Docker Project 01

Project Overview

In this project, you'll go through all three lifecycles of Docker: pulling an image and creating a container, modifying the container and creating a new image, and finally, creating a Dockerfile to build and deploy a web application.

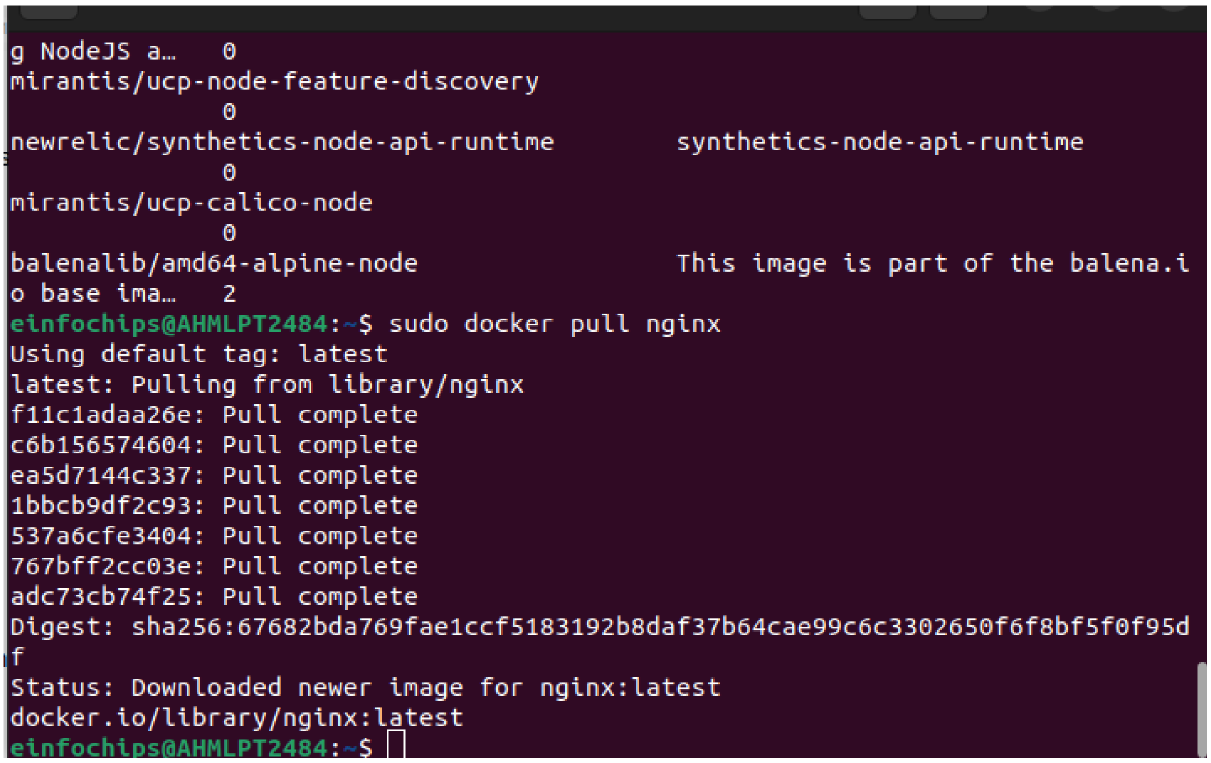
Part 1: Creating a Container from a Pulled Image

Objective: Pull the official Nginx image from Docker Hub and run it as a container.

Steps:

Pull the Nginx Image:

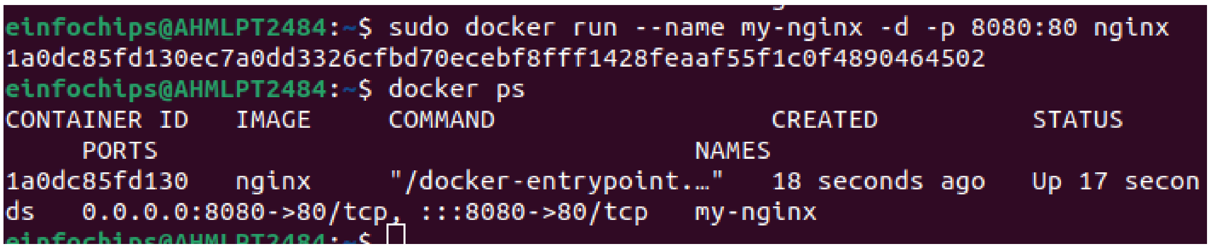
```
docker pull nginx
```



```
g NodeJS a…   0
mirantis/ucp-node-feature-discovery
              0
newrelic/synthetics-node-api-runtime        synthetics-node-api-runtime
              0
mirantis/ucp-calico-node
              0
balenalib/amd64-alpine-node                 This image is part of the balena.i
o base ima…   2
einfochips@AHMLPT2484:~$ sudo docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
f11c1adaa26e: Pull complete
c6b156574604: Pull complete
ea5d7144c337: Pull complete
1bbcb9df2c93: Pull complete
537a6cfe3404: Pull complete
767bff2cc03e: Pull complete
adc73cb74f25: Pull complete
Digest: sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95d
f
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
einfochips@AHMLPT2484:~$
```

Run the Nginx Container:

```
docker run --name my-nginx -d -p 8080:80 nginx
```

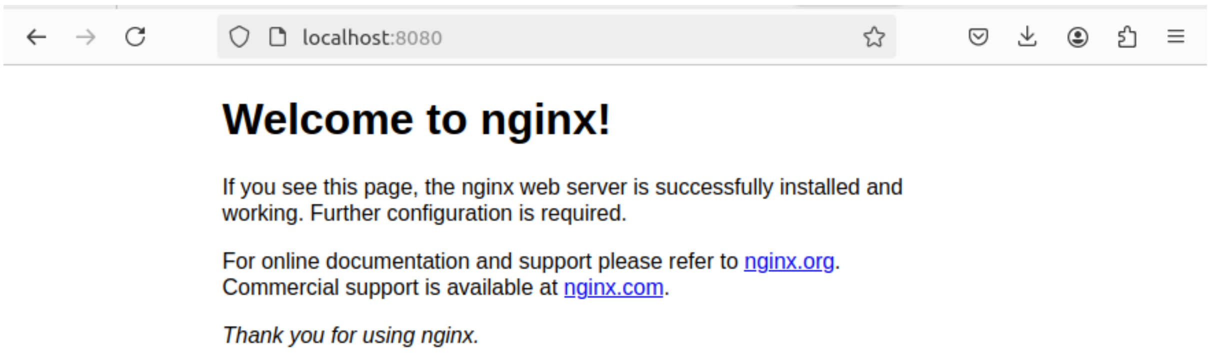


```
einfochips@AHMLPT2484:~$ sudo docker run --name my-nginx -d -p 8080:80 nginx
1a0dc85fd130ec7a0dd3326cfbd70ecebf8fff1428feaaf55f1c0f4890464502
einfochips@AHMLPT2484:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
     PORTS                                   NAMES
1a0dc85fd130   nginx     "/docker-entrypoint.…"   18 seconds ago   Up 17 secon
ds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-nginx
```

Verify the Container is Running:

```
docker ps
```

1.
    - Visit `http://localhost:8080` in browser. The Nginx welcome page.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Part 2: Modifying the Container and Creating a New Image

Objective: Modify the running Nginx container to serve a custom HTML page and create a new image from this modified container.

Steps:

Access the Running Container:

```
docker exec -it my-nginx /bin/bash
```

1.

Create a Custom HTML Page:

```
echo "<html><body><h1>Hello from Docker!</h1></body></html>" >
/usr/share/nginx/html/index.html
```

2.

