St. JOSEPH'S COLLEGE OF ENGINEERING

omr CHENNAI - 119

Placement Empowerment Program

Cloud Computing and DevOps Centre

BUILD AND RUN A CUSTOM DOCKER IMAGE

NAME: PRADEEPA AJITHA A DEPT: ECE

|  |
| --- |
| st. JOSEPH'S I@st. JOSEPH'S |
|

 AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY 

# 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.

## Overview

This POC demonstrates how to:

l. Create a Dockerized static website using NGINX. 2. Build a custom Docker image with an HTML file.

1. Run a Docker container to serve the HTML content.
2. Access the webpage via http://localhost:8080.

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

## Objectives

Understand the Dockerfile and key Docker commands (FROM, COPY, CMD).

* Learn to build and run a Docker contamenzed application.
* Gain hands-on experience in port mapping and container management.
* Demonstrate how Docker simplifies deployment compared to traditional setups

# Importance

* Portability — Docker ensures the application runs identically across different environments.
* Efficiency — Contamers are lightweight and require mmimal system resources.
* Scalability — Containers make it easier to scale applications without dependency conflicts.
* Faster Deployment — Eliminates the need for manual configurations and installations.

Step-by-Step Overview Step 1:

Install Docker (If Not Installed)

I . Open Command Prompt (cmd) and check if Docker is Installed

## docker - -version

1. If not installed, download and install

Docker from Docker's official website.

1. Ensure Docker Desktop is runnmg before proceeding

|  |
| --- |
| C : \Users\renit>docker ——version  Docker version 27.5.1 build 9f9e4@5 |

Step 2:

Create a Project Directory

Open Command Prompt and run:

mkdir my-docker-html && cd mydocker-html

mkdir my-docker-html \* Creates a new directory. cd my-docker-html Moves inside the directory.

C:\Users\renit>mkdir my—docker—html && cd my—docker—html

C : \Users\renit\my—docker—htmI>l

Step 3:

Create an HTML File (index.html)

Run this command to create an empty index.html file:

type nul > index.html

Open the file in Notepad:

## notepad index.html

Add the following content and save the file

 :\Users\renit\my—docker—htmt>type nut > index. html

C:\Users\renit\my—docker—htmI>notepad index. html

File Edit View

(html



html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale—I. O" > HTML

< / head>

< body> from Docker!

<p>This HTML page is served from inside a Docker container.</p>

< / bodY>

</html>

Step 4:

Create a Dockerfile

Run this command to create an empty

Dockerfile:

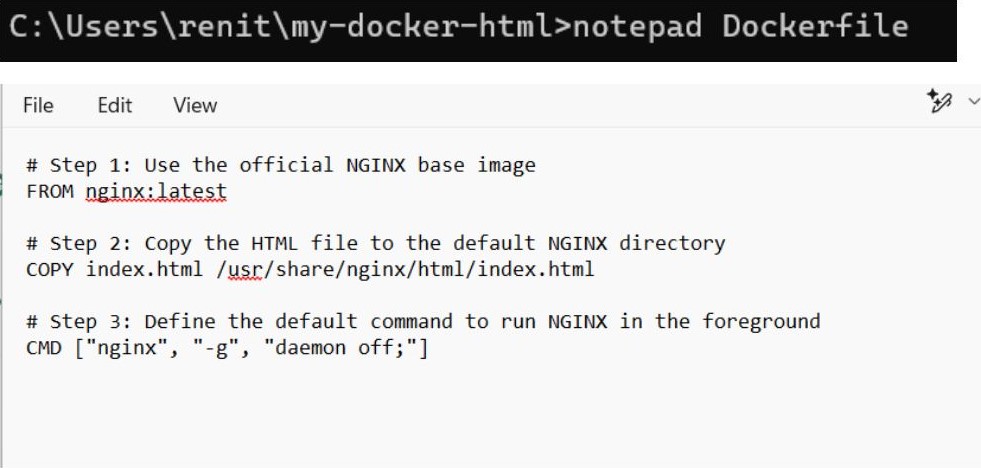
type nul > Dockerfile

Open the file in Notepad:

## notepad Dockerfile

Add the following content and save the file:

C:\Users\renit\my—docker—htmI>type nut >DockerfiIe



Step 5:

Build the Docker Image

Run the following command inside the mydocker-html directory:

docker build -t my-html-image .

