

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: Swathika K

DEPARTMENT: AML

INTRODUCTION:

Containerization has revolutionized the way applications are developed, deployed, and managed. Docker provides a lightweight and efficient way to package applications with all dependencies, ensuring consistency across different environments. In this POC, we will build and run a custom Docker image that serves a simple HTML web page using NGINX.

OBJECTIVE:

This POC demonstrates how to:

- Create a Docker static website using NGINX.
- Build a custom Docker image with an HTML file.
- Run a Docker container to serve the HTML content.
- Access the webpage via `http://localhost:1802`.

By the end of this exercise, you will have a functional web server running inside a Docker container.

STEP BY STEP OVERVIEW:

Step 1: INSTALL DOCKER DESKTOP

- Download Docker Desktop for Windows from the official website.
- Check if docker is installed using the following command.

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

Step 2: CREATE PROJECT DIRECTORY

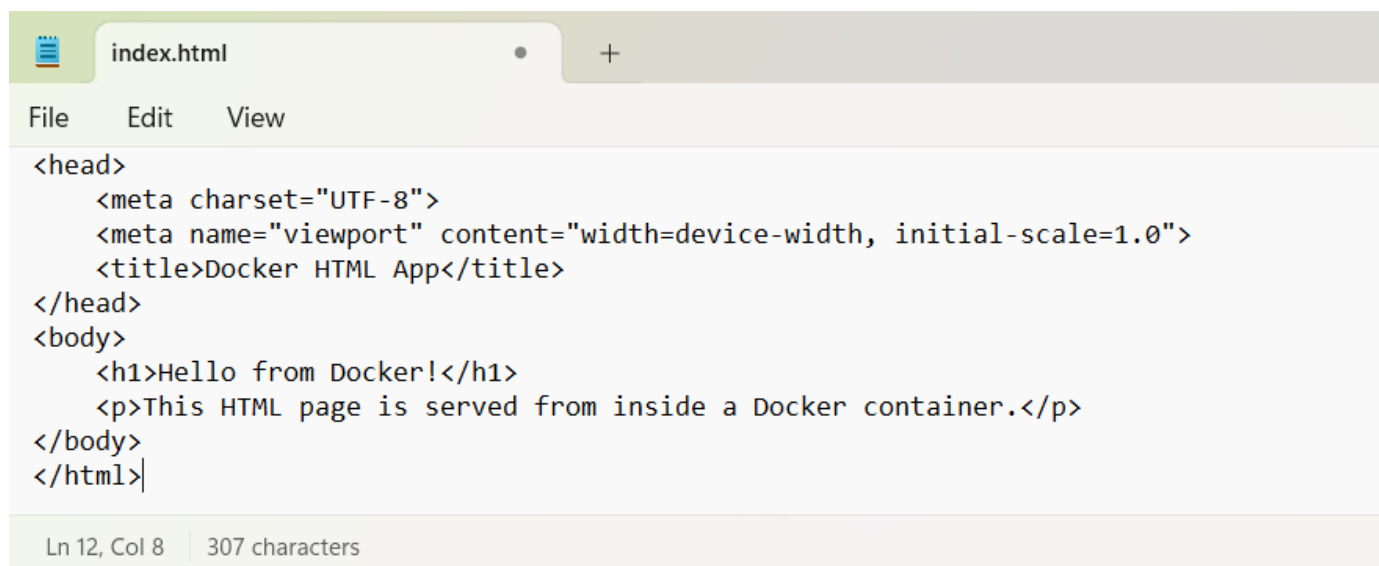
- Open Command Prompt and run the following command to create a Project Directory.
- The command 'mkdir my-docker-html' creates a new directory and 'cd my-docker-html' moves it inside the directory.

```
C:\Users\Swathika>mkdir mydocker-html && cd mydocker-html
C:\Users\Swathika\mydocker-html>|
```

Step 3: CREATE AN HTML FILE

- To create an empty html file using the command prompt use the following commands:
- 'type nul > index.html' – creates a html file named 'index.html' and 'notepad index.html' opens the file in notepad.

```
C:\Users\Swathika\mydocker-html>type nul > index.html
C:\Users\Swathika\mydocker-html>notepad index.html
```



```
index.html
File Edit View
<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>
Ln 12, Col 8 307 characters
```

Step 4: CREATE A DOCKER FILE

- Create a docker file via command prompt using the following command:
- ‘type nul> Dockerfile’. This creates an empty docker file.
- Now, use the command ‘notepad Dockerfile’ to open the file created in notepad.

```
C:\Users\Swathika\mydocker-html>type nul >Dockerfile
C:\Users\Swathika\mydocker-html>notepad Dockerfile
C:\Users\Swathika\mydocker-html>
```



