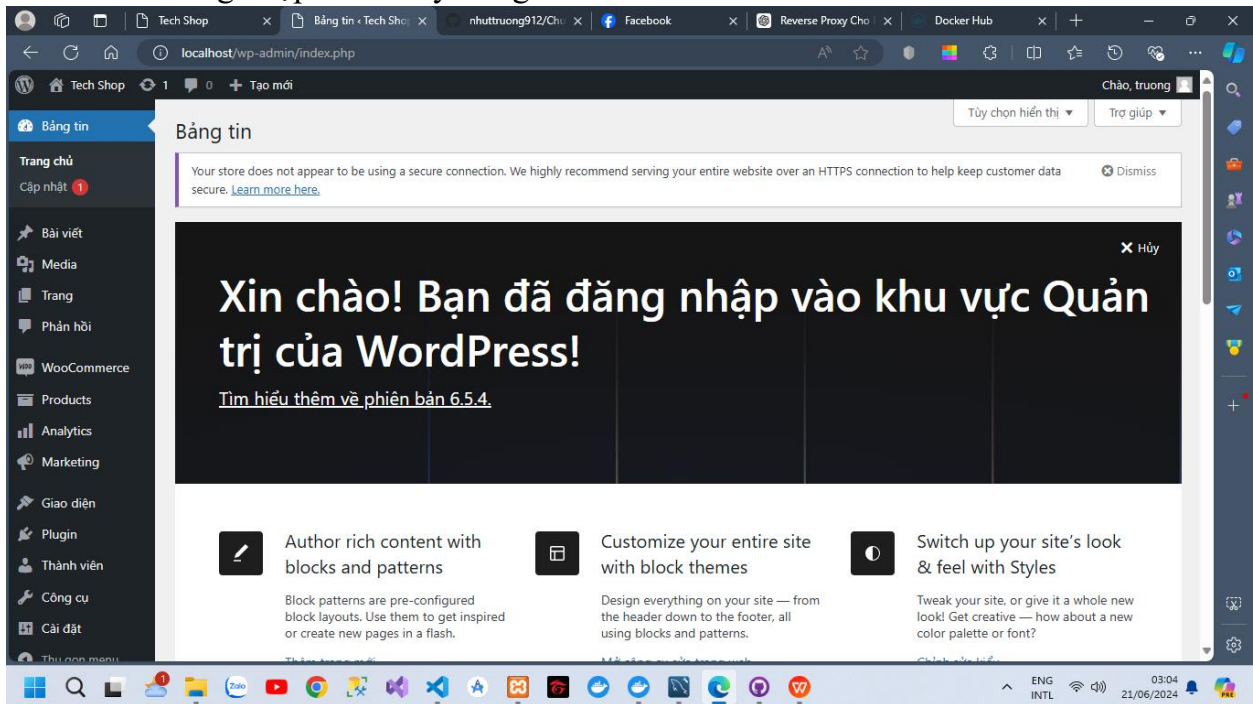
## 6. Triển khai CMS Wordpress

- Trên trình duyệt nhập “<http://localhost/wp-admin/>” để truy cập trang web quản trị viên
- Lúc này, sẽ hiển thị một trang đăng nhập wordpress.



