

# 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>
```

```
</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:

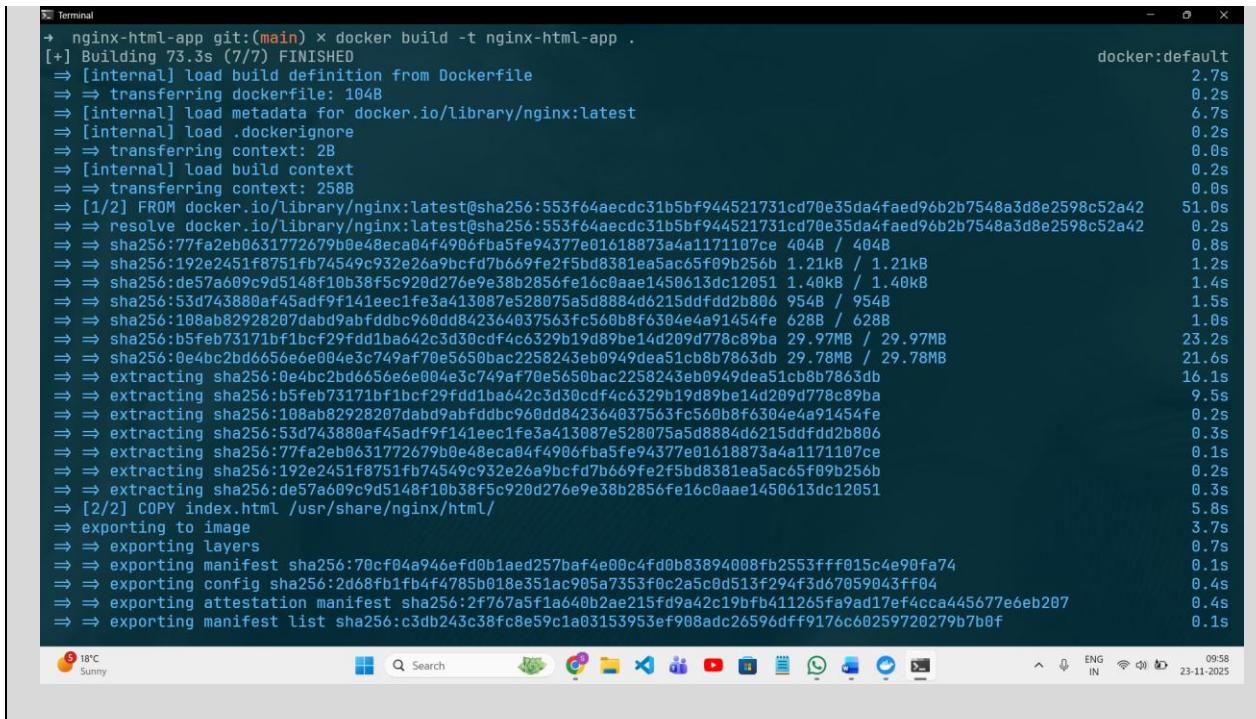
```
FROM nginx:latest  
COPY index.html /usr/share/nginx/html/  
EXPOSE 80
```