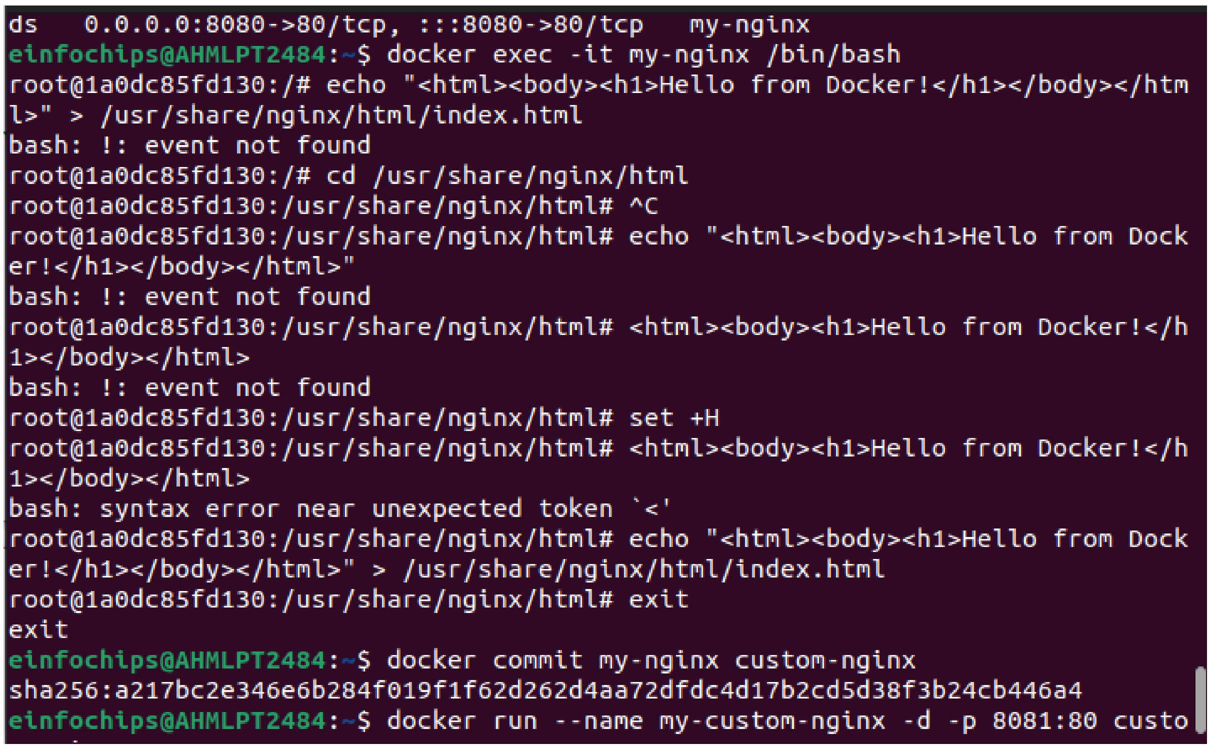
Exit the Container:

```
exit
```

3.

Commit the Changes to Create a New Image:

```
docker commit my-nginx custom-nginx
```



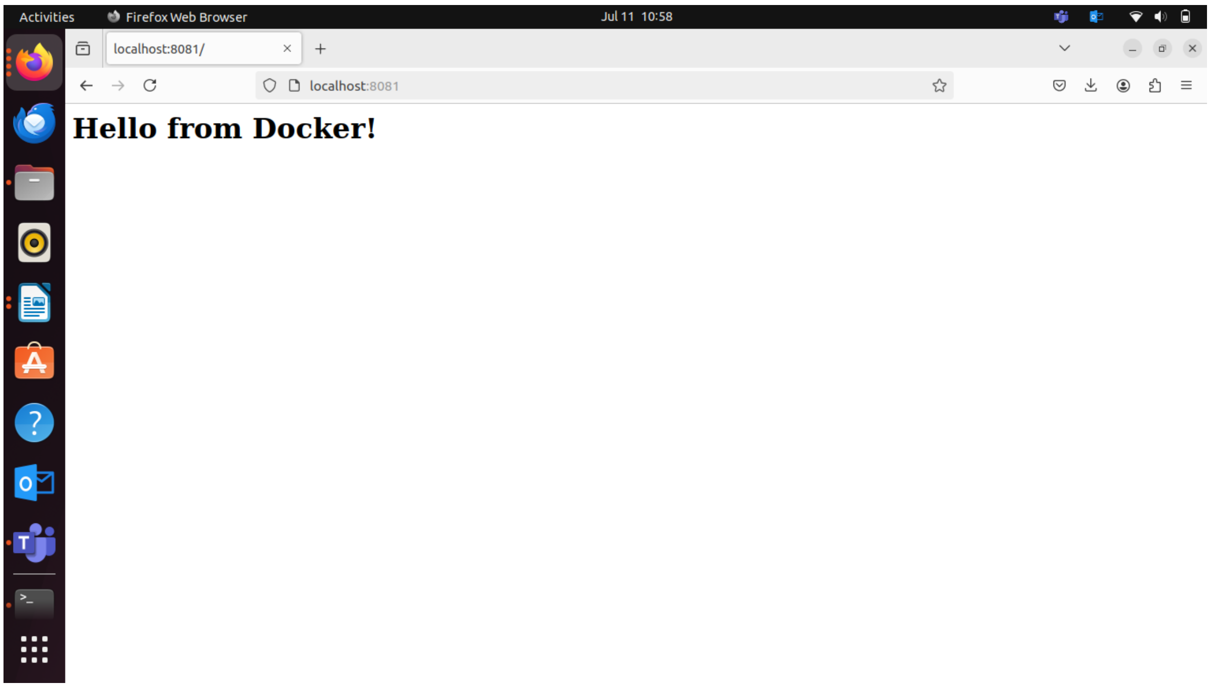
```
ds   0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-nginx
einfochips@AHMLPT2484:~$ docker exec -it my-nginx /bin/bash
root@1a0dc85fd130:/# echo "<html><body><h1>Hello from Docker!</h1></body></htm
l>" > /usr/share/nginx/html/index.html
bash: !: event not found
root@1a0dc85fd130:/# cd /usr/share/nginx/html
root@1a0dc85fd130:/usr/share/nginx/html# ^C
root@1a0dc85fd130:/usr/share/nginx/html# echo "<html><body><h1>Hello from Dock
er!</h1></body></html>"
bash: !: event not found
root@1a0dc85fd130:/usr/share/nginx/html# <html><body><h1>Hello from Docker!</h
1></body></html>
bash: !: event not found
root@1a0dc85fd130:/usr/share/nginx/html# set +H
root@1a0dc85fd130:/usr/share/nginx/html# <html><body><h1>Hello from Docker!</h
1></body></html>
bash: syntax error near unexpected token `<'
root@1a0dc85fd130:/usr/share/nginx/html# echo "<html><body><h1>Hello from Dock
er!</h1></body></html>" > /usr/share/nginx/html/index.html
root@1a0dc85fd130:/usr/share/nginx/html# exit
exit
einfochips@AHMLPT2484:~$ docker commit my-nginx custom-nginx
sha256:a217bc2e346e6b284f019f1f62d262d4aa72dfdc4d17b2cd5d38f3b24cb446a4
einfochips@AHMLPT2484:~$ docker run --name my-custom-nginx -d -p 8081:80 custo
```

4.

Run a Container from the New Image:

```
docker run --name my-custom-nginx -d -p 8081:80 custom-nginx
```

5. **Verify the New Container:**
    - Visit `http://localhost:8081` in your browser. You should see your custom HTML page.



Hello from Docker!

Part 3: Creating a Dockerfile to Build and Deploy a Web Application

Objective: Write a Dockerfile to create an image for a simple web application and run it as a container.

Steps:

Create a Project Directory:

```
mkdir my-webapp
cd my-webapp
```

1.
2. **Create a Simple Web Application:**

Create an `index.html` file:

```
<!DOCTYPE html>
<html>
<body>
```

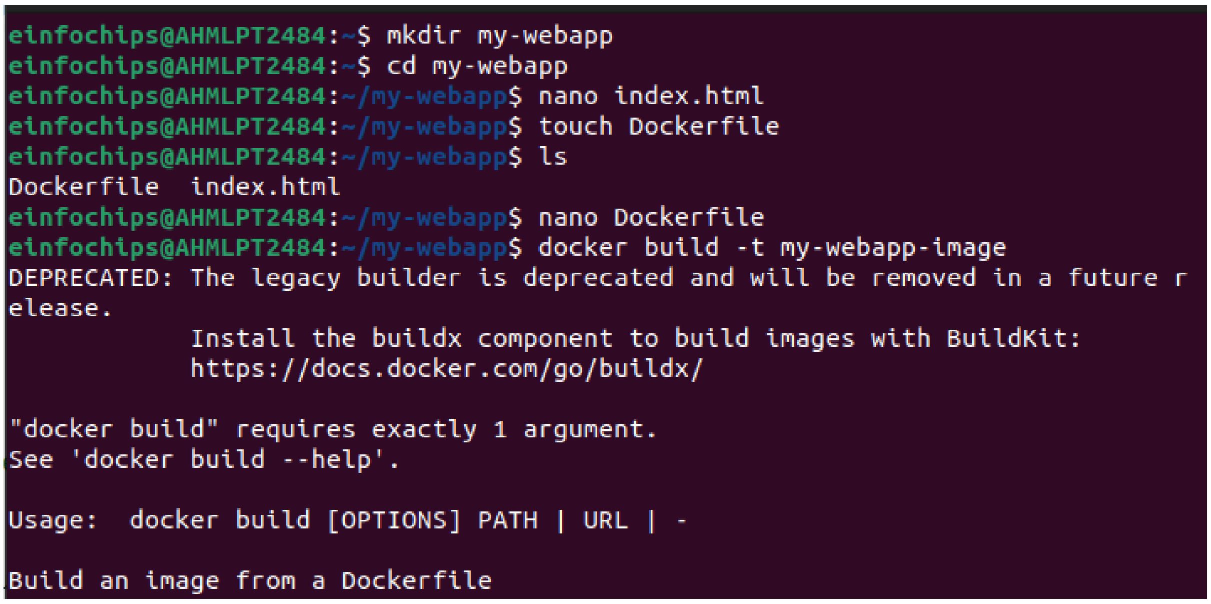
```
    <h1>Hello from My Web App!</h1>
</body>
</html>
```

- Save this file in the `my-webapp` directory.

