**Assignment 1**

You have been asked to:

● Pull ubuntu container

● Run this container, and map port 80 on the local

● Install apache2 on this container

● Check if you are able to access the apache page on your browser

**Solution Approach**

**Step 1: Create ec2 instances and install docker**

* We will create 1 ec2 instance
* Run the following codes to set up docker

# Install docker on ec2

sudo yum install -y docker

# Need to enable the docker service

sudo systemctl enable docker

sudo systemctl start docker

sudo usermod -aG docker $USER # attaching a user to a docker group

newgrp docker # activates the group with new user

getent group docker # check is user is attached

**Step 2: Pull the ubuntu container**

docker pull ubuntu

# Get public ip

Curl ifconfig.me

**Step 3: Run the container and map port 80 on the local**

docker run -it -p 80:80 --name apache-ubuntu ubuntu

**Step 4: Install apache2 on the container**

# Update package lists

apt update

# Install apache2

apt install -y apache2

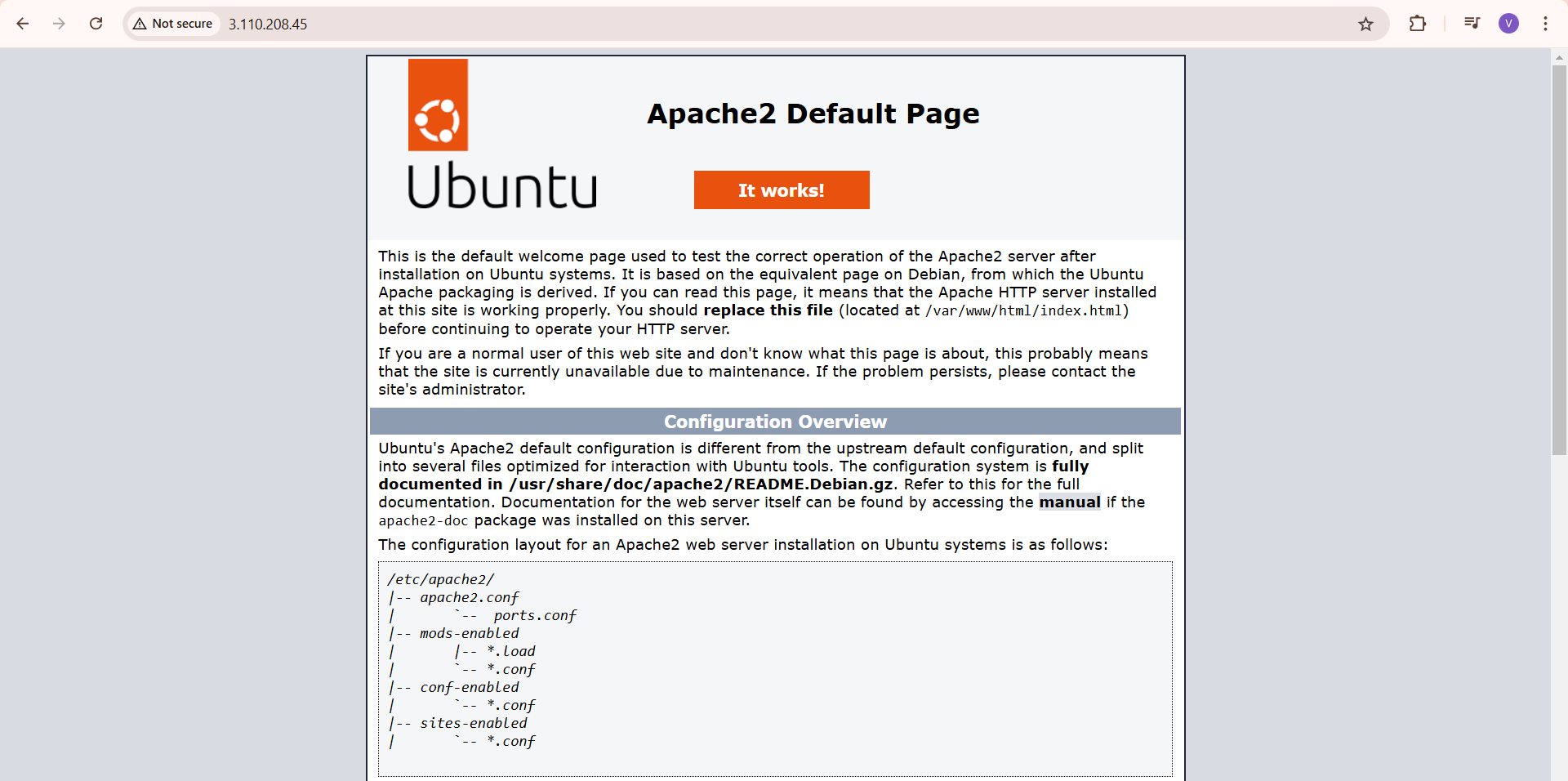
# Start apache service

service apache2 start

# Check if apache2 is running

service apache2 status

**Step 5: Check if you are able to access the Apache page in the browser**



**Assignment 2**

You have been asked to:

● Save the image created in Assignment 1 as a Docker image

● Launch container from this new image and map the port to 81

● Go inside the container and start the apache2 service

● Check if you are able to access it on the browser

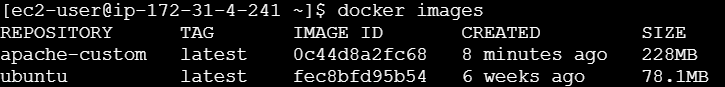
**Solution Approach**

**Step 1: Save the image created in Assignment 1 as Docker Image**

# Save the above image as another image

docker commit apache-ubuntu apache-custom

docker images



**Step 2: Launch container from this new image and map to port 81**

docker run -it -p 81:80 --name apache-custom-container apache-custom

**Step 3: Go inside the container and start apache2 service**

# Go inside the container

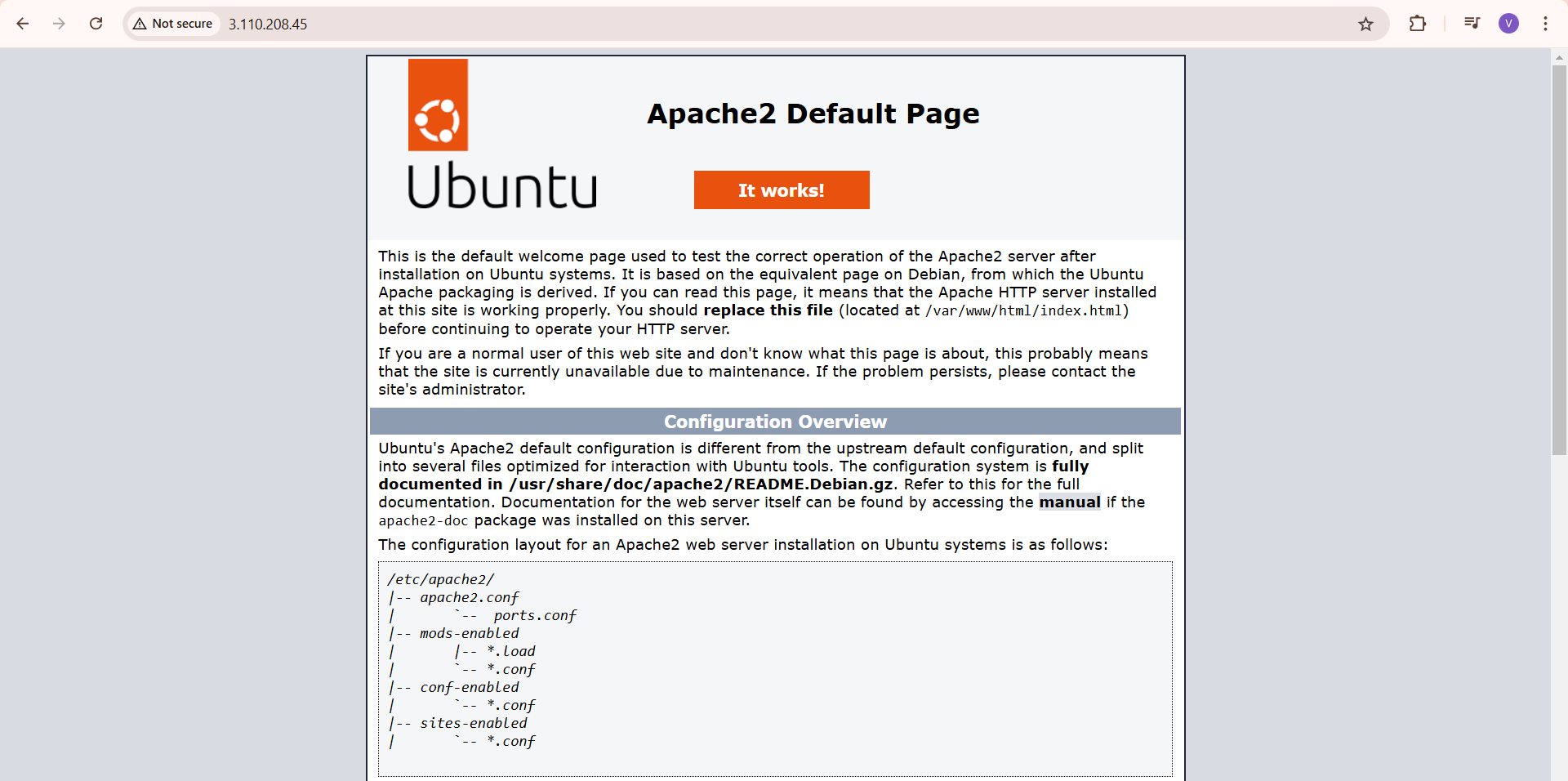
docker exec -it apache-custom-container bash

