Placement Empowerment Program

Cloud Computing and DevOps Centre

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

Name: Santhoshini T

Department: CSE

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 (my-docker-html)

Step 2:

Install docker

1.open command prompt and check

If docker is installed

```
C:\Users\santhoshini>docker --version
Docker version 27.5.1, build 9f9e405
```

docker –version

Step 3:

Create a new Directory

mkdir my-docker-html &&

cd my-docker-html

```
C:\Users\santhoshini>mkdir my-docker-html && cd my- docker-html
A subdirectory or file my-docker-html already exists.
```

Step 4:

Create an extensions in **cd my-docker-html**

Step-by-Step Overview

Step 4:

Create a extensions

Type nul > index.html

```
C:\Users\santhoshini\my-docker-html>type nul > index.html
```

Step 5:

Create a Simple index.html File

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

Type nul > index.html
notepad index.html

Step 6:

Add the following simple HTML code:

index.html

File Edit View

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Docker HTML App</title>
</head>
<body>
    <h1>Hello from Docker!</h1>
    <p>This HTML page is served from inside a Docker container.</p>
</body>
</html>
```

Step 7:

Go Back to the Main Project Folder

Type nul >Dockerfile

Create a New File Named Dockerfile

notepad Dockerfile

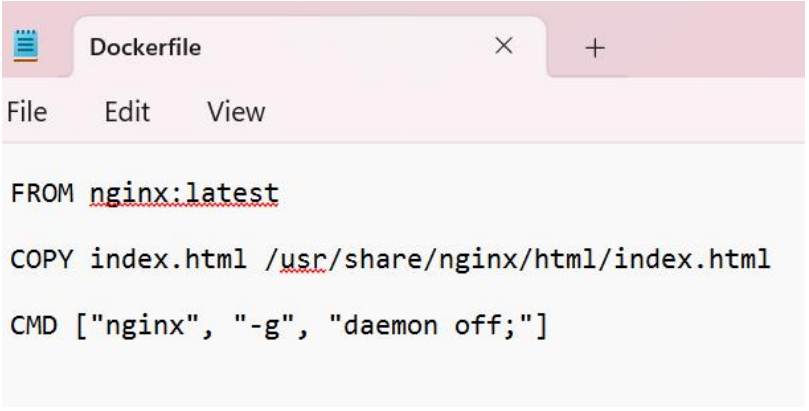
```
C:\Users\santhoshini\my-docker-html>type nul >Dockerfile

C:\Users\santhoshini\my-docker-html>notepad Dockerfile
```

Step 8:

Add the Following Content to the Dockerfile

Click **File → Save**

Close Notepad



Step 9:

Build the Docker Image

docker build -t my-html-image .

```
C:\Users\santhoshini\my-docker-html>docker build -t my-html-image .
[+] Building 0.6s (7/7) FINISHED                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 144B                                                          0.0s
 => [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                                                             0.2s
 => => transferring context: 357B                                                             0.1s
 => [1/2] FROM docker.io/library/nginx:latest                                                 0.2s
 => [2/2] COPY index.html /usr/share/nginx/html/index.html                                    0.1s
 => exporting to image                                                                        0.1s
 => => exporting layers                                                                       0.1s
 => => writing image sha256:ecf15c4cbf33e72f9451b27b1d3b65f68d3300ef5cc945994637ee4b54c174b4  0.0s
 => => naming to docker.io/library/my-html-image                                              0.0s
```

Step 11:

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

Run the Container :

docker run -d -p 9000:80 my-html-image .

```
C:\Users\santhoshini\my-docker-html>docker run -d -p 9000:80 my-html-image .
8f2c0af449faeda98be9b670d725f470ef2a3975cf6665415c09322f95a8f344
```

Step 12:

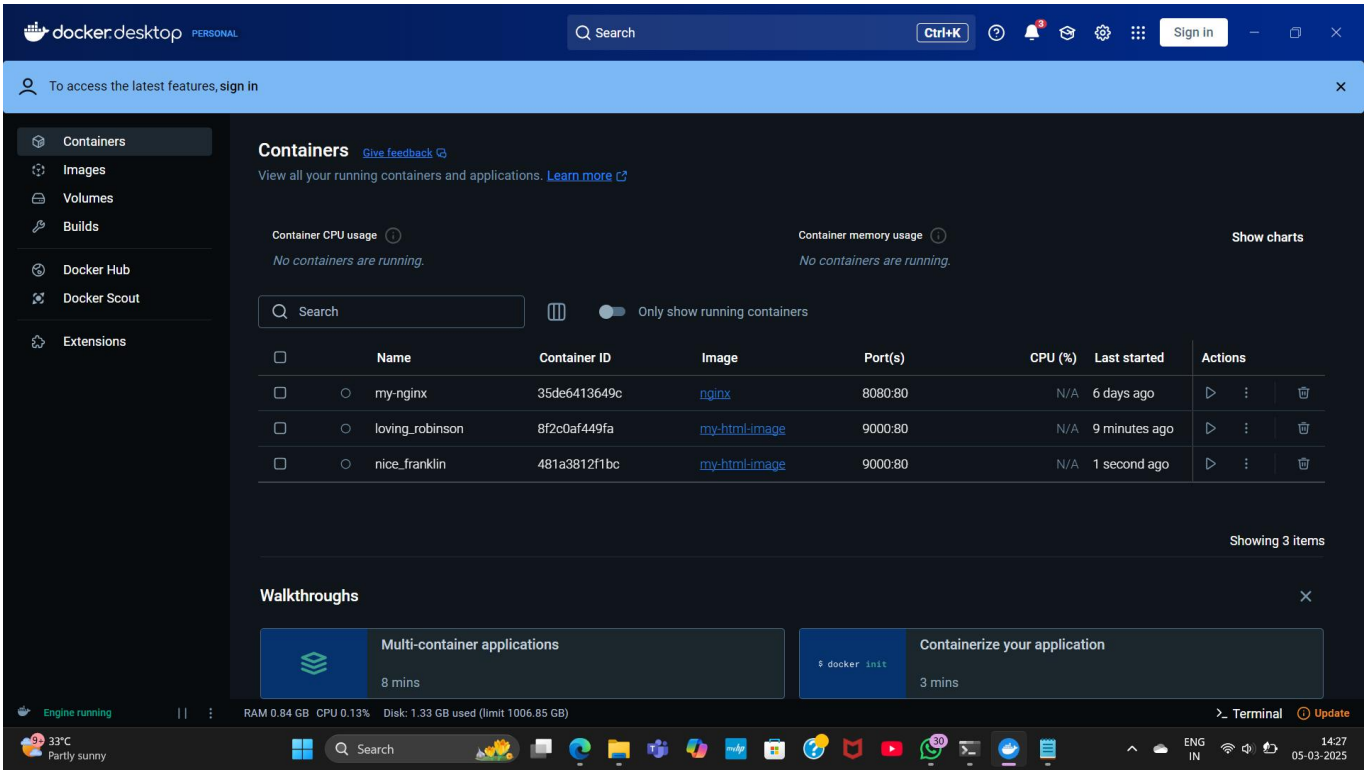
Test the Website

Open your browser and visit:

https://localhost:9000

If everything is correct, you should see **Hello from Docker!**

Step 13:



You can also see the Docker Images in Docker Desktop.

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.