*Ảnh 16. Trang đăng nhập quản trị wordpress.*

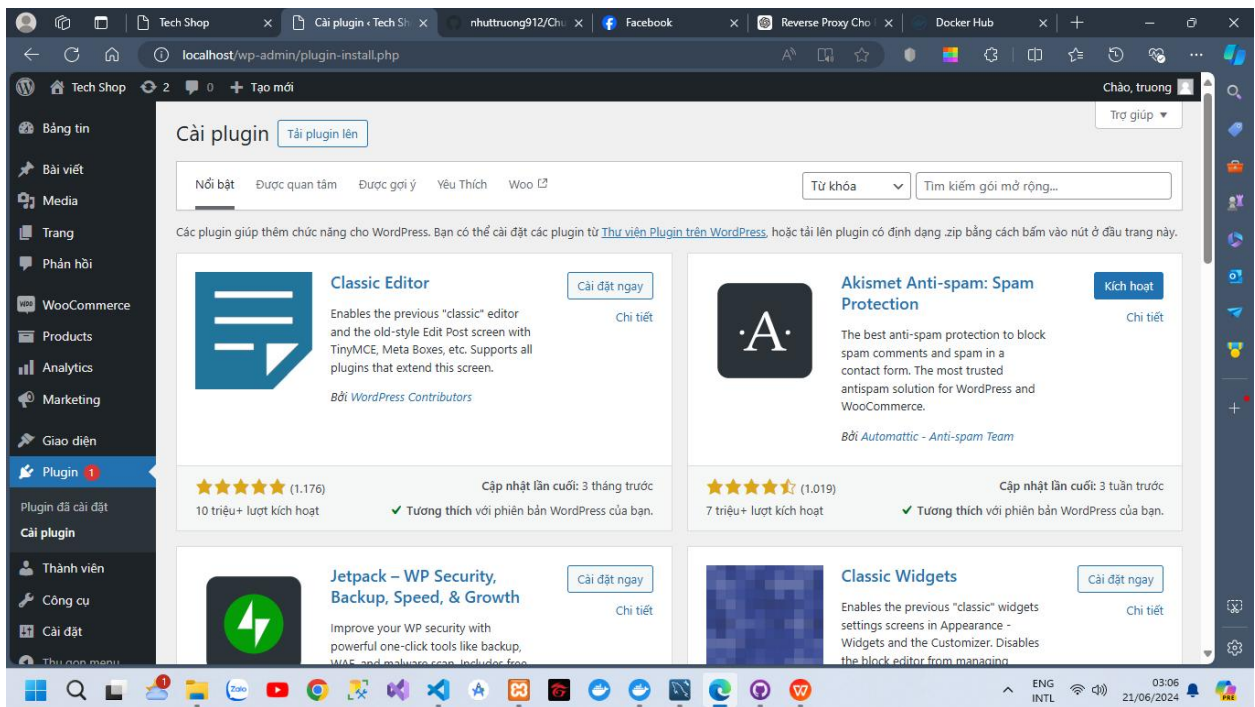
- Sau khi đăng nhập ta sẽ thấy trang chủ:



Ảnh 17. Trang quản trị admin

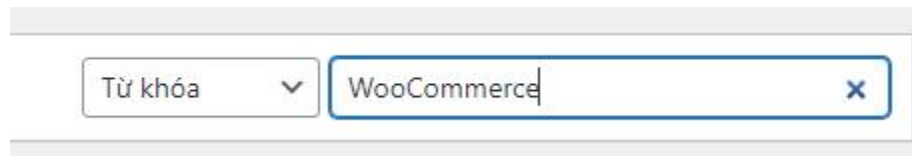
## 7. Triển khai cài đặt E-Commerce (WooCommerce)

- Wordpress cho phép cài đặt nhiều tiện ích phục vụ cho trang web
- Tại trang quản trị, trong thanh công cụ bên trái chọn “Plugin” Sau đó chọn “Cài plugin”



Ảnh 18. Các gói tiện ích của Wordpress


- Nhập từ khóa “WooCommerce” để tìm kiếm



Ảnh 19. Tìm WooCommerce

- Ta tiến hành cài đặt ” WooCommerce”, Sau khi cài đặt xong nhấn “kích hoạt” để áp dụng vào trang Web.

---




## WooCommerce

Everything you need to launch an online store in days and keep it growing for years. From your first sale to millions in revenue, Woo is with you.

*Bởi Automattic*

[Kích hoạt](#)  
[Chi tiết](#)



7 triệu+ lượt kích hoạt

Cập nhật lần cuối: 5 ngày trước

✓ Tương thích với phiên bản WordPress của bạn.