# Start the apache service

service apache2 start

service apache2 status

**Step 4: Check if you are able to access it in the browser**



**Assignment 3**

You have been asked to:

● Use the saved image in the previous assignment, upload this image on Dockerhub

● On a separate machine pull this dockerhub image, and launch it on port 80

● Start the apache2 service

● Verify if you are able to see the apache2 service

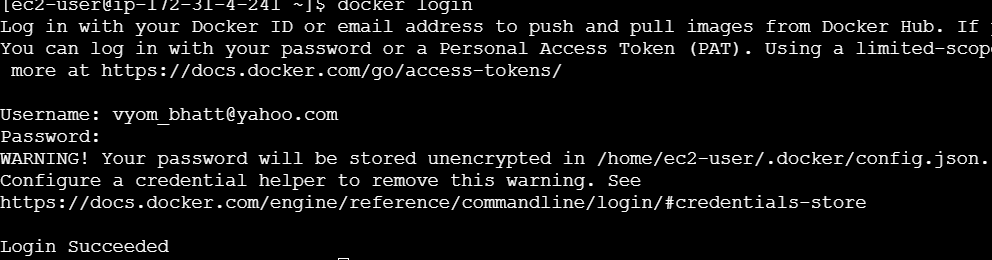
**Solution Approach**

**Step 1: Use the saved image in the previous assignment, upload this image on Dockerhub**

# Login to Docker hub

docker login

# Provide username and password

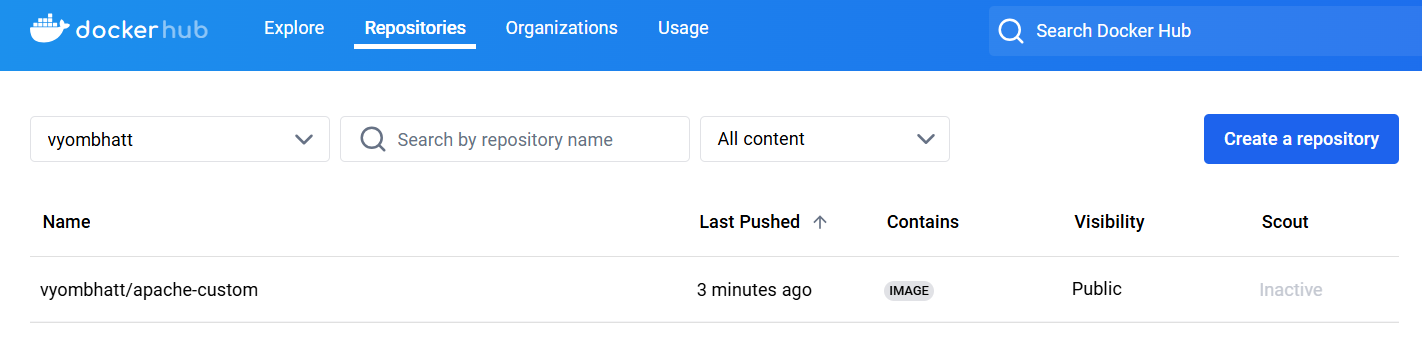


# Tag the image

docker tag apache-custom vyombhatt/apache-custom

# Push to docker hub

docker push vyombhatt/apache-custom



**Step 2: On a separate machine pull this dockerhub image, and launch it on port 80**

1. Create another ec2 instance and set up docker
2. Pull the docker image

docker login

docker pull vyombhatt/apache-custom

1. Launch the docker image on port 80

docker run -it -p 80:80 --name apache-custom-container vyombhatt/apache-custom