### 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 metadata for docker.io/library/nginx:latest
  → [internal] load .dockerignore
  → [internal] transfer context: 28
  → [internal] load build context
  → [internal] transfer context: 2588
[1/2] FROM docker.io/library/nginx:latest@sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c52a42
      2.7s
  → resolve docker.io/library/nginx:latest@sha256:553f64aecdc31b5bf944521731cd70e35da4faed96b2b7548a3d8e2598c52a42
      0.2s
  → sha256:77fa2eb0631772679b0e48eca04f4906fba5fe94377e01618873a4a117107ce 404B / 404B
      6.7s
  → sha256:192e2451f8751fb74549c932e26a9bcfd7b669fe2f5bd8381ea5ac65f09b256b 1.21kB / 1.21kB
      0.2s
  → sha256:de57a609c9d5148f10b38f5c920d27e9e3802856fe16cb8aae1450613dc12051 1.40kB / 1.40kB
      0.0s
  → sha256:53d743880af45adff9f141eec1fe3a413087e528075a5d8884d6215ddfd2b806 954B / 954B
      0.0s
  → sha256:108ab82928207dabd9abfddbc960dd842364037563fc560b8f6304e4a91454fe 628B / 628B
      0.0s
  → sha256:b5feb73171bf1bcf29fd1ba642c3d30cdf4c6329b19d89be14d209d778c89ba 29.97MB / 29.97MB
      23.2s
  → sha256:8e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db 29.78MB / 29.78MB
      21.6s
  → extracting sha256:8e4bc2bd6656e6e004e3c749af70e5650bac2258243eb0949dea51cb8b7863db
      16.1s
  → extracting sha256:b5feb73171bf1bcf29fd1ba642c3d30cdf4c6329b19d89be14d209d778c89ba
      9.5s
  → extracting sha256:108ab82928207dabd9abfddbc960dd842364037563fc560b8f6304e4a91454fe
      0.2s
  → extracting sha256:53d743880af45adff9f141eec1fe3a413087e528075a5d8884d6215ddfd2b806
      0.3s
  → extracting sha256:77fa2eb0631772679b0e48eca04f4906fba5fe94377e01618873a4a117107ce
      0.1s
  → extracting sha256:192e2451f8751fb74549c932e26a9bcfd7b669fe2f5bd8381ea5ac65f09b256b
      0.2s
  → extracting sha256:d57a609c9d5148f10b38f5c920d27e9e38b2856fe16cb8aae1450613dc12051
      0.3s
[2/2] COPY index.html /usr/share/nginx/html/
      5.8s
  → exporting to image
      3.7s
  → exporting layers
      0.7s
  → exporting manifest sha256:70cf04a946ef0b1aed257baf4e00c4fd0b83894008fb2553fff015c4e90fa74
      0.1s
  → exporting config sha256:2d68fb1fb4f4785b018e551ac985a7353f02a5c0d513f294f3d67859043ff04
      0.4s
  → exporting attestation manifest sha256:2f767a5f1a640b2ae215fd9a42c19fb411265fa9ad17ef4cca445677e6eb207
      0.4s
  → exporting manifest list sha256:c3db243c38fc8e59c1a03153953ef908adc26596dff9176c60259720279b7b0f
      0.1s
```

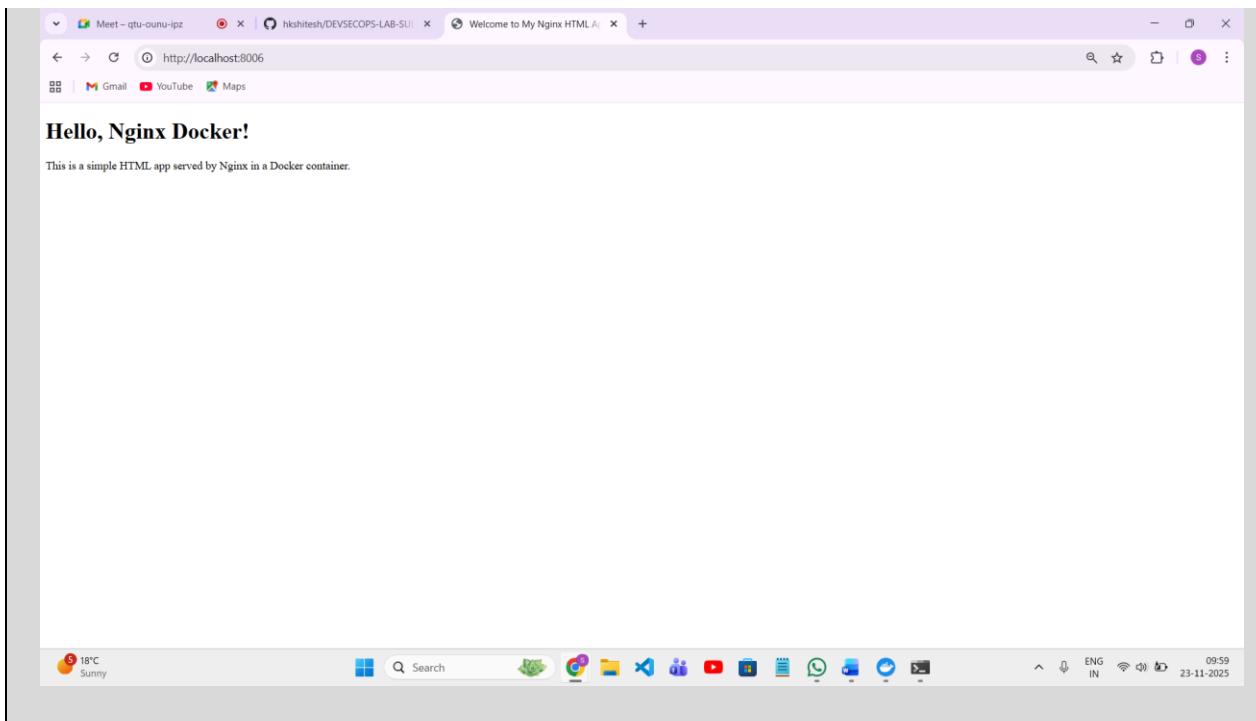
The screenshot shows a terminal window on a Linux desktop. The terminal output displays the Docker build process for an 'nginx-html-app' image. It shows the download of the Nginx base image, copying of the 'index.html' file, and the final export of the manifest. The desktop interface includes a weather widget (18°C, sunny), a search bar, and various system icons.

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
```

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	jolly_banach				
4c67be114138	jenkins/jenkins:jdk21	"/usr/bin/tini -- /u..."	2 months ago	Up 5 minutes	8080/tcp, 50000/tcp
	angry_pike				

```
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
```