
PROJECT REPORT

Containerization Using Docker and Deployment on AWS EC2

**Submitted by:
Namratha T G**

1. Introduction

In today's software industry, applications are expected to run reliably in different environments such as development, testing, and production. One common problem developers face is that an application works on one system but fails on another due to environment differences. Docker helps solve this problem by packaging the application along with all its required dependencies into a container.

Cloud computing platforms like Amazon Web Services (AWS) provide virtual machines that allow applications to be deployed and accessed from anywhere over the internet. AWS EC2 is a widely used service that offers scalable and cost-effective virtual servers.

In this project, a simple static website is created and containerized using Docker with the Nginx web server. The Docker container is first tested locally and then deployed on an AWS EC2 instance running Ubuntu Linux. Finally, the application is made publicly accessible using the EC2 instance's public IP address. This project demonstrates the basic concepts of Docker containerization and cloud deployment, which are essential skills in DevOps.

2. Objective

The objectives of this project are:

- To containerize a website using Docker
 - To deploy the Docker container on AWS EC2
 - To access the deployed application using a public IP
-

3. Tools and Technologies Used

- Docker
- Nginx
- AWS EC2
- Ubuntu 22.04 LTS
- Windows 11
- HTML
- SSH

4. Project Architecture

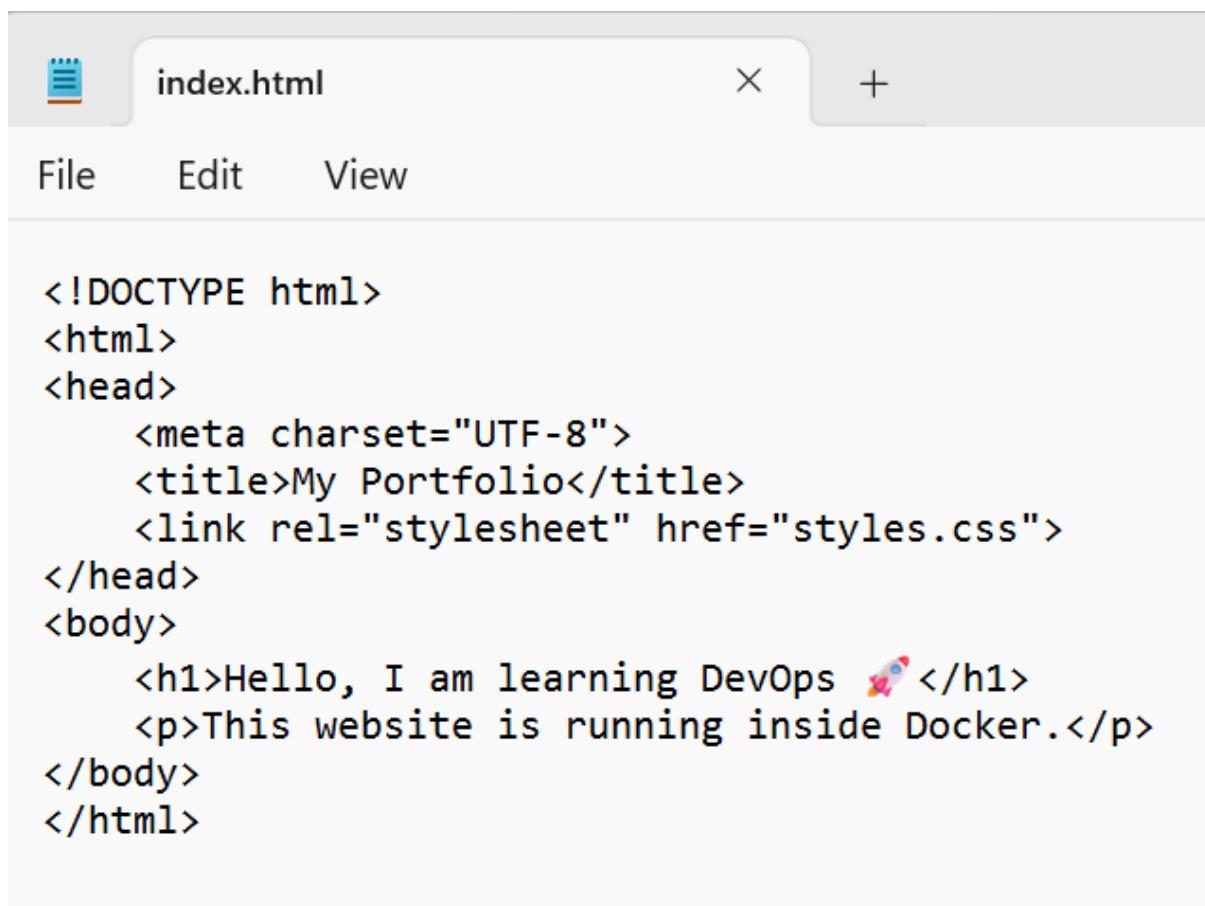
The project follows this workflow:

1. Create a simple HTML website
2. Write a Dockerfile using Nginx
3. Build a Docker image
4. Launch an AWS EC2 instance
5. Install Docker on EC2
6. Transfer project files to EC2
7. Build and run Docker container on EC2
8. Access the website using public IP

5. Implementation

Step 1: Website Creation

A simple HTML file was created to display a webpage.



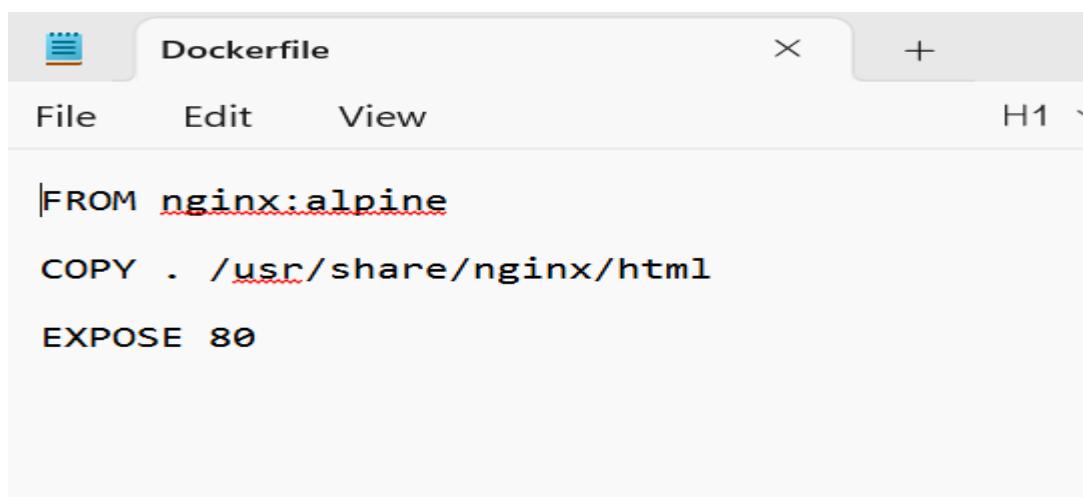
The screenshot shows a code editor window with a tab labeled "index.html". The file contains the following HTML code:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>My Portfolio</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Hello, I am learning DevOps <img alt="rocket icon" style="vertical-align: middle;"></h1>
    <p>This website is running inside Docker.</p>
</body>
</html>
```

Figure 1.1 : Index.html file

Step 2: Dockerfile Creation

A Dockerfile was created to use Nginx and copy website files into the container.



The screenshot shows a code editor window with a tab labeled "Dockerfile". The file contains the following Dockerfile code:

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
EXPOSE 80
```