*Ảnh 20. Kích hoạt WooCommerce*

### III. Triển khai CI/CD

- Trước tiên vào Setting trong Github -> Secret and variables -> Actions -> New repository secret để thiết lập các tham số bí mật

#### Actions secrets / New secret

Name \*

YOUR\_SECRET\_NAME




















Secret \*

Add secret

Ảnh 21: Thiết lập bí mật

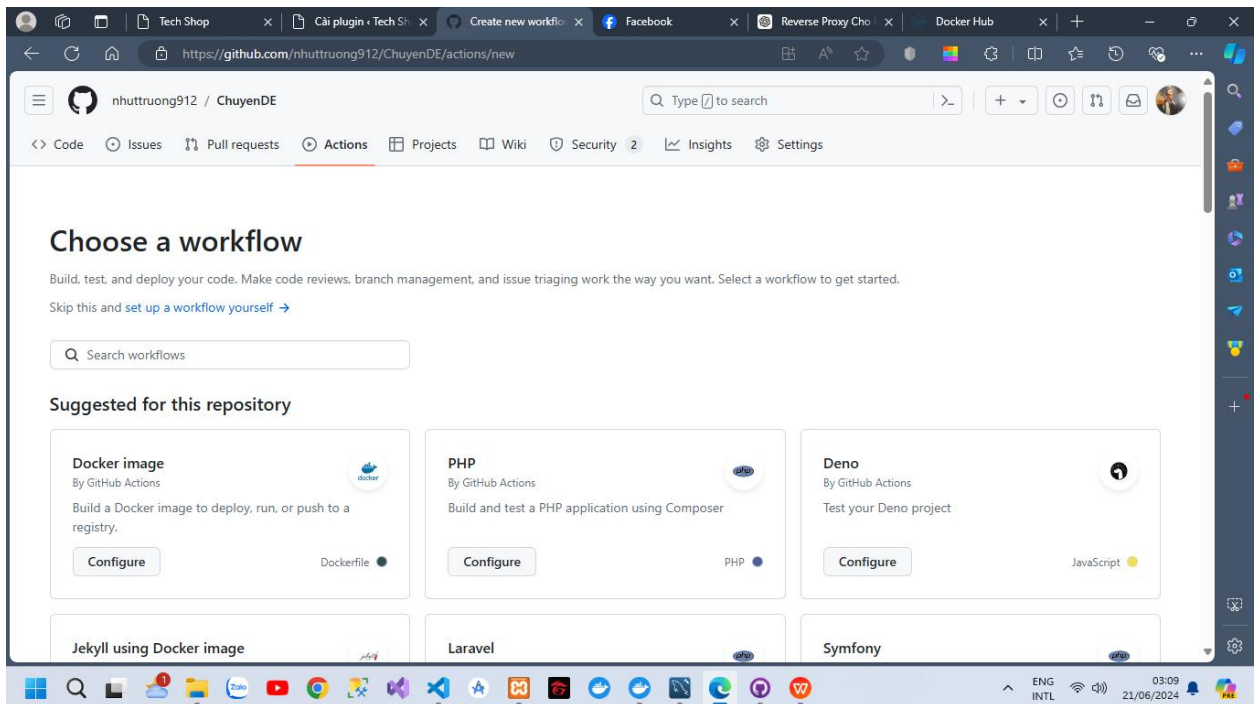
#### Repository secrets

New repository secret

Name 	Last updated
 DOCKER_PASSWORD	6 hours ago  
 DOCKER_USERNAME	6 hours ago  
 MYSQL_DATABASE	6 hours ago  
 MYSQL_HOST	6 hours ago  
 MYSQL_ROOT_PASSWORD	6 hours ago  
 MYSQL_USER	6 hours ago  

Ảnh 22: Các bí mật được thiết lập

- Sử dụng Github Actions của trang Github.



*Ảnh 23. Github actions*

- Click vào “set up a workflow yourself”



- Khi Configure, Github sẽ tạo một file có đuôi **.yaml** trong thư mục **.github/workflows**.

Dưới đây là **giải thích chi tiết** cho từng phần của đoạn **cấu hình GitHub Actions**:

```
name: CI/CD for WordPress
```

```
on:
```

```
  push:
```

```
    branches: [master]
```

- `name`: Đặt tên cho quy trình là “CI/CD for WordPress”.
- `on`: Định nghĩa sự kiện kích hoạt quy trình, ở đây là sự kiện `push` vào nhánh `master`.

```
jobs:
```

```
  build:
```

```
    runs-on: ubuntu-latest
```

- `jobs`: Định nghĩa công việc cần thực hiện.
- `build`: Tên công việc, có thể đặt tùy ý.
- `runs-on`: Chọn hệ điều hành cho runner, ở đây là phiên bản mới nhất của Ubuntu.

```
steps:
```

```
- name: Checkout code
```

```
  uses: actions/checkout@v2
```