3. **Write the Dockerfile:**

Create a `Dockerfile` in the `my-webapp` directory with the following content:

```
# Use the official Nginx base image
FROM nginx:latest

# Copy the custom HTML file to the appropriate location
COPY index.html /usr/share/nginx/html/

# Expose port 80
EXPOSE 80
```



```
einfochips@AHMLPT2484:~$ mkdir my-webapp
einfochips@AHMLPT2484:~$ cd my-webapp
einfochips@AHMLPT2484:~/my-webapp$ nano index.html
einfochips@AHMLPT2484:~/my-webapp$ touch Dockerfile
einfochips@AHMLPT2484:~/my-webapp$ ls
Dockerfile  index.html
einfochips@AHMLPT2484:~/my-webapp$ nano Dockerfile
einfochips@AHMLPT2484:~/my-webapp$ docker build -t my-webapp-image
DEPRECATED: The legacy builder is deprecated and will be removed in a future r
elease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
```

-

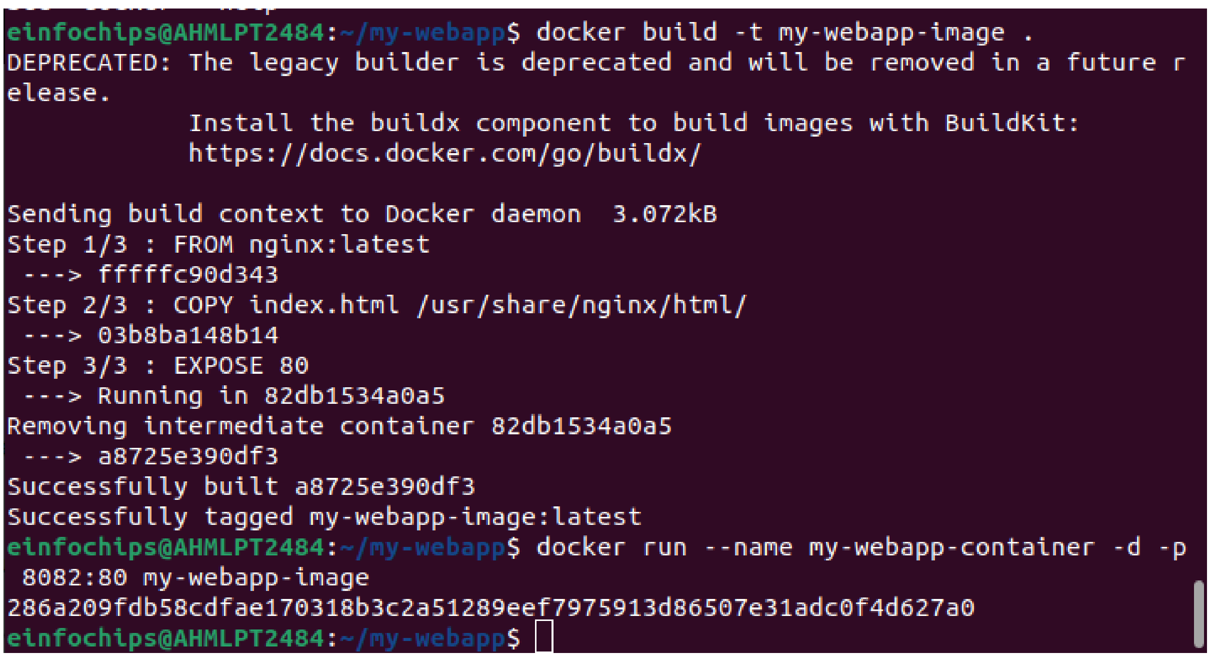
Build the Docker Image:

```
docker build -t my-webapp-image .
```

4.

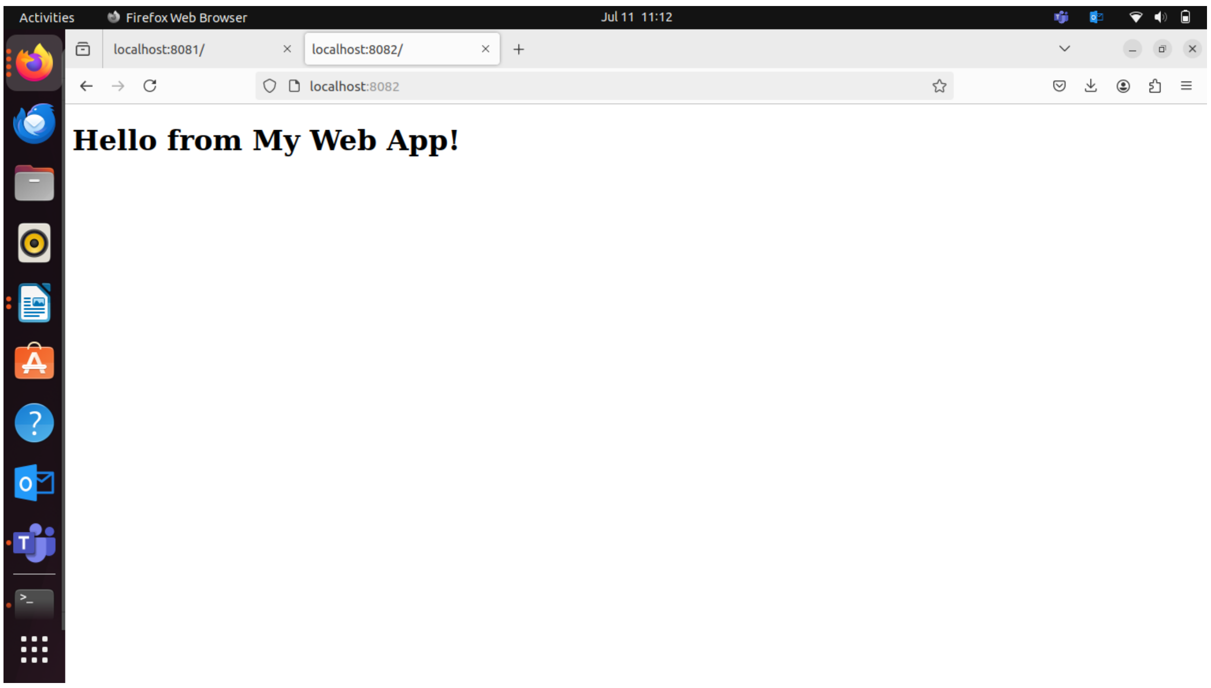
Run a Container from the Built Image:

```
docker run --name my-webapp-container -d -p 8082:80 my-webapp-image
```



```
einfochips@AHMLPT2484:~/my-webapp$ docker build -t my-webapp-image .
DEPRECATED: The legacy builder is deprecated and will be removed in a future r
elease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.072kB
Step 1/3 : FROM nginx:latest
 ---> fffffc90d343
Step 2/3 : COPY index.html /usr/share/nginx/html/
 ---> 03b8ba148b14
Step 3/3 : EXPOSE 80
 ---> Running in 82db1534a0a5
Removing intermediate container 82db1534a0a5
 ---> a8725e390df3
Successfully built a8725e390df3
Successfully tagged my-webapp-image:latest
einfochips@AHMLPT2484:~/my-webapp$ docker run --name my-webapp-container -d -p
 8082:80 my-webapp-image
286a209fdb58cdfae170318b3c2a51289eef7975913d86507e31adc0f4d627a0
einfochips@AHMLPT2484:~/my-webapp$
```

5. **Verify the Web Application:**
   - Visit `http://localhost:8082` in your browser. You should see your custom web application.



Hello from My Web App!

Part 4: Cleaning Up

Objective: Remove all created containers and images to clean up your environment.