**Step 3: Go inside the container and start apache2 service**

# Go inside the container

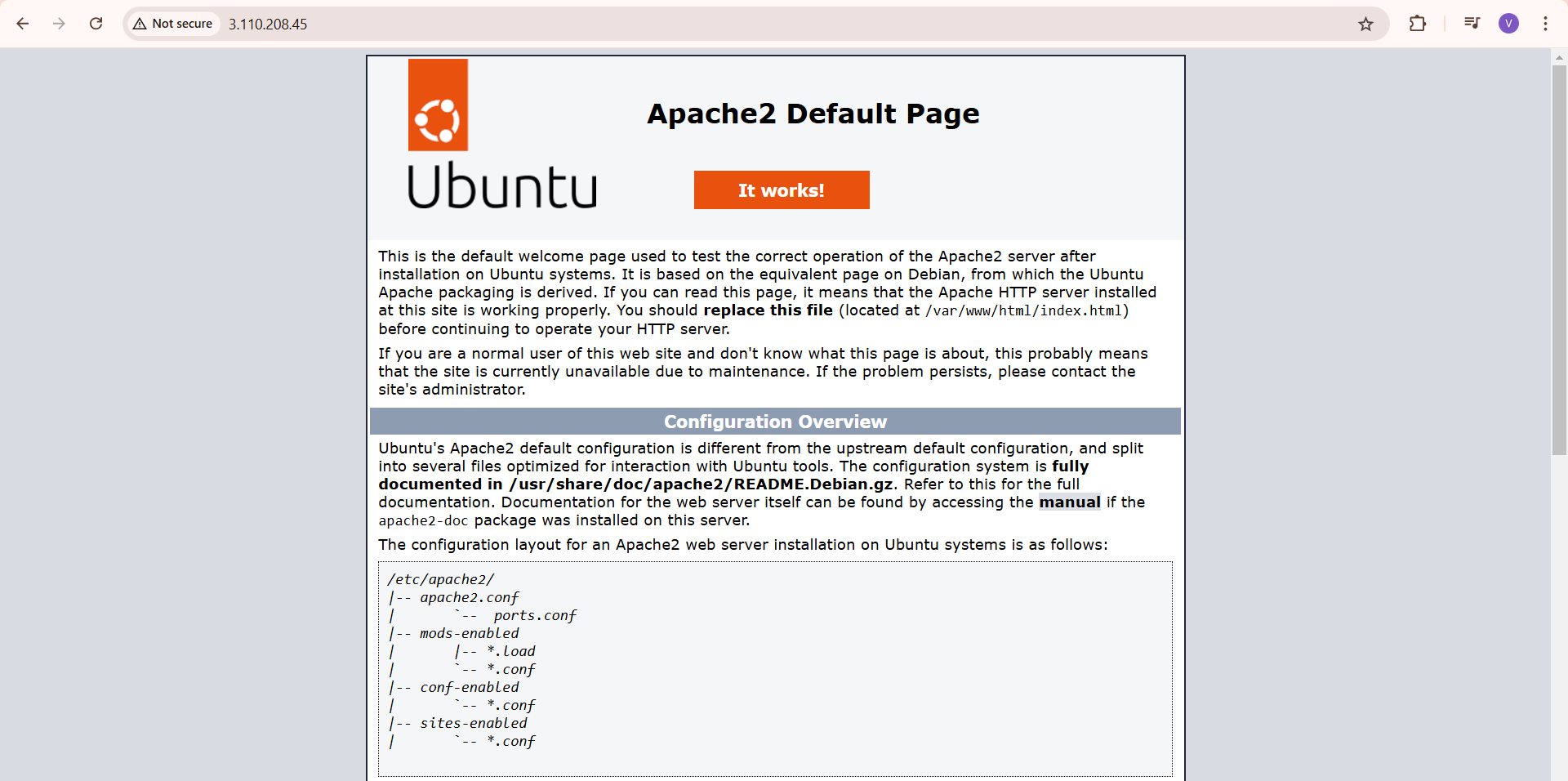
docker exec -it apache-custom-container bash

# Start the apache service

service apache2 start

service apache2 status

**Step 4: Check if you are able to access it in the browser**



**Assignment 4**

You have been asked to:

Create a dockerfile with the following specs:

● Ubuntu container

● Apache2 installed

● Apache2 should automatically run once the container starts

Submit the dockerfile, for assignment completion

**Solution Approach**

**Step 1: Creating the Dockerfile and saving it in working directory**

# Use the official Ubuntu base image

FROM ubuntu:latest

# Update package list and install Apache2

RUN apt-get update && \

apt-get install -y apache2 && \

apt-get clean

# Ensure Apache2 runs in the foreground

RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf

# Expose port 80 for web traffic

EXPOSE 80

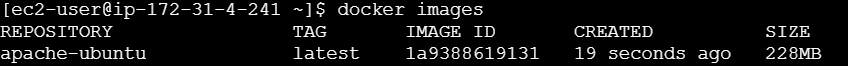
# Start Apache2 when the container starts

CMD ["apachectl", "-D", "FOREGROUND"]

**Step 2: Building the Dockerfile**

# Building the Dockerfile

docker build -t apache-ubuntu .