- `steps`: Các bước thực hiện trong công việc.
- `Checkout code`: Bước này sử dụng action `actions/checkout@v2` để sao chép mã nguồn từ GitHub repository.

```
- name: Set up Docker Buildx
  uses: docker/setup-buildx-action@v2
```

- Set up Docker Buildx: Cài đặt Docker Buildx để hỗ trợ xây dựng hình ảnh Docker đa nền tảng.

```
- name: Cache Docker layers
  uses: actions/cache@v2
  with:
    path: /tmp/.buildx-cache
    key: ${ runner.os }-buildx-${ github.sha }
    restore-keys: |
      ${ runner.os }-buildx-
```

- Cache Docker layers: Lưu trữ các lớp Docker để tăng tốc độ xây dựng trong tương lai. Sử dụng `actions/cache@v2` để quản lý cache.

```
- name: Log in to Docker Hub
  uses: docker/login-action@v2
  with:
    username: ${ secrets.DOCKER_USERNAME }
    password: ${ secrets.DOCKER_PASSWORD }
```

- Log in to Docker Hub: Đăng nhập vào Docker Hub sử dụng tên người dùng và mật khẩu được lưu trữ trong secrets của GitHub.

```
- name: Build and push Docker image
  uses: docker/build-push-action@v4
  with:
    context: ./wordpress
    push: true
    tags: nhuttruong912/demo_wordpress_cicd:latest
```

- Build and push Docker image: Xây dựng và đẩy image Docker lên Docker Hub, `. context` chỉ đường dẫn tới thư mục chứa Dockerfile,

push: true để đẩy image lên sau khi xây dựng, và tags để đặt tag cho image.

```
- name: Set up environment variables
run: |
    echo "MYSQL_HOST=${{ secrets.MYSQL_HOST }}" >> .env
    echo "MYSQL_USER=${{ secrets.MYSQL_USER }}" >> .env
    echo "MYSQL_PASSWORD=${{ secrets.MYSQL_PASSWORD }}" >> .env
    echo "MYSQL_DATABASE=${{ secrets.MYSQL_DATABASE }}" >> .env
    echo "MYSQL_ROOT_PASSWORD=${{ secrets.MYSQL_ROOT_PASSWORD }}" >> .env
```

- Set up environment variables: Tạo file .env chứa các biến môi trường cho MySQL, sử dụng giá trị từ secrets.

```
- name: Set up Docker Compose
run: docker-compose -f docker-compose.yml up -d
```

- Set up Docker Compose: Khởi chạy các dịch vụ được định nghĩa trong docker-compose.yml một cách độc lập.

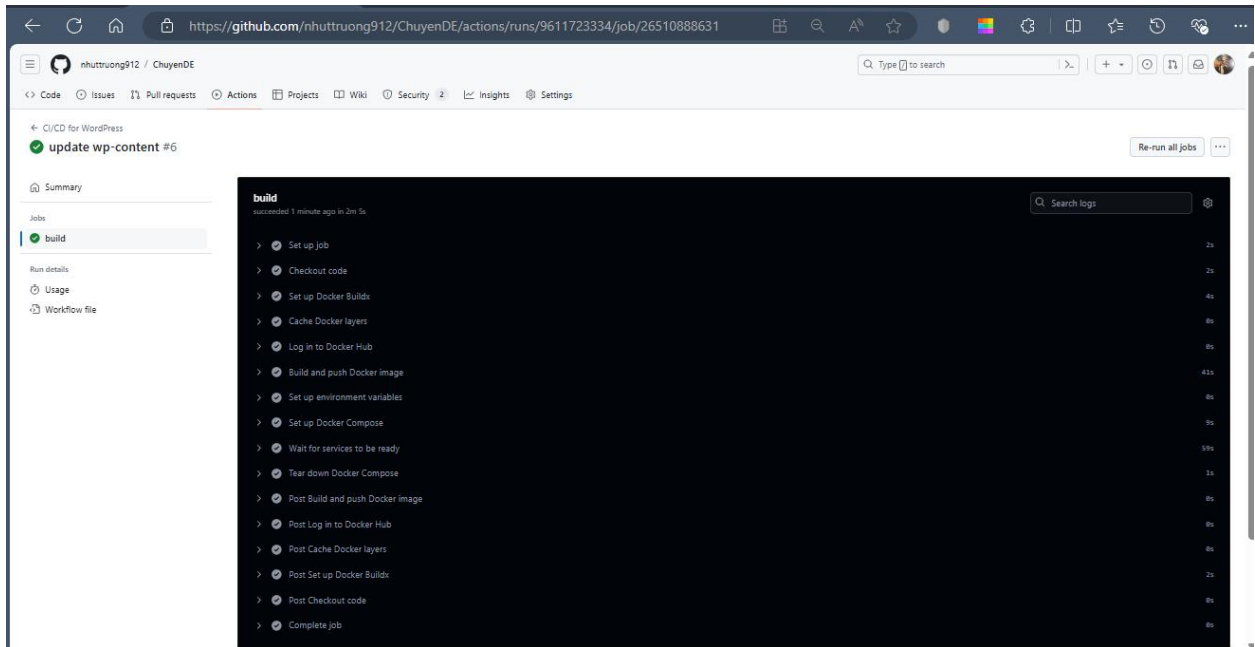
```
- name: Wait for services to be ready
run: sleep 60
```

