



Placement Empowerment Program

Cloud Computing and DevOps Centre

**Build and Run a Custom Docker Image:
Create a Docker file to package your static website into
a Docker container and run it locally.**

Name: Sylashri Rajendran

Department: IT



Introduction

With the increasing adoption of containerization in modern software development, **Docker** has become a key technology for packaging applications in a consistent and portable way. In this Proof of Concept (POC), we explore how to **containerize a static website** using **Docker** and **Nginx**.

By creating a **Docker image** for our website, we ensure that it can run consistently across different environments without worrying about dependencies, configurations, or setup issues. This POC is especially useful for developers and DevOps engineers who want to deploy static sites in a **lightweight and efficient manner**.

Overview

This POC demonstrates how to:

1. Create a **Dockerfile** to define a containerized static website.
2. Use **Nginx** as a web server to serve the website inside a container.

3. Build a **Docker image** for the static site.

4. Run a **Docker container** to host and test the website locally.

By the end of this POC, we will have a working **Dockerized static website** that can be easily deployed and shared.

Objectives

The key goals of this POC are:

1. **Understand the basics of Docker and Dockerfiles.**
2. **Learn how to use Nginx to serve static files inside a container.**
3. **Practice building and running Docker containers for web applications.**
4. **Ensure the website runs consistently across different systems.**
5. **Prepare for real-world deployment scenarios using containerized environments.**

Importance

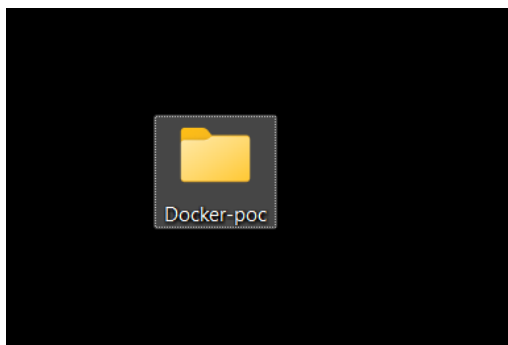
1. **Portability:** The website runs the same way on any system with Docker installed.
2. **Consistency:** No dependency issues since everything is inside the container.
3. **Fast Deployment:** Running the website takes just a few commands.

4. **DevOps Skill Development:** Provides hands-on experience with Docker, an essential tool in DevOps.
5. **Scalability:** Can be extended for cloud deployments using AWS, Azure, or Kubernetes.

Step-by-Step Overview

Step 1:

Create a folder (Docker-poc)



Step 2:

Open Command Prompt and navigate to the folder which is created.

```
C:\Users\sylas>cd C:\Users\sylas\OneDrive\Desktop\Docker-poc
```

Step 3:

Create a new Directory

mkdir docker-static-website

cd docker-static-website

```
C:\Users\sylas\OneDrive\Desktop\Docker-poc>mkdir docker-static-website
```

```
C:\Users\sylas\OneDrive\Desktop\Docker-poc>cd docker-static-website
```

Step 4:

Create a Folder for Your Static Website **mkdir**

html

```
C:\Users\sylas\OneDrive\Desktop\Docker-poc\docker-static-website>mkdir html
```

Step 5:

Create a Simple index.html File

Inside html, create a new file named index.html:

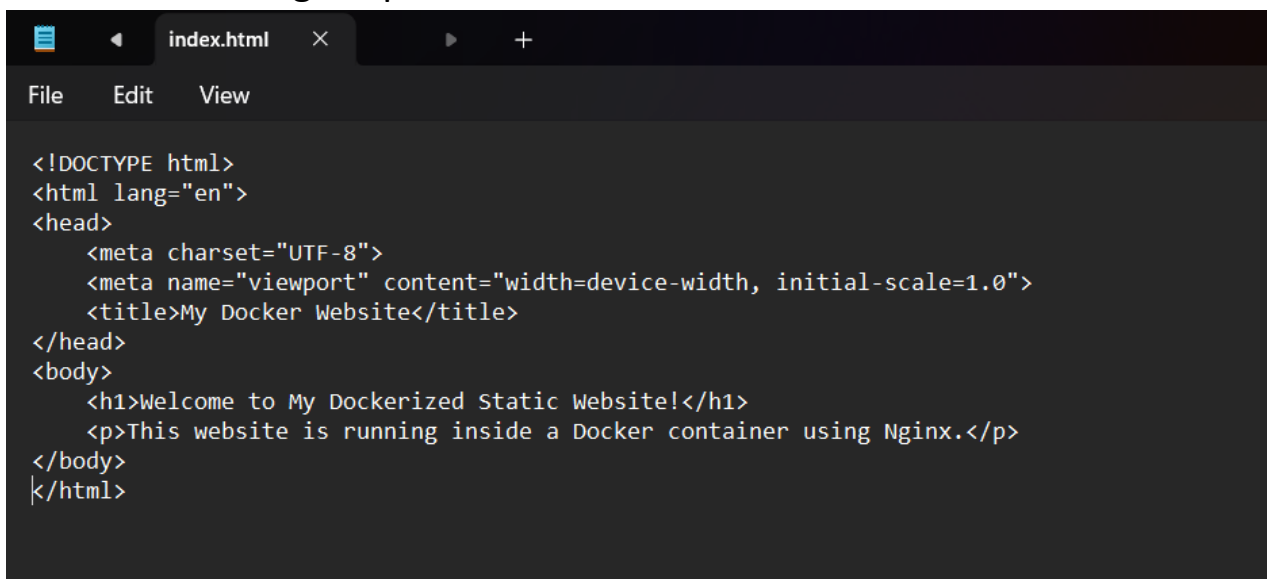
cd html notepad

index.html

```
C:\Users\sylas\OneDrive\Desktop\Docker-poc\docker-static-website>cd html
C:\Users\sylas\OneDrive\Desktop\Docker-poc\docker-static-website\html>notepad index.html
```

Step 6:

Add the following simple HTML code:

A screenshot of a Notepad application window. The title bar shows 'index.html' and standard window controls. The menu bar includes 'File', 'Edit', and 'View'. The text area contains the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Docker Website</title>
</head>
<body>
  <h1>Welcome to My Dockerized Static Website!</h1>
  <p>This website is running inside a Docker container using Nginx.</p>
</body>
</html>
```

Step 7:

Go Back to the Main Project Folder **cd**

```
..
```

Create a New File Named Dockerfile **notepad**

Dockerfile

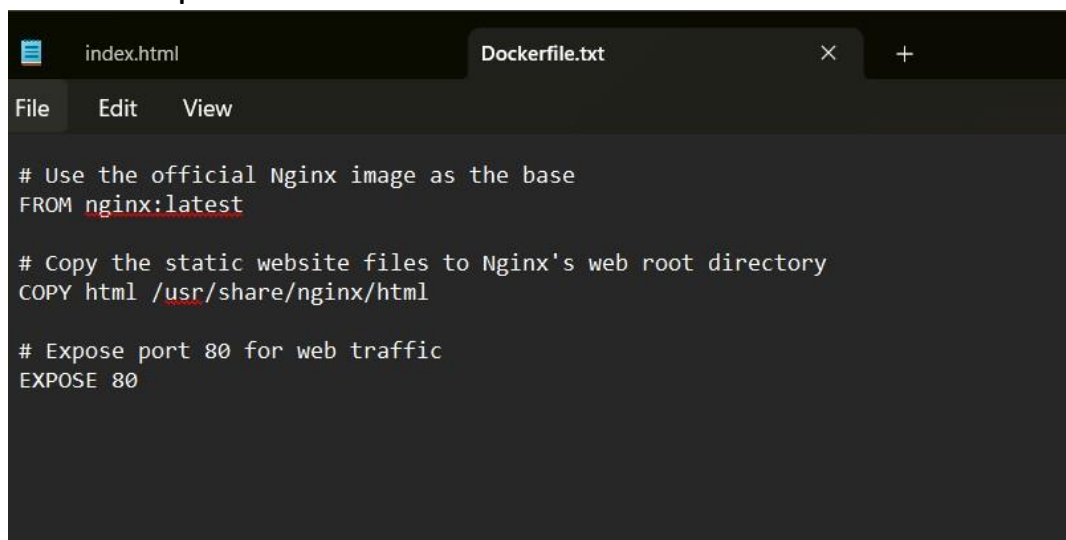
```
C:\Users\subam\OneDrive\Desktop\Docker-poc\docker-static-website\html>cd ..
C:\Users\subam\OneDrive\Desktop\Docker-poc\docker-static-website>notepad Dockerfile
```

Step 8:

Add the Following Content to the Dockerfile

Click **File** → **Save**

Close Notepad



```
# Use the official Nginx image as the base
FROM nginx:latest

# Copy the static website files to Nginx's web root directory
COPY html /usr/share/nginx/html

# Expose port 80 for web traffic
EXPOSE 80
```

Step 9:

Build the Docker Image **docker**

build -t my-static-website .

```
[+] Building 3.3s (8/8) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 247B                             0.1s
=> [internal] load metadata for docker.io/library/nginx:latest  0.0s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 422B                                  0.0s
=> [1/2] FROM docker.io/library/nginx:latest@sha256:9d6b58feebd2dbd3c56ab5853333d627cc6e281011cfd6050fa4bcf2072c 2.4s
=> => resolve docker.io/library/nginx:latest@sha256:9d6b58feebd2dbd3c56ab5853333d627cc6e281011cfd6050fa4bcf2072c 2.2s
=> [auth] library/nginx:pull token for registry-1.docker.io    0.0s
=> [2/2] COPY html /usr/share/nginx/html                       0.0s
=> exporting to image                                           0.3s
=> => exporting layers                                           0.1s
=> => exporting manifest sha256:2409d11951878345f7421c345c5d1013024539b52ca24d5b64286db05894cb88 0.0s
=> => exporting config sha256:fd7a63616e79eb39357c56a1a87a09c8a95a1893e5be955c3a4fb674edf6e38d 0.0s
=> => exporting attestation manifest sha256:5f9627049a785abde5eec5e580c418b1c326aeded3bfde63389b32360c530cee 0.0s
=> => exporting manifest list sha256:9125b0a60ce5d3f7ff6b4dae17e969fdd42c9d021f3437821ab7dcc2c2d67927 0.0s
=> => naming to docker.io/library/my-static-website:latest     0.0s
=> => unpacking to docker.io/library/my-static-website:latest  0.1s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/vuyu3q9wm9iabplc6aaod03rr
```

Step 10:

Once the build is complete, check if the image was created successfully: **docker images**

You should see a list of Docker images, including **my-static-website**.

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|--------------------------|--------|--------------|----------------|--------|
| my-static-website | latest | 9125b0a60ce5 | 43 seconds ago | 279MB |
| nginx | latest | 9d6b58feebd2 | 3 weeks ago | 279MB |
| docker/welcome-to-docker | latest | eedaff45e3c7 | 16 months ago | 29.5MB |

Step 11:

Now, we will create and start a container from the **my-static-website** image.

Run the Container : **docker run -d -p**

8080:80 my-static website

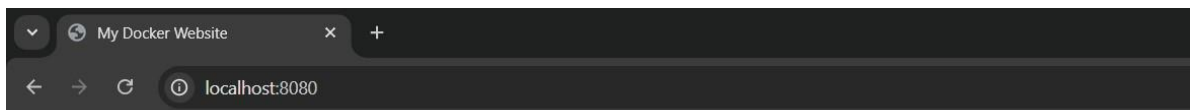
Step 12:

Test the Website

Open your browser and visit:

http://localhost:8080

If everything is correct, you should see **your static website running!**

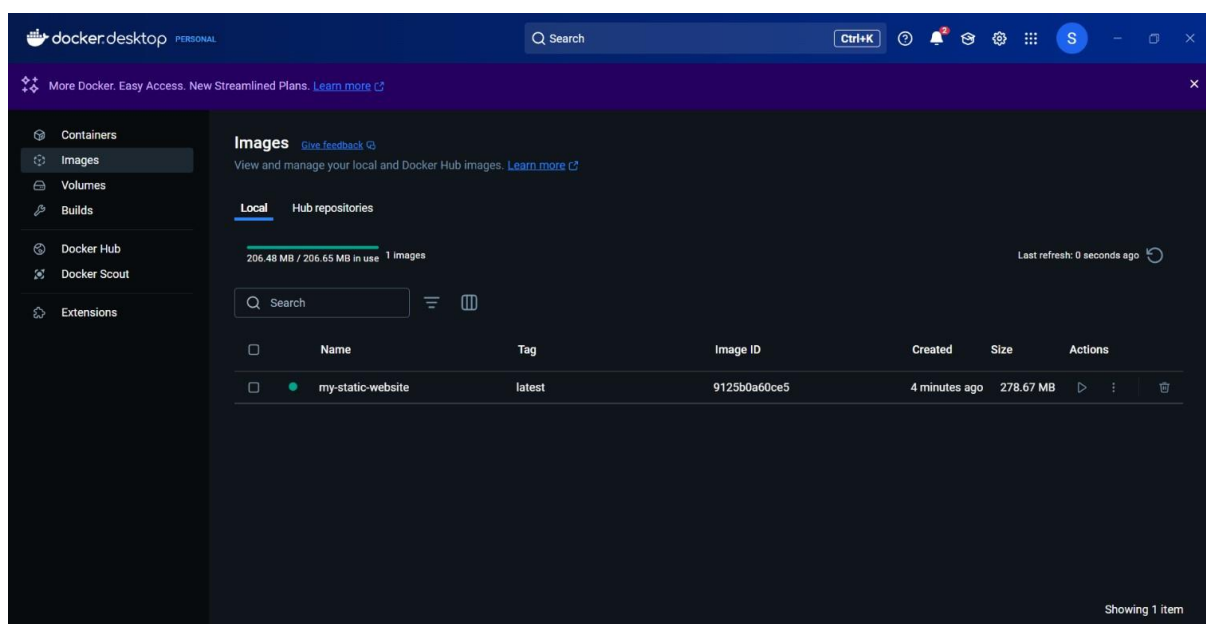


Welcome to My Dockerized Static Website!

This website is running inside a Docker container using Nginx.

Step 13:

You can also see the Docker Images in Docker Desktop.



Step 14:

Stop and Remove the Container (Optional) If you want to stop the running container:

docker ps # Get the container ID

docker stop <container_id> To

remove the container: **docker rm**

<container_id>

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS | NAMES |
|-------------------------|-------------------|-------------------------|---------------|--------------|----------------------|--------|
| 9273ef1ae909 hermann | my-static-website | "/docker-entrypoint..." | 3 minutes ago | Up 3 minutes | 0.0.0.0:8080->80/tcp | crazy_ |

Outcomes

By completing this POC, you will:

1. **Create and Configure a Dockerfile** – Learn to define a containerized static website using Dockerfile commands.
2. **Build a Docker Image** – Package the static website into a Docker image using docker build.
3. **Run a Docker Container** – Deploy the website inside a container using Nginx as the web server.
4. **Expose and Access the Website** – Map ports to access the running container via a web browser.
5. **Manage Docker Containers** – Use essential Docker commands to start, stop, and remove containers.
6. **Understand Containerization Benefits** – Explore how Docker simplifies deployment, improves portability, and streamlines DevOps workflows.