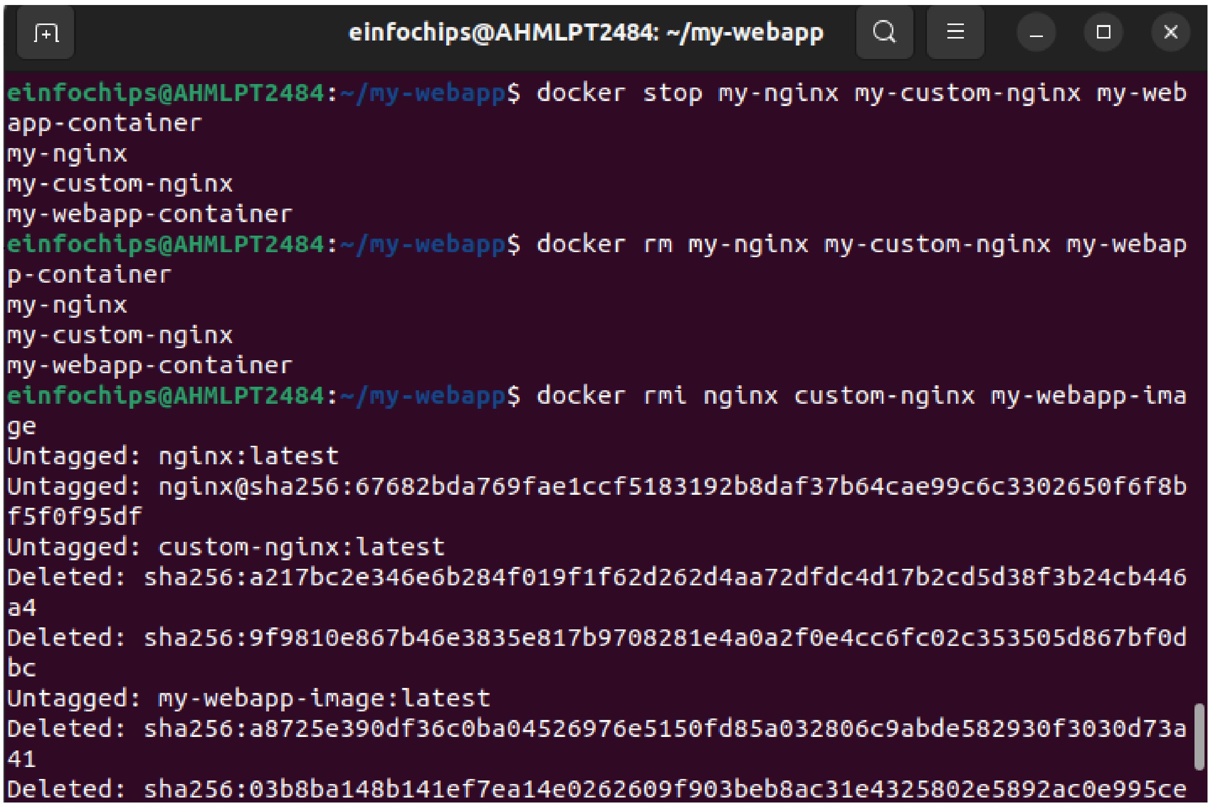
Steps:

Stop and Remove the Containers:

```
docker stop my-nginx my-custom-nginx my-webapp-container
docker rm my-nginx my-custom-nginx my-webapp-container
```

1. **Remove the Images:**

   ```
   docker rmi nginx custom-nginx my-webapp-image
   ```



```
einfochips@AHMLPT2484:~/my-webapp$ docker stop my-nginx my-custom-nginx my-web
app-container
my-nginx
my-custom-nginx
my-webapp-container
einfochips@AHMLPT2484:~/my-webapp$ docker rm my-nginx my-custom-nginx my-webap
p-container
my-nginx
my-custom-nginx
my-webapp-container
einfochips@AHMLPT2484:~/my-webapp$ docker rmi nginx custom-nginx my-webapp-ima
ge
Untagged: nginx:latest
Untagged: nginx@sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8b
f5f0f95df
Untagged: custom-nginx:latest
Deleted: sha256:a217bc2e346e6b284f019f1f62d262d4aa72dfdc4d17b2cd5d38f3b24cb446
a4
Deleted: sha256:9f9810e867b46e3835e817b9708281e4a0a2f0e4cc6fc02c353505d867bf0d
bc
Untagged: my-webapp-image:latest
Deleted: sha256:a8725e390df36c0ba04526976e5150fd85a032806c9abde582930f3030d73a
41
Deleted: sha256:03b8ba148b141ef7ea14e0262609f903beb8ac31e4325802e5892ac0e995ce
```

Docker Project 02

Project Overview

In this advanced project, you'll build a full-stack application using Docker. The application will consist of a front-end web server (Nginx), a back-end application server (Node.js with Express), and a PostgreSQL database. You will also set up a persistent volume for the database and handle inter-container communication. This project will take more time and involve more detailed steps to ensure thorough understanding.

Part 1: Setting Up the Project Structure

Objective: Create a structured project directory with necessary configuration files.

Steps:

Create the Project Directory:

```
mkdir fullstack-docker-app
cd fullstack-docker-app
```

1.

Create Subdirectories for Each Service:

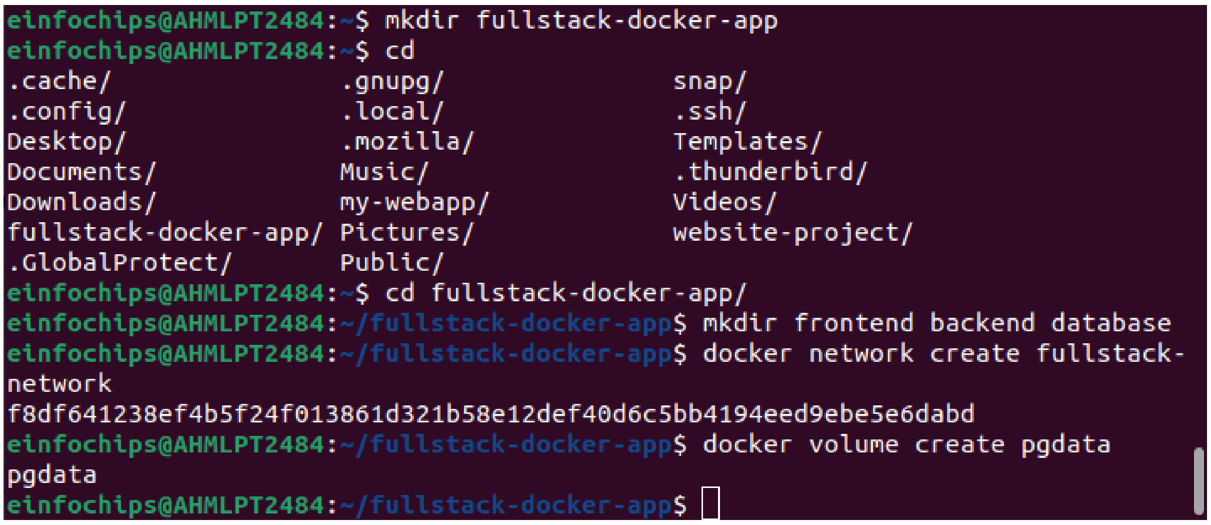
```
mkdir frontend backend database
```

2. **Create Shared Network and Volume:**
    - Docker allows communication between containers through a shared network.

```
docker network create fullstack-network
```

3.
    - Create a volume for the PostgreSQL database.

```
docker volume create pgdata
```



```
einfochips@AHMLPT2484:~$ mkdir fullstack-docker-app
einfochips@AHMLPT2484:~$ cd
.cache/              .gnupg/              snap/
.config/             .local/              .ssh/
Desktop/             .mozilla/            Templates/
Documents/           Music/               .thunderbird/
Downloads/           my-webapp/           Videos/
fullstack-docker-app/ Pictures/           website-project/
.GlobalProtect/      Public/
einfochips@AHMLPT2484:~$ cd fullstack-docker-app/
einfochips@AHMLPT2484:~/fullstack-docker-app$ mkdir frontend backend database
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker network create fullstack-
network
f8df641238ef4b5f24f013861d321b58e12def40d6c5bb4194eed9ebe5e6dabd
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker volume create pgdata
pgdata
einfochips@AHMLPT2484:~/fullstack-docker-app$
```

Part 2: Setting Up the Database

Objective: Set up a PostgreSQL database with Docker.