- Wait for services to be ready: Đợi 60 giây để các dịch vụ khởi động và sẵn sàng.

```
- name: Tear down Docker Compose
run: docker-compose -f docker-compose.yml down
```

- Tear down Docker Compose: Tắt và dọn dẹp các dịch vụ sau khi quá trình CI/CD hoàn tất.

- Nên từ nhánh main, ta thực hiện push, Github Actions sẽ được khởi chạy.



*Ảnh 24. CI/CD thành công*

## IV. Triển khai Reverser Proxy bằng Nginx

```
nginx > nginx.conf
1  events {
2      worker_connections 1024;
3  }
4
5  http {
6      include mime.types;
7      default_type application/octet-stream;
8
9      sendfile on;
10     keepalive_timeout 65;
11     server {
12         # Xác định cổng (port)
13         listen 80;
14         # Xác định tên miền hoặc địa chỉ IP của máy chủ web
15         server_name localhost; # yourdomain.com
16
17         # Xác định thư mục chứa mã nguồn của website
18         location / {
19             # Khi nginx nhận yêu cầu, nó sẽ chuyển yêu cầu đến dịch vụ "localhost trên cổng 8080"
20             proxy_pass http://localhost:8080;
21
22             #
23             proxy_set_header Host $host;
24
25             # Xác định địa chỉ IP của người dùng
26             proxy_set_header X-Real-IP $remote_addr;
27
28             #
29             proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
30         }
31     }
32 }
```

Ảnh 25. File *nginx.conf* cấu hình Nginx triển khai Reverse Proxy

- **events block:** Đây là nơi bạn có thể thiết lập các thông số liên quan đến cách mà Nginx xử lý các sự kiện.
- ***worker\_connections 1024;*** Chỉ định số kết nối tối đa mà mỗi worker process có thể xử lý đồng thời. Ở đây, giá trị là 1024
- **http block:** Đây là nơi bạn cấu hình các thông số liên quan đến việc xử lý các yêu cầu HTTP.
- ***include mime.types;*** Nạp file *mime.types*, file này chứa các định nghĩa về kiểu MIME. Các kiểu MIME giúp Nginx xác định loại dữ liệu đang được truyền, ví dụ như *text/html* cho các file HTML, *image/jpeg* cho các file hình ảnh JPEG, v.v.
- ***default\_type application/octet-stream;*** Đặt kiểu MIME mặc định cho các phản hồi HTTP. *application/octet-stream* là kiểu MIME mặc định cho các file nhị phân, và được sử dụng khi không xác định được kiểu MIME của file.