-t my-html-image —+ Names the image myhtml-image.

 —+ Uses the current directory (where the

Dockerfile is located).

C: build —t

Building 67.qs (8/8) FINISHED docker desktop—Linux

> [internal] load build definition from Dockerfile O. 25

> transferring dockerfite: 321B 0.1s

> [internal] load metadata for docker. io/library/nginx: Latest 5. Is

> [auth] library/nginx: pull token for registry-I.docker. io

> [internal] load . dockerignore transferring context: 2B

> [internal] load build context O. Is transferring context: 357B

> [1/2] FROM 61 . 2s resolve O. Is sha256:9U3eaefec2eU2ccacc72ac657e13u7eadb2b9cb22828fac30f1Ueebba3d37ß88 1.21kB / 1.21kB 0.8s sha256:193f5ecb3e9f2eeu31b555078cce5e8df3db6ddc2e5Ud71ua10b99Ueq3ee98a3 1.UBkB / 1. U9kB 9.6s sha256: 9dd21ad5aUa6a856d82bb6bb61u7c3aad9ßa9768c3651c5577535ue7649bc7Ud ue6B / 1406B 1.2s sha256:d91uf92d532du16c7bgeadb2uU+1u+73+db3d2ead120264b749e3u270e82Uf3c 957B / 957B 1.2s sha256:513c3649bb1u8eca9aeuc73f32eb6b5a9e9e2ueuac18ae72+d56b8182u1d6738 626B / 626B 0.7s sha256:b+9acace21ua6c2363e8e3d98911+1fd7d1ba06a3ß83fea62+de36a6d1d8e27u u3 .95MB / 43 . 95MB 59. sha256: 7c+63256a31auccuuf6de+e8e1a+95363aee5Fa75f30a248d95cae684f87c53c 28 .22"B / 28. 22MB extracting sha256:7c+63256a31auccuuf6de+e8e1a+95363aee5+a75+3ßa248d95cae68uf87c53c extracting sha256:bf9acace21ua6c2363ß803d9ß911+1+d7d1baß6a3e83+oa62+dB36a6d1d8e274 extracting sha256: 513c3649bb1480ca9aouc73+320b6b5age9eueuac18ae72+d56b818241d673ß extracting sha256:d01u+92d532d416c7b9eadb244f14+73fdb3d2ead12ß26ub749e3427ß082u+3c extracting sha256: 9dd21ad5aua6a856d82bb6bb61u7c3eadgea9768c3651c5577535ue7649bc74d o.es extracting sha256:943ea0foc2e42ccacc72ac657013u7eadb2becb22828fac3ef1ueebba3d37988 extracting sha256:1e3f50cb3e9f20e431b555e78cce5e8df3db6ddc2e54dnua1ßb994e439e98a3

[2/2] COPY index . html /usr/share/nginx/htmL/index. html 9.3s > exporting to image 9.35 exporting layers exporting manifest sha256:6158ucge7138adb2abL121238219a9d795euvead65e10eeba351ucccdf3f7e03c exporting config exporting attestation manifest sha256: 5ecIq3ca1769afufS31fcfbdb3b3fe17deed9dfe4aea956a66d818e96f5c5dß5 exporting manifest list sha256:cbUdfcdd35519b91beb92b528b368609b6eU25d8U7ea39cd6e6a36cfae5a6ef6 naming to docker. io/tibrary 'my—htm I—image : latest

GnpÄckirg to docker ihrary/my—html latest

Step 6:

Run the Docker Contamer

Run this command to start a container from the image and expose it on port 8080:

docker run -d -p 8080:80 my-html-image

-d Runs the container in detached mode

(background).

-p 8080:80 —+ Maps port 8080 on your computer to port 80 inside the container.

|  |
| --- |
| C:\Users\renit\my—docker—htmI>docker run —d —p 8080 : 80 my—html—image a495U6+6c302160U29ee11857676002127de6f336b465e396c1285c2f9e7dd17 |

Step 7:

View the Web Page in a Browser

Open a browser and go to:

## http://localhost:8080

You should see the "Hello from Docker! " message.

e



Hello from Docker!

This HTML page is served from inside a Docker container.

## Outcome

Successfully built and ran a Docker container hosting an HTML webpage.

Accessed the webpage using http://localhost:8080.

Understood how Docker images and containers work for web applications.

Gained practical experience in contamerizatlon using Docker.