Steps:

1. **Create a Dockerfile for PostgreSQL:**

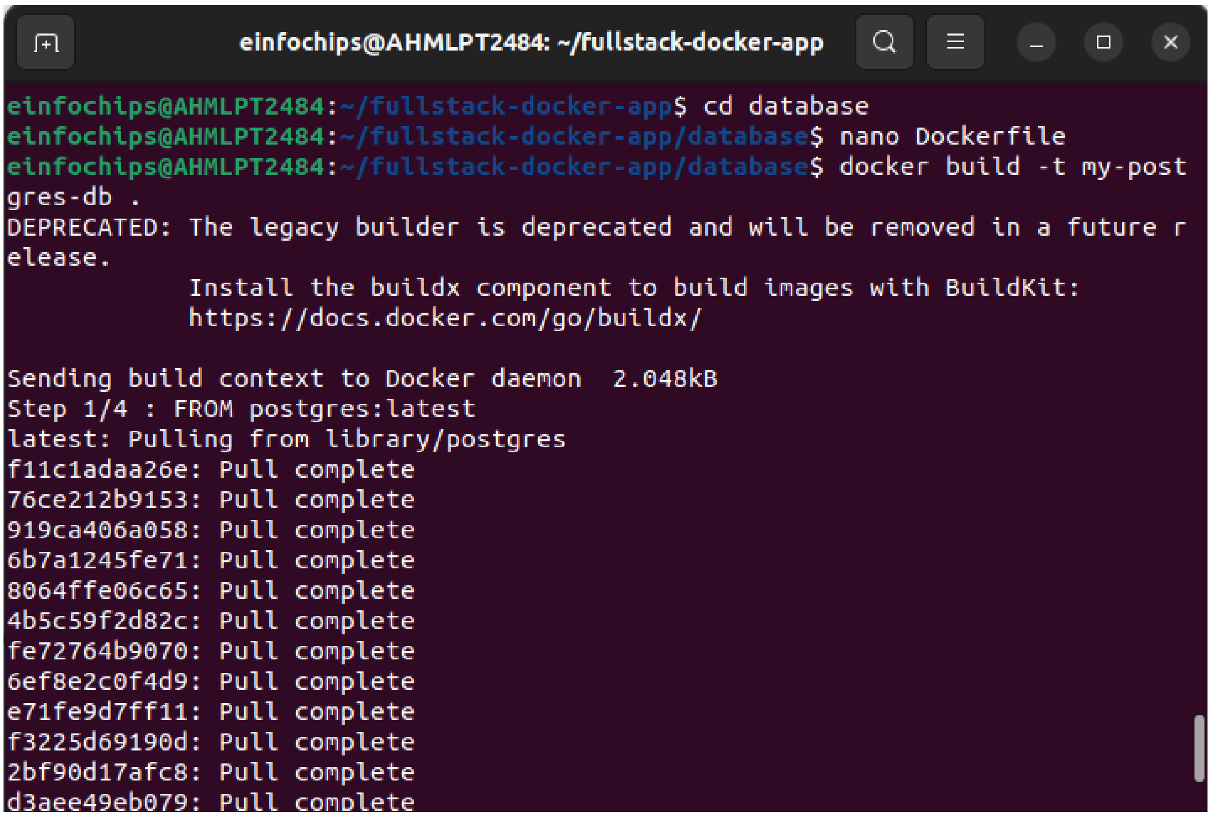
In the `database` directory, create a file named `Dockerfile` with the following content:

```
FROM postgres:latest
ENV POSTGRES_USER=user
ENV POSTGRES_PASSWORD=password
ENV POSTGRES_DB=mydatabase
```

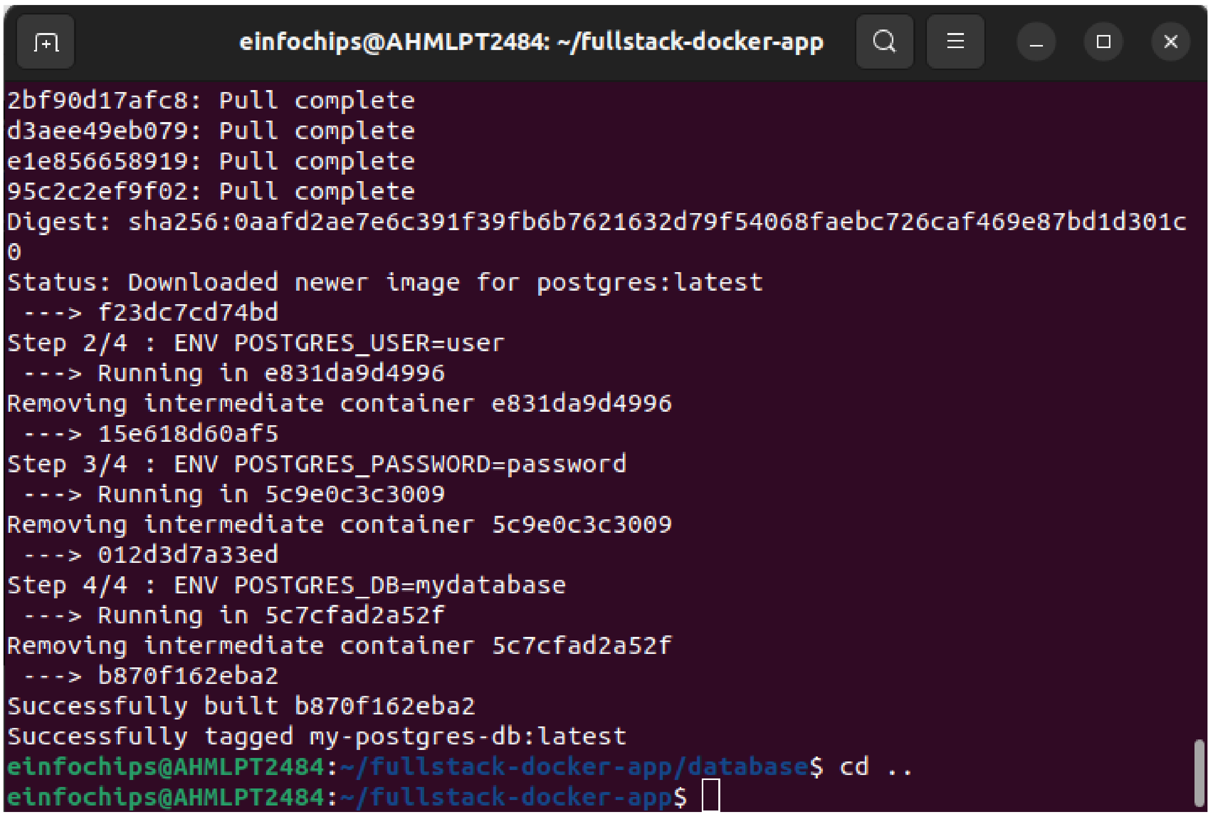
○

Build the PostgreSQL Image:

```
cd database
docker build -t my-postgres-db .
cd ..
```



```
einfochips@AHMLPT2484:~/fullstack-docker-app$ cd database
einfochips@AHMLPT2484:~/fullstack-docker-app/database$ nano Dockerfile
einfochips@AHMLPT2484:~/fullstack-docker-app/database$ docker build -t my-post
gres-db .
DEPRECATED: The legacy builder is deprecated and will be removed in a future r
elease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  2.048kB
Step 1/4 : FROM postgres:latest
latest: Pulling from library/postgres
f11c1adaa26e: Pull complete
76ce212b9153: Pull complete
919ca406a058: Pull complete
6b7a1245fe71: Pull complete
8064ffe06c65: Pull complete
4b5c59f2d82c: Pull complete
fe72764b9070: Pull complete
6ef8e2c0f4d9: Pull complete
e71fe9d7ff11: Pull complete
f3225d69190d: Pull complete
2bf90d17afc8: Pull complete
d3aee49eb079: Pull complete
```

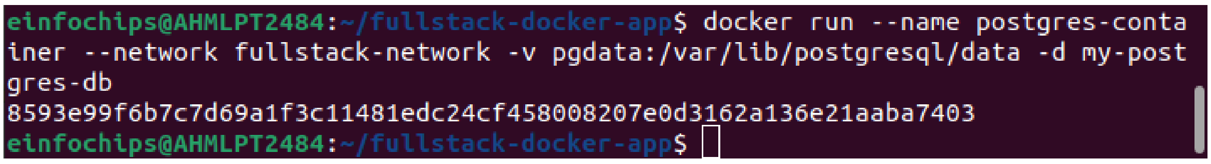


```
2bf90d17afc8: Pull complete
d3aee49eb079: Pull complete
e1e856658919: Pull complete
95c2c2ef9f02: Pull complete
Digest: sha256:0aafd2ae7e6c391f39fb6b7621632d79f54068faebc726caf469e87bd1d301c
0
Status: Downloaded newer image for postgres:latest
 ---> f23dc7cd74bd
Step 2/4 : ENV POSTGRES_USER=user
 ---> Running in e831da9d4996
Removing intermediate container e831da9d4996
 ---> 15e618d60af5
Step 3/4 : ENV POSTGRES_PASSWORD=password
 ---> Running in 5c9e0c3c3009
Removing intermediate container 5c9e0c3c3009
 ---> 012d3d7a33ed
Step 4/4 : ENV POSTGRES_DB=mydatabase
 ---> Running in 5c7cfad2a52f
Removing intermediate container 5c7cfad2a52f
 ---> b870f162eba2
Successfully built b870f162eba2
Successfully tagged my-postgres-db:latest
einfochips@AHMLPT2484:~/fullstack-docker-app/database$ cd ..
einfochips@AHMLPT2484:~/fullstack-docker-app$ 
```

2.