- *sendfile on*:: Bật chức năng sendfile. Đây là một kỹ thuật tối ưu hóa giúp tăng hiệu suất khi phục vụ các file tĩnh. Nó cho phép Nginx gửi file trực tiếp từ ổ đĩa tới socket mạng mà không cần phải sao chép dữ liệu giữa không gian người dùng và kernel.
- *keepalive\_timeout 65*:: Thiết lập thời gian tối đa (tính bằng giây) mà kết nối keep-alive sẽ được duy trì mở. Ở đây, giá trị là 65 giây. Keep-alive cho phép giữ kết nối HTTP mở giữa client và server để tiết kiệm thời gian và tài nguyên khi xử lý nhiều yêu cầu từ cùng một client.
- *server*: Khởi tạo định nghĩa một máy chủ ảo trong Nginx.
- *listen 80*:: Lắng nghe các yêu cầu trên cổng 80, cổng mặc định cho HTTP.
- *server\_name localhost*:: Đặt tên máy chủ, có thể là localhost cho môi trường phát triển hoặc tên miền thực tế của bạn.

Trong khối *location /*:

- *proxy\_pass http://localhost:8080*:: Chuyển hướng yêu cầu đến dịch vụ localhost chạy trên cổng 8080. Đây là địa chỉ của container Docker chạy WordPress.
- *proxy\_set\_header Host \$host*:: Thiết lập header Host trong yêu cầu đến giá trị của biến *\$host*, tức là tên máy chủ của yêu cầu đến.
- *proxy\_set\_header X-Real-IP \$remote\_addr*:: Thiết lập header X-Real-IP để chứa địa chỉ IP thực của client.
- *proxy\_set\_header X-Forwarded-For \$proxy\_add\_x\_forwarded\_for*:: Thêm địa chỉ IP của client vào header *X-Forwarded-For*, thông thường được sử dụng để xác định địa chỉ IP của client qua các proxy.

Cấu hình này cho phép Nginx hoạt động như một reverse proxy, chuyển hướng yêu cầu từ client đến máy chủ WordPress và sau đó trả lại phản hồi từ WordPress đến client. Điều này giúp ẩn thông tin về cấu trúc nội bộ của mạng và cung cấp một lớp bảo mật bổ sung.

Sau khi cấu hình Nginx, có 2 cách để triển khai:

Cách 1: Cấu hình nginx server trong file docker-compose.yml để chạy trên docker container:

```

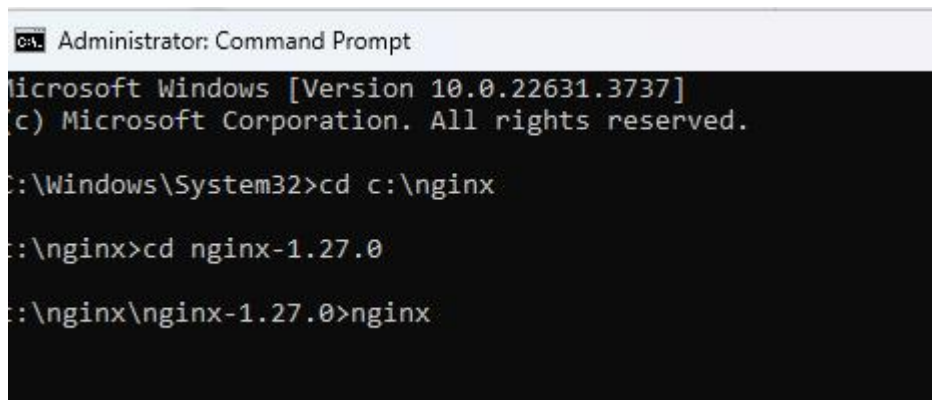
nginx:
  image: nginx:latest
  restart: always
  ports:
    - "80:80"
  volumes:
    - ./nginx:/etc/nginx/conf.d
    - ./wordpress:/var/www/html
  networks:
    - app-network

```

*Ảnh 26: Cấu hình nginx server trong file docker-compose.yml để chạy trên docker container*

## Cách 2: Chạy nginx bằng Terminal

- Tải nginx tại: <https://nginx.org/en/download.html>, lưu vào 1 folder được chọn từ trước
- Vào thư mục conf, vào file nginx.conf cấu hình tương tự *Ảnh 25*
- Khởi chạy Terminal với quyền Admin, vào trong thư mục chứa folder nginx, gõ lệnh “nginx”



```

Administrator: Command Prompt
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd c:\nginx

C:\nginx>cd nginx-1.27.0

C:\nginx\nnginx-1.27.0>nginx

```

*Ảnh 27: Triển khai nginx với Terminal*