**Assignment 5**

You have been asked to:

● Create a sample HTML file

● Use the Dockerfile from the previous task and replace this sample HTML file inside the docker container with the default page

**Solution Approach**

**Step 1: Create a sample HTML file**

# Create a new folder

mkdir my-web-content

cd my-web-content

# Create a new index.html file

touch index.html

sudo vi index.html

*index.html file*

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Custom Apache Page</title>

</head>

<body>

<h1>Welcome to My Custom Apache Page!</h1>

<p>This is a sample HTML file served by Apache inside a Docker container.</p>

</body>

</html>

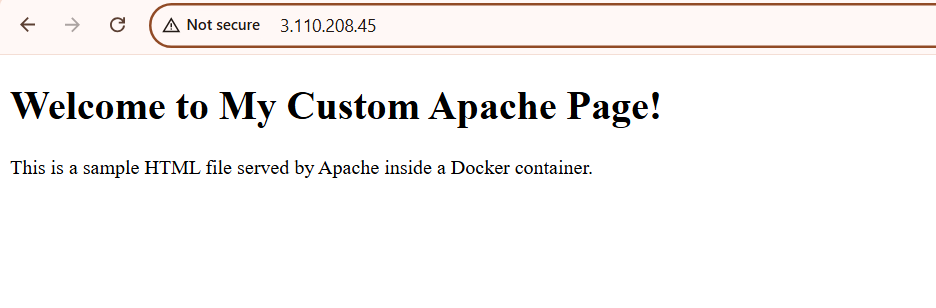
**Step 2: Use Dockerfile from previous task and replace this sample HTML file inside the docker container with the default page**

We will use volume mounting to mount the directory in local into the docker file

# Docker run using volume mounting

docker run -it -p 80:80 --name apache-volume-container -v $(pwd)/my-web-content:/var/www/html apache-custom

**Step 3: Preview the page on the browser**

****

**Assignment 6**

You have been asked to:

● Launch the Apache2 container created in previous module

● Create a Docker volume on /var/www/html

Please submit the commands, in order to complete this assignment

**Solution Approach**

**Step 1: Launch the apache2 container**

# Launch the container from image

docker run -it --name apache2 apache-custom

**Step 2: Create a Docker volume on /var/www/html**

# Create a volume

docker volume create apache-html-data

**Assignment 7**

You have been asked to:

● Use the apache2 container created in previous module, create a bind mount on /var/www/html to replace html files dynamically

Submit the commands, to complete the assignment

**Solution Approach**

**Step 1: Launch the apache2 container**

# Creating a bind mount

docker run -d -p 80:80 --name apache2 -v apache-html-data:/var/www/html apache-custom

**Assignment 8**

You have been asked to:

● Create 5 custom container, with 5 different default pages

● Using docker compose, deploy these 5 containers on port 81, 82, 83, 84 and 85 respectively

**Solution Approach**

**Step 1: Create a new directory and subdirectories. Inside each subdirectory, add an index.html file**

mkdir multi-container-apache

cd multi-container-apache

*Default Page 1:*

# Create sub directory

mkdir container1

cd mkdir container1

# Create new index.html file

touch index.html

# Write into the index.html file

sudo vi index.html

<h1>Welcome to Container 1</h1>

<p>This is the default page for Container 1.</p>

*Default Page 2:*

# Create sub directory

mkdir container2

cd mkdir container2

# Create new index.html file

touch index.html

# Write into the index.html file

sudo vi index.html

<h1>Welcome to Container 2</h1>

<p>This is the default page for Container 2.</p>

Repeat the above steps and create 5 different sub-directories

**Step 2: Write the docker compose file**

version: '3.0'

services:

container1:

image: httpd:latest

container\_name: container1

ports:

- "81:80"

volumes:

- ./container1:/usr/local/apache2/htdocs/

container2:

image: httpd:latest

container\_name: container2

ports:

- "82:80"

volumes:

- ./container2:/usr/local/apache2/htdocs/

container3:

image: httpd:latest

container\_name: container3

ports:

- "83:80"

volumes:

- ./container3:/usr/local/apache2/htdocs/

container4:

image: httpd:latest

container\_name: container4

ports:

- "84:80"

volumes:

- ./container4:/usr/local/apache2/htdocs/

container5:

image: httpd:latest

container\_name: container5

ports:

- "85:80"

volumes:

- ./container5:/usr/local/apache2/htdocs/