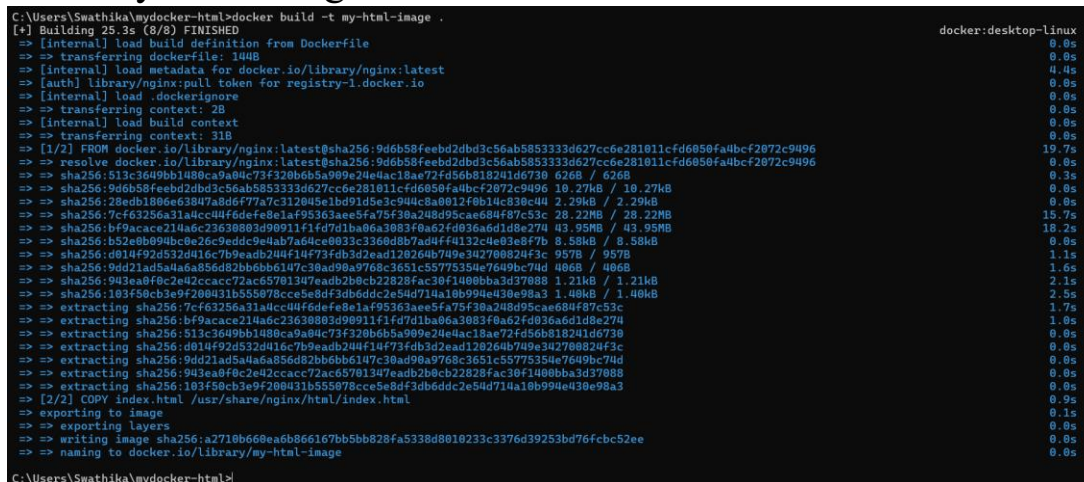
```
FROM nginx:latest

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

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

Step 5: BUILD DOCKER IMAGE

- To build a Docker image via command prompt, use the command ‘build -t my-html-image’.



```
C:\Users\Swathika\mydocker-html>docker build -t my-html-image .
[+] Building 25.3s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 144B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 31B
=> [1/2] FROM docker.io/library/nginx:latest@sha256:9d6b58feebd2dbd3c56ab585333d627cc6e281011cfd6056fa4bcf2072c9496
=> resolve docker.io/library/nginx:latest@sha256:9d6b58feebd2dbd3c56ab585333d627cc6e281011cfd6056fa4bcf2072c9496
=> sha256:513c3649bb1480ca9a04c73f320b6b5a909e24e4ac18ae72fd56b818241d6730 626B / 626B
=> sha256:9d6b58feebd2dbd3c56ab585333d627cc6e281011cfd6056fa4bcf2072c9496 10.27kB / 10.27kB
=> sha256:28ed180e63847a8d6f77a7c312045e1bd91d5e3c944c8a0012f0b14c830c44 2.29kB / 2.29kB
=> sha256:7cf63256a31a4cc44f6defe8e1af95363aee5fa75f30a248d95cae684f87c53c 28.22MB / 28.22MB
=> sha256:b79acace214a6c23638803d90911f1fd7d1ba06a3083f8a62fd036a6d1d8e274 43.95MB / 43.95MB
=> sha256:b52eb09b0b0e26e9dde9e4b7a6d4ce003c33668087a44f4132c4e93a8f7b 8.58kB / 8.58kB
=> sha256:d014f92d532d416c7b9eadb244f14f73fdb3d2ead120264b749e342708824f3c 957B / 957B
=> sha256:9dd21ad5a4a6a856d82bb6b6147c30ad90a9768c3651c55775354e7649bc74d 486B / 486B
=> sha256:943ea0f0c2e42ccacc72ac65701347eadb2b0cb22828fac30f1400bba3d37088 1.21kB / 1.21kB
=> sha256:103f50cb3e9f200431b555078cce5e8df3db6ddc2e54d714a10b9994e430e98a3 1.40kB / 1.40kB
=> extracting sha256:7cf63256a31a4cc44f6defe8e1af95363aee5fa75f30a248d95cae684f87c53c
=> extracting sha256:b79acace214a6c23638803d90911f1fd7d1ba06a3083f8a62fd036a6d1d8e274
=> extracting sha256:513c3649bb1480ca9a04c73f320b6b5a909e24e4ac18ae72fd56b818241d6730
=> extracting sha256:d014f92d532d416c7b9eadb244f14f73fdb3d2ead120264b749e342708824f3c
=> extracting sha256:9dd21ad5a4a6a856d82bb6b6147c30ad90a9768c3651c55775354e7649bc74d
=> extracting sha256:943ea0f0c2e42ccacc72ac65701347eadb2b0cb22828fac30f1400bba3d37088
=> extracting sha256:103f50cb3e9f200431b555078cce5e8df3db6ddc2e54d714a10b9994e430e98a3
=> [2/2] COPY index.html /usr/share/nginx/html/index.html
=> exporting to image
=> exporting layers
=> writing image sha256:a2710b660ea6866167bb5bb828fa5338d8010233c3376d39253bd76fcb52ee
=> naming to docker.io/library/my-html-image
C:\Users\Swathika\mydocker-html>
```

Step 6: RUN THE DOCKER CONTAINER

- Run the following command to start the container from the image and expose it on port 1802.

```
C:\Users\Swathika\mydocker-html>docker run -d -p 1802:80 my-html-image  
27afa5a5027918f75eeee7f86995146fb2140cb7a3e1751e976f3f16e7b1e395
```

Step 7: VIEW THE WEBPAGE

- Open any web browser and search for localhost:1802 and you will see your webpage.



Hello from Docker!

This HTML page is served from inside a Docker container.

EXPECTED OUTCOME

From this POC, we have learnt:

- Successful installation of Docker Desktop on Windows.
- Verification that Docker is running correctly through PowerShell commands.
- Build a docker image via command prompt and run the container.
- Accessing the Nginx default welcome page in a web browser at 'http://localhost:1802'
- Understanding basic Docker commands.