

# Lab Exercise 21- Building a Docker Image for an HTML App Using Nginx

## 1. Setup

You will need:

- Docker installed on your machine.
- A simple HTML file for the app.

## 2. Step 1: Create the HTML File

Create a directory for your HTML app and place an index.html file in it.

```
mkdir nginx-html-app
```

```
cd nginx-html-app
```

Inside the nginx-html-app directory, create the HTML file.

```
touch index.html
```

Edit the index.html file with the following content (or any custom HTML content you want):

```
<!DOCTYPE html>

<html>

<head>

  <title>Welcome to My Nginx HTML App</title>

</head>

<body>

  <h1>Hello, Nginx Docker!</h1>

  <p>This is a simple HTML app served by Nginx in a Docker container.</p>
```

```
</body>
```

```
</html>
```

### 3. Step 2: Create a Dockerfile

In the same directory, create a Dockerfile. This file will define how to build the Docker image using Nginx as the base image.

```
touch Dockerfile
```

Edit the Dockerfile and add the following content:

```
FROM nginx:latest
```

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

```
EXPOSE 80
```

### 4. Step 3: Build the Docker Image

Now that you have the Dockerfile and index.html, it's time to build the Docker image.

Run the following command to build the image, giving it a tag (e.g., nginx-html-app):

```
docker build -t nginx-html-app .
```

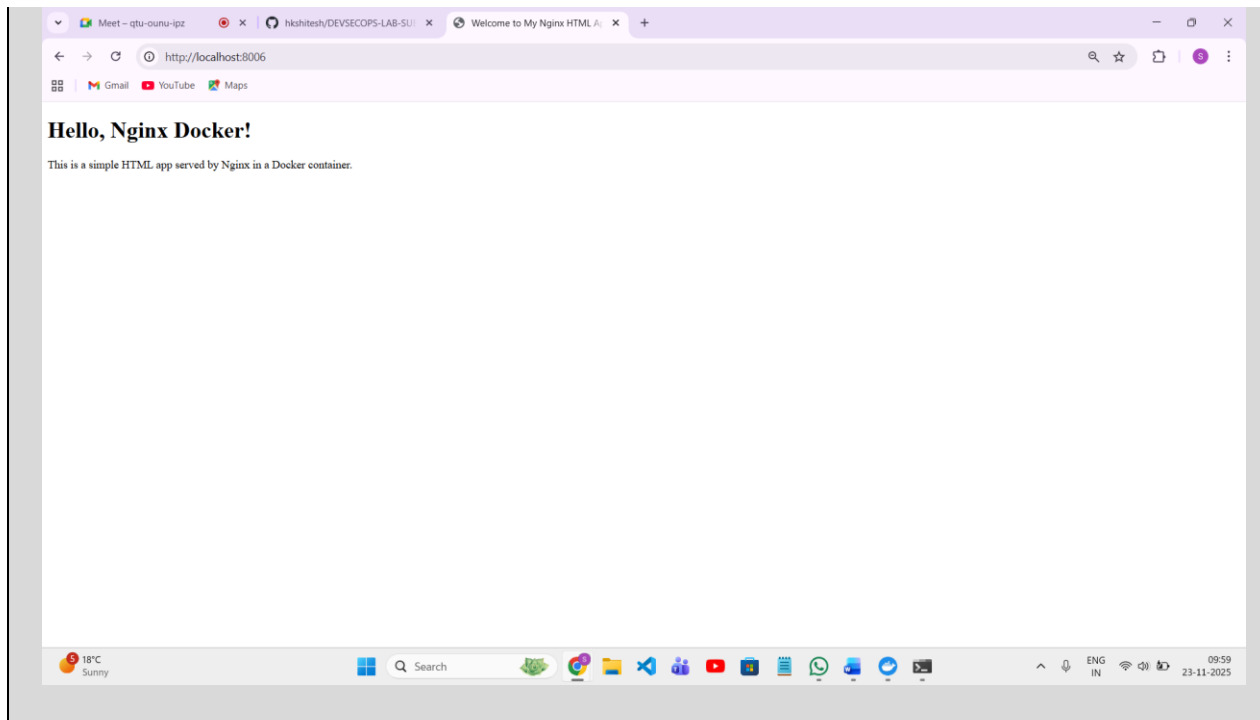
```
Terminal
→ nginx-html-app git:(main) × docker build -t nginx-html-app .
[+] Building 73.3s (7/7) FINISHED
⇒ [internal] load build definition from Dockerfile
⇒ [internal] load build definition from Dockerfile
⇒ [internal] load metadata for docker.io/library/nginx:latest
⇒ [internal] load .dockerignore
⇒ [internal] load build context
⇒ [internal] load build context
⇒ [internal] load build context
⇒ [1/2] FROM docker.io/library/nginx:latest@sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c52a42
⇒ [2/2] COPY index.html /usr/share/nginx/html/
⇒ exporting to image
⇒ exporting layers
⇒ exporting manifest sha256:70cf04a946efd0b1aed257baf4e00c4fd0b83894008fb2553fff015c4e90fa74
⇒ exporting config sha256:2d68fb1fb4f4785b018e351ac905a7353f0c2a5c0d513f294f3d67059043ff04
⇒ exporting attestations manifest sha256:2f767a5f1a640b2ae215fd9a42c19bfb411265fa9ad17ef4cca445677e6eb207
⇒ exporting manifest list sha256:c3db243c38fc8e59c1a03153953ef908adc26596dff9176c60259720279b7b0f
```

Docker will use the Nginx base image, copy your index.html into the appropriate directory, and build the image.

## 5. Step 4: Run the Docker Container

After building the image, you can run the container with the following command:

```
docker run -d -p 8006:80 nginx-html-app
```



This command runs the container in detached mode (-d) and maps port 8006 on your host machine to port 80 inside the container, where Nginx is serving your HTML app.

## 6. Step 5: Verify

Open a browser and go to <http://localhost:8006>. You should see your HTML page with the message “Hello, Nginx Docker!”.

## 7. Step 6: Stop and Remove the Container

Once you're done, you can stop and remove the container:

`docker ps` # to see running containers

```
+ nginx-html-app git:(main) x docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
a61243c194ca   nginx-html-app   "/docker-entrypoint..." About a minute ago Up About a minute   0.0.0.0:8006->80/tcp,
[::]:8006->80/tcp
4c67be114138   jenkins/jenkins:jdk21   "/usr/bin/tini -- /u..." 2 months ago   Up 5 minutes      8080/tcp, 50000/tcp
```

`docker stop <container-id>`

```
→ nginx-html-app git:(main) × docker stop a61243c194ca  
a61243c194ca
```

docker rm <container-id>

```
→ nginx-html-app git:(main) × docker rm a61243c194ca  
a61243c194ca
```