Figure 1.2 : Dockerfile

Step 3: Docker Image Build on AWS

The Docker image was built on the AWS EC2 instance using the Dockerfile.

```
docker build -t portfolio-website .
```

```
ubuntu@ip-172-31-31-17: ~$ x Windows PowerShell x + - x
6b7b6c7061b7: Download complete
955a8478f9ac: Verifying Checksum
955a8478f9ac: Download complete
399d0898a94e: Verifying Checksum
399d0898a94e: Download complete
6d397a54a185: Verifying Checksum
6d397a54a185: Download complete
5e7756927bef: Verifying Checksum
5e7756927bef: Download complete
589002ba0eae: Pull complete
bca5d04786e1: Pull complete
3e2c181db1b0: Pull complete
6b7b6c7061b7: Pull complete
399d0898a94e: Pull complete
955a8478f9ac: Pull complete
6d397a54a185: Pull complete
5e7756927bef: Pull complete
Digest: sha256:5878d06ae4c83d73285438255f705bb3f9a736f41cd24876ed25bb33faf76c7d
Status: Downloaded newer image for nginx:alpine
--> b76de378d572
Step 2/3 : COPY . /usr/share/nginx/html
--> f306ae2c74d3
Step 3/3 : EXPOSE 80
--> Running in e7767908cbec
--> Removed intermediate container e7767908cbec
--> a0c1015b37de
Successfully built a0c1015b37de
Successfully tagged portfolio-website:latest
ubuntu@ip-172-31-31-17:~/portfolio-website$ client_loop: send disconnect: Connection reset
PS C:\Users\Namratha TG> ssh -i "C:\Users\Namratha TG\docker-key.pem" ubuntu@54.227.90.155
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1040-aws x86_64)

 * Documentation: https://help.ubuntu.com

 4 ENG IN 08:57 07-02-2026
```

Figure 1.3 : Image was built

Step 4: Run Docker Container on AWS

The Docker container was started and exposed on port 80.

```
docker run -d -p 80:80 portfolio-website
```

```
ubuntu@ip-172-31-31-17:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ubuntu@ip-172-31-31-17:~$ ls
portfolio-website
ubuntu@ip-172-31-31-17:~$ cd portfolio-website
ubuntu@ip-172-31-31-17:~/portfolio-website$ docker run -d -p 80:80 portfolio-website
1a83f931dfd55fb3e29078fbdb365a773ae14717f909e9b0cb62bc96175dfa9d6
ubuntu@ip-172-31-31-17:~/portfolio-website$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
MES
1a83f931dfd5  portfolio-website  "/docker-entrypoint..."  23 seconds ago  Up 23 seconds  0.0.0.0:80->80/tcp, [::]:80->80/tcp  sw
eet_cerf
```

Figure 1.4 : Docker ps

Step 5: Access Website Using Public IP

The website was accessed using the public IPv4 address of the EC2 instance.