Run the PostgreSQL Container:

```
docker run --name postgres-container --network fullstack-network -v
pgdata:/var/lib/postgresql/data -d my-postgres-db
```



```
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker run --name postgres-conta
iner --network fullstack-network -v pgdata:/var/lib/postgresql/data -d my-post
gres-db
8593e99f6b7c7d69a1f3c11481edc24cf458008207e0d3162a136e21aaba7403
einfochips@AHMLPT2484:~/fullstack-docker-app$ 
```

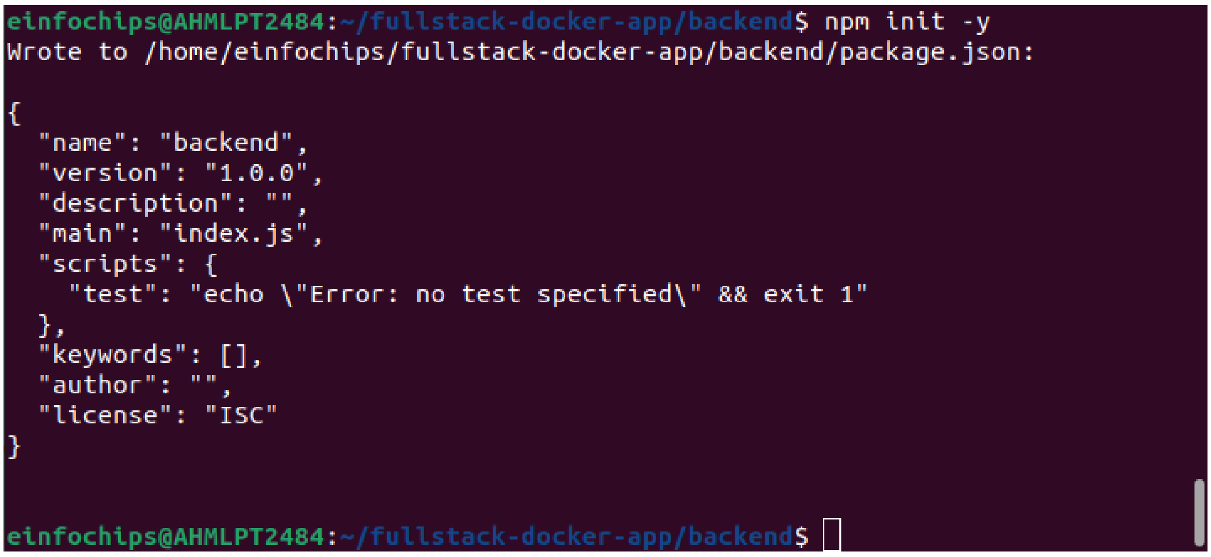
Part 3: Setting Up the Backend (Node.js with Express)

Objective: Create a Node.js application with Express and set it up with Docker.

Steps:

Initialize the Node.js Application:

```
cd backend
npm init -y
```

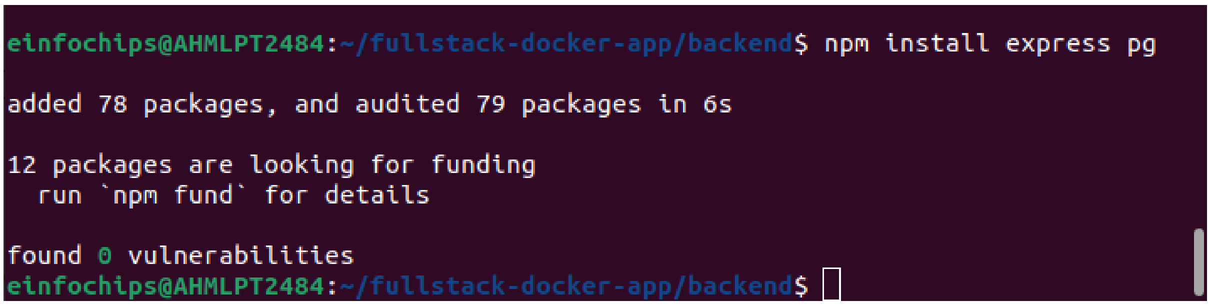


```
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ npm init -y
Wrote to /home/einfochips/fullstack-docker-app/backend/package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}


einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ 
```

1.

Install Express and pg (PostgreSQL client for Node.js):

```
npm install express pg
```



```
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ npm install express pg

added 78 packages, and audited 79 packages in 6s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ 
```

2.
3. **Create the Application Code:**

In the `backend` directory, create a file named `index.js` with the following content:

```
const express = require('express');
const { Pool } = require('pg');
const app = express();
const port = 3000;

const pool = new Pool({
    user: 'user',
    host: 'postgres-container',
    database: 'mydatabase',
    password: 'password',
    port: 5432,
});

app.get('/', (req, res) => {
    res.send('Hello from Node.js and Docker!');
});

app.get('/data', async (req, res) => {
    const client = await pool.connect();
    const result = await client.query('SELECT NOW()');
    client.release();
    res.send(result.rows);
});

app.listen(port, () => {
    console.log(`App running on http://localhost:${port}`);
});
```

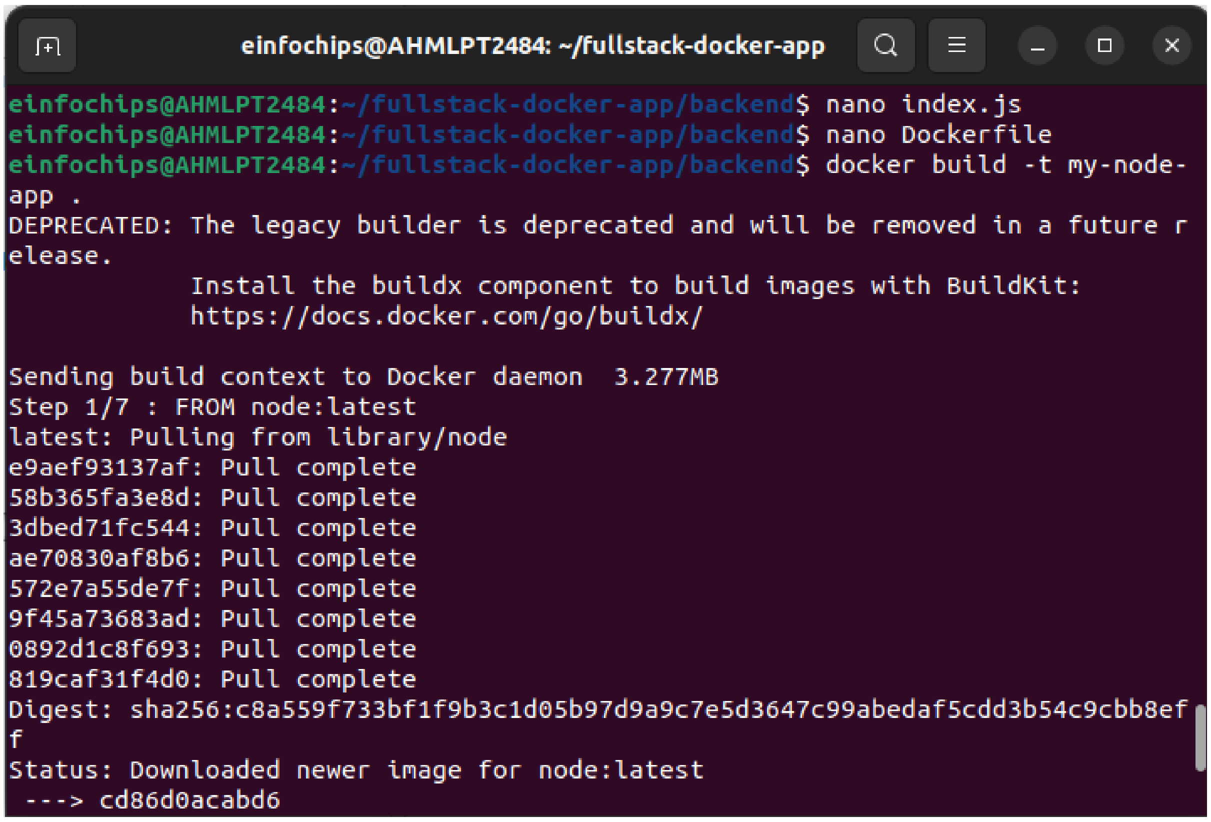
4. **Create a Dockerfile for the Backend:**

In the `backend` directory, create a file named `Dockerfile` with the following content:

```
FROM node:latest

WORKDIR /usr/src/app

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 3000
CMD ["node", "index.js"]
```