<http://54.227.90.155/>

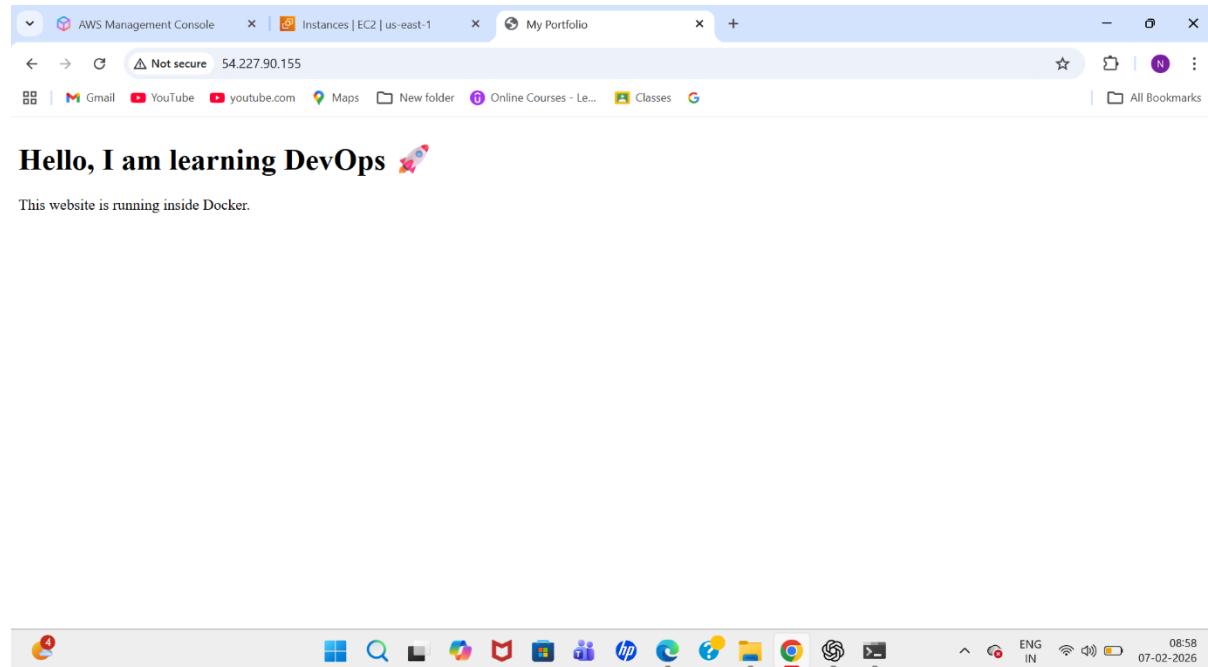


Figure 1.5: Website Running

6. AWS EC2 Configuration

- Instance Type: t2.micro
- Operating System: Ubuntu 22.04 LTS
- Security Groups:
 - SSH (Port 22) – Enabled
 - HTTP (Port 80) – Enabled

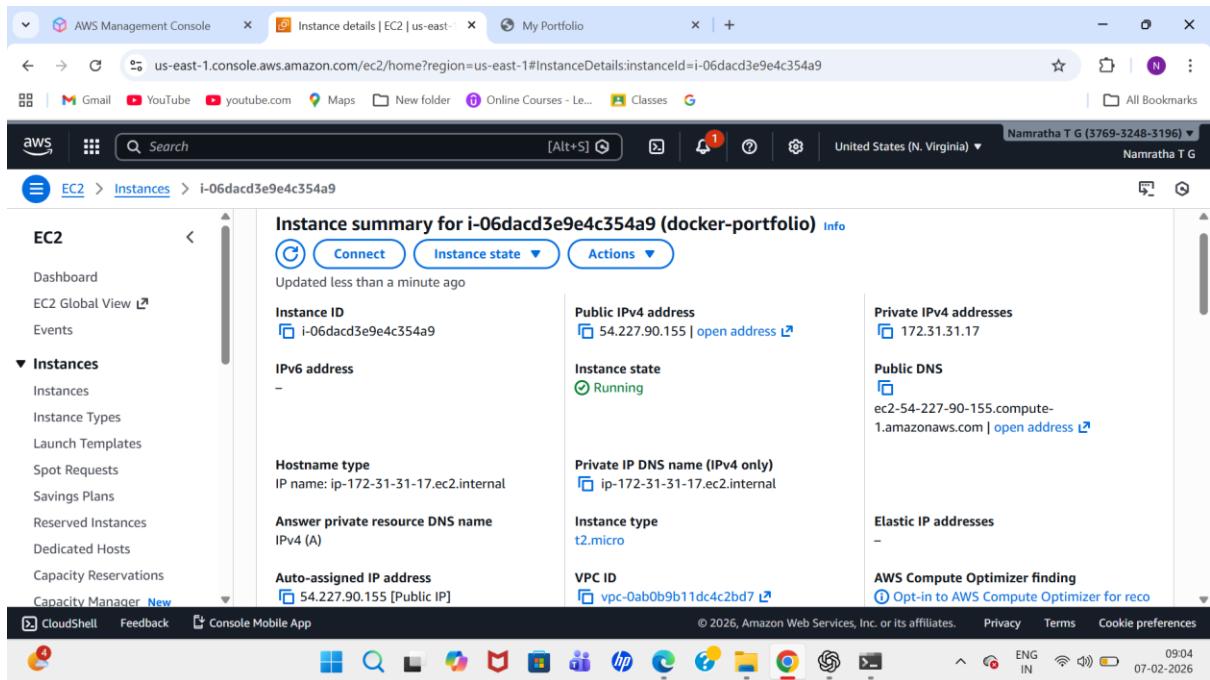


Figure 1.6 : EC2 Instance

7. Result

The website was successfully deployed inside a Docker container on AWS EC2 and accessed using a public IP address.

8. Conclusion

This project demonstrates Docker containerization and cloud deployment using AWS EC2. It provides hands-on experience with real-world DevOps tools and workflows.