Build the Backend Image:

```
docker build -t my-node-app .
cd ..
```

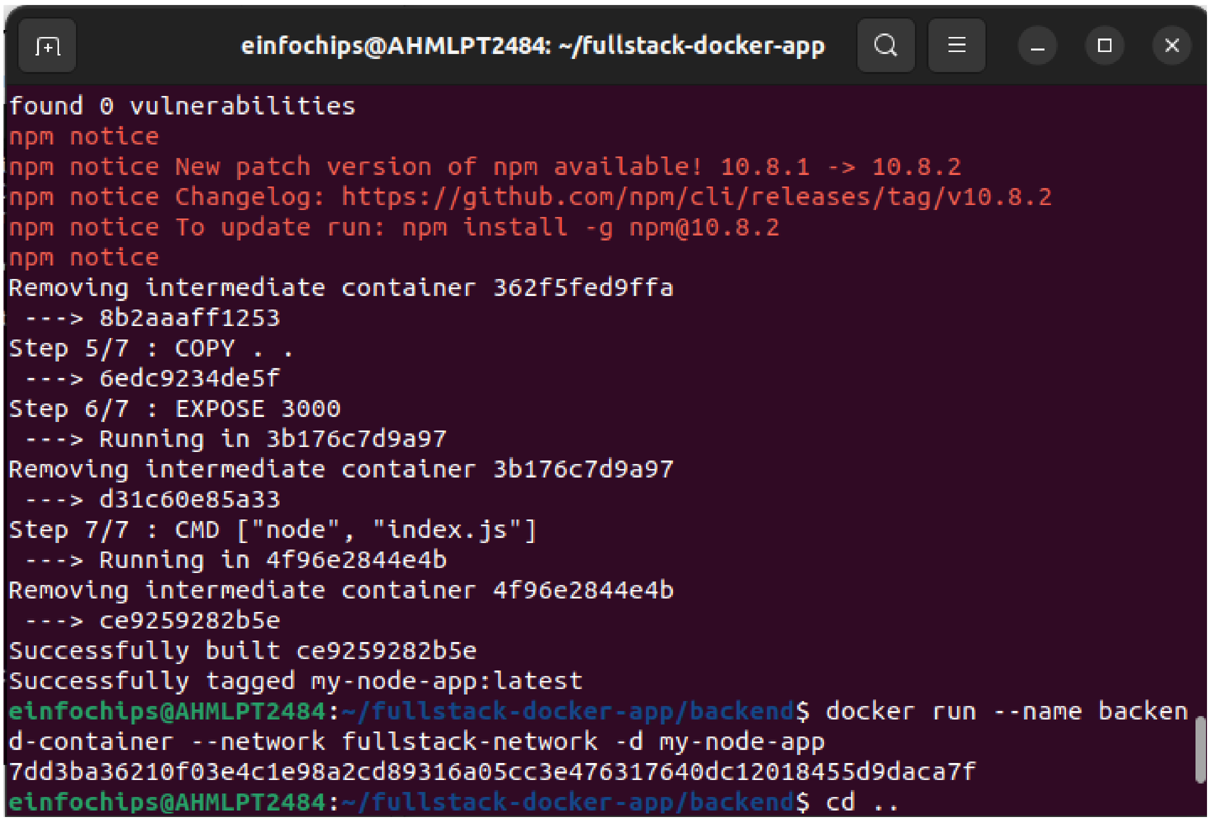


```
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ nano index.js
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ nano Dockerfile
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ docker build -t my-node-
app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future r
elease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.277MB
Step 1/7 : FROM node:latest
latest: Pulling from library/node
e9aef93137af: Pull complete
58b365fa3e8d: Pull complete
3dbed71fc544: Pull complete
ae70830af8b6: Pull complete
572e7a55de7f: Pull complete
9f45a73683ad: Pull complete
0892d1c8f693: Pull complete
819caf31f4d0: Pull complete
Digest: sha256:c8a559f733bf1f9b3c1d05b97d9a9c7e5d3647c99abedaf5cdd3b54c9cbb8ef
f
Status: Downloaded newer image for node:latest
 ---> cd86d0acabd6
```

5.

Run the Backend Container:

```
docker run --name backend-container --network fullstack-network -d
my-node-app
```



```
found 0 vulnerabilities
npm notice
npm notice New patch version of npm available! 10.8.1 -> 10.8.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.8.2
npm notice To update run: npm install -g npm@10.8.2
npm notice
Removing intermediate container 362f5fed9ffa
 ---> 8b2aaaff1253
Step 5/7 : COPY . .
 ---> 6edc9234de5f
Step 6/7 : EXPOSE 3000
 ---> Running in 3b176c7d9a97
Removing intermediate container 3b176c7d9a97
 ---> d31c60e85a33
Step 7/7 : CMD ["node", "index.js"]
 ---> Running in 4f96e2844e4b
Removing intermediate container 4f96e2844e4b
 ---> ce9259282b5e
Successfully built ce9259282b5e
Successfully tagged my-node-app:latest
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ docker run --name backen
d-container --network fullstack-network -d my-node-app
7dd3ba36210f03e4c1e98a2cd89316a05cc3e476317640dc12018455d9daca7f
einfochips@AHMLPT2484:~/fullstack-docker-app/backend$ cd ..
```

Part 4: Setting Up the Frontend (Nginx)

Objective: Create a simple static front-end and set it up with Docker.

Steps:

1. **Create a Simple HTML Page:**

In the `frontend` directory, create a file named `index.html` with the following content:

```
<!DOCTYPE html>
<html>
<body>
    <h1>Hello from Nginx and Docker!</h1>
    <p>This is a simple static front-end served by Nginx.</p>
</body>
</html>
```

○

2. **Create a Dockerfile for the Frontend:**

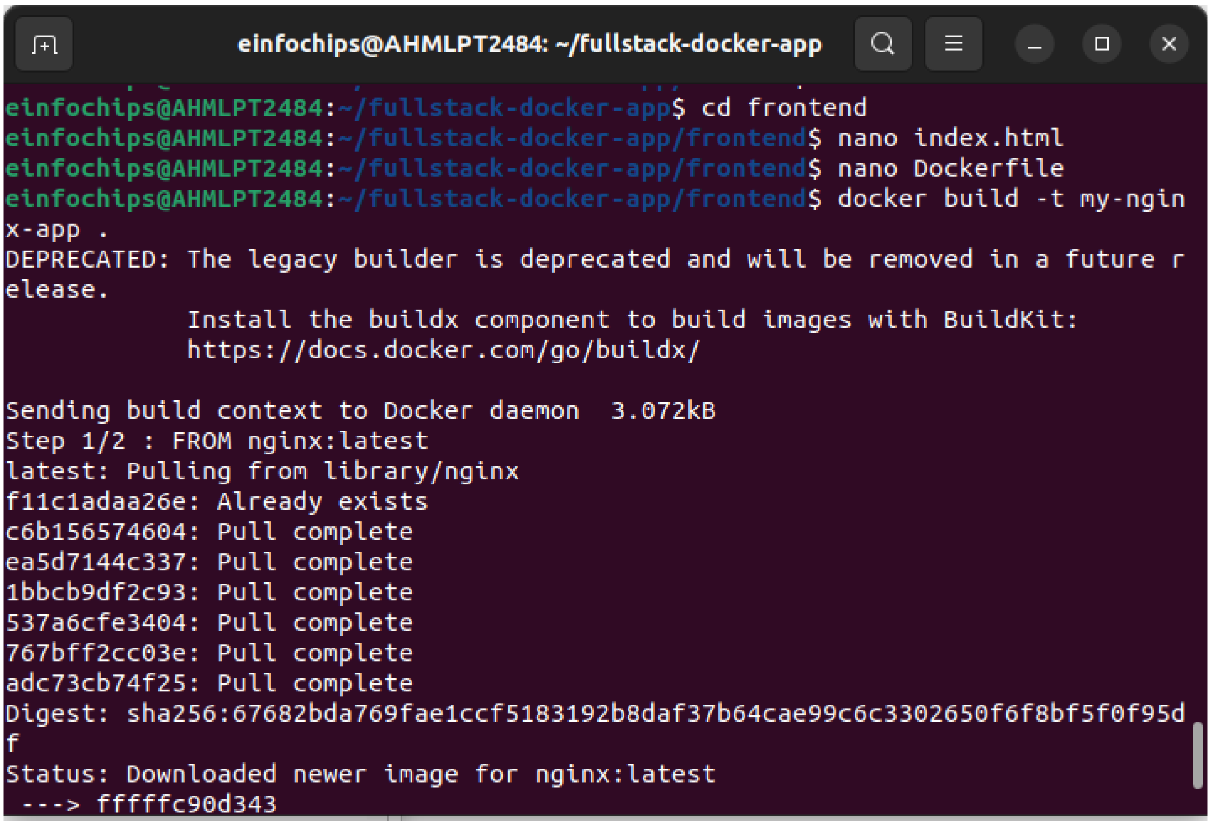
In the `frontend` directory, create a file named `Dockerfile` with the following content:

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

○

Build the Frontend Image:

```
cd frontend
docker build -t my-nginx-app .
cd ..
```

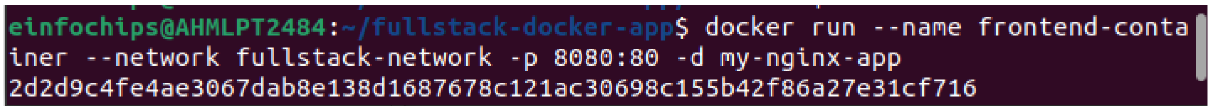


```
einfochips@AHMLPT2484:~/fullstack-docker-app$ cd frontend
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ nano index.html
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ nano Dockerfile
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ docker build -t my-ngin
x-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future r
elease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx:latest
latest: Pulling from library/nginx
f11c1adaa26e: Already exists
c6b156574604: Pull complete
ea5d7144c337: Pull complete
1bbcb9df2c93: Pull complete
537a6cfe3404: Pull complete
767bff2cc03e: Pull complete
adc73cb74f25: Pull complete
Digest: sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95d
f
Status: Downloaded newer image for nginx:latest
 ---> fffffc90d343
```

3.

Run the Frontend Container:

```
docker run --name frontend-container --network fullstack-network -p
8080:80 -d my-nginx-app
```



```
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker run --name frontend-conta
iner --network fullstack-network -p 8080:80 -d my-nginx-app
2d2d9c4fe4ae3067dab8e138d1687678c121ac30698c155b42f86a27e31cf716
```

Part 5: Connecting the Backend and Database

Objective: Ensure the backend can communicate with the database and handle data requests.

Steps:

1. **Update Backend Code to Fetch Data from PostgreSQL:**
    - Ensure that the `index.js` code in the backend handles `/data` endpoint correctly as written above.
2. **Verify Backend Communication:**

Access the backend container:

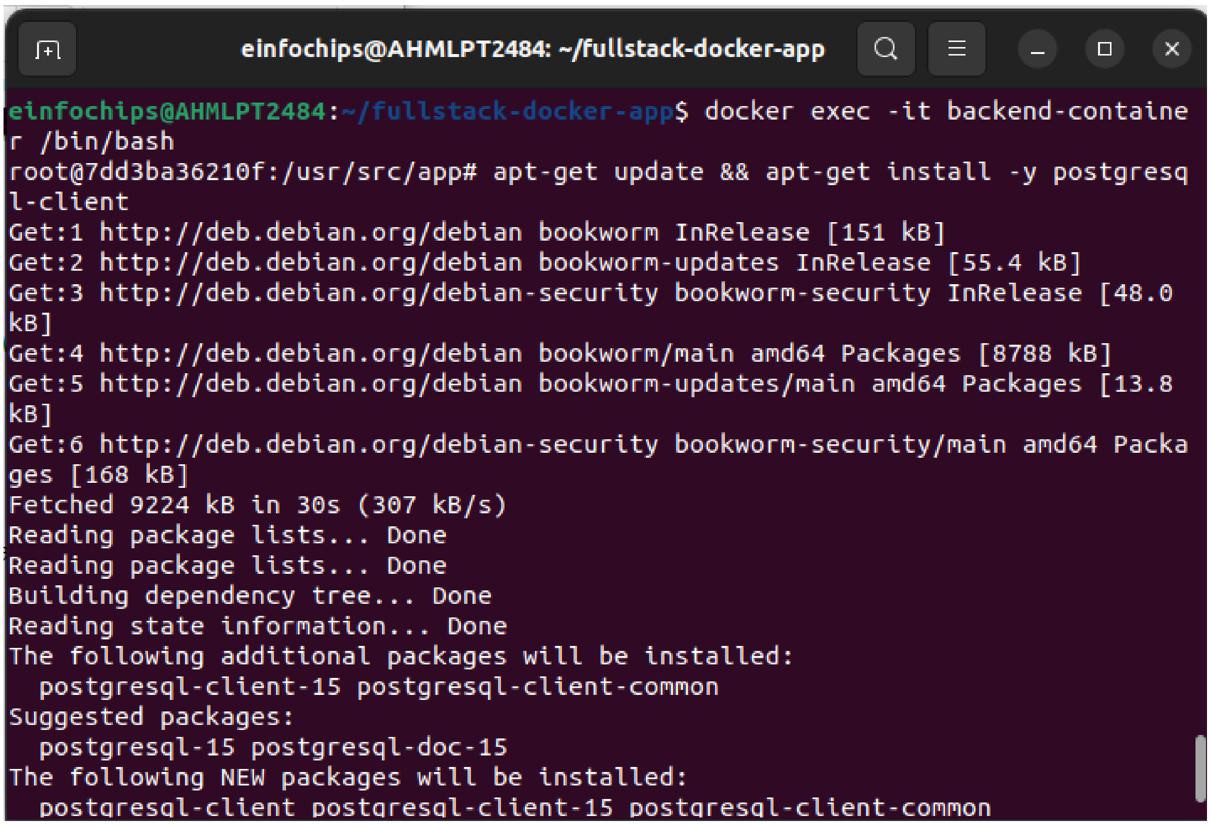
```
docker exec -it backend-container /bin/bash
```

Test the connection to the database using `psql`:

```
apt-get update && apt-get install -y postgresql-client
psql -h postgres-container -U user -d mydatabase -c "SELECT NOW();"
```

Exit the container:

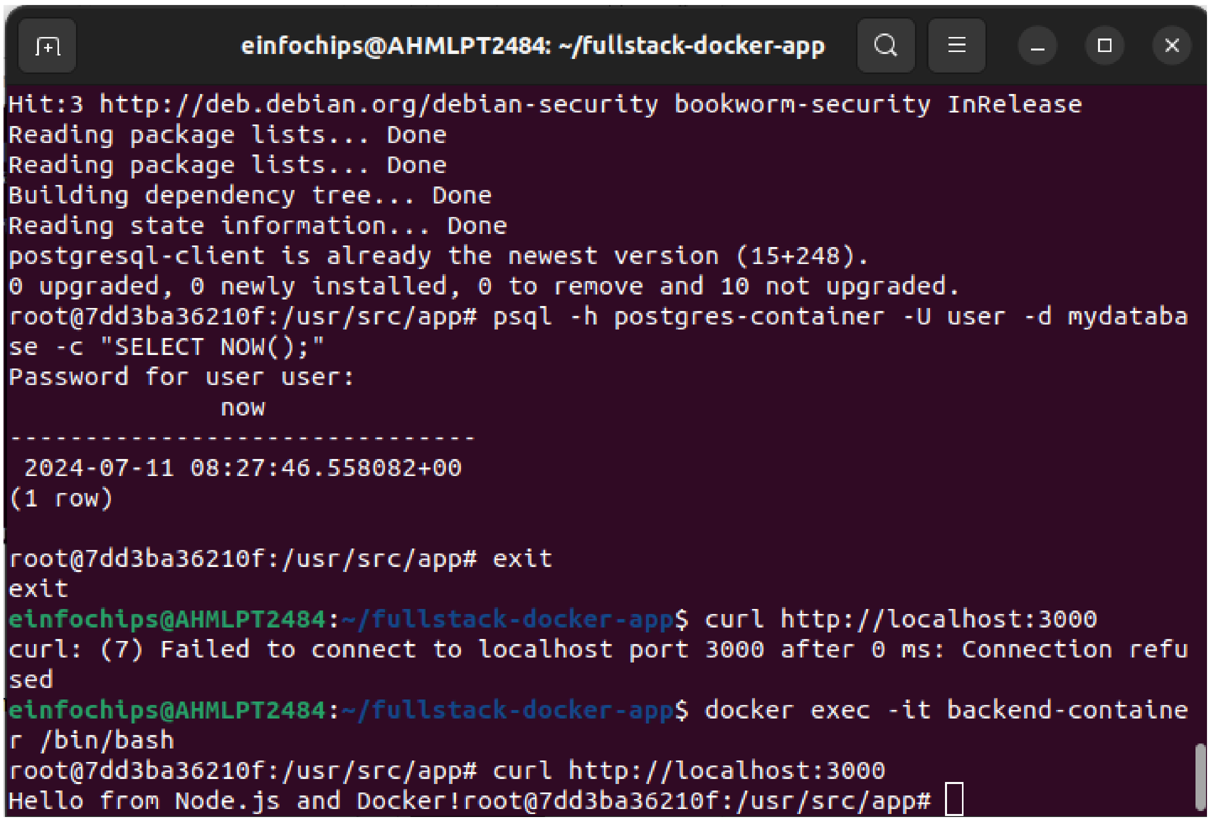
```
exit
```



```
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker exec -it backend-containe
r /bin/bash
root@7dd3ba36210f:/usr/src/app# apt-get update && apt-get install -y postgresq
l-client
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0
kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8788 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [13.8
kB]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packa
ges [168 kB]
Fetched 9224 kB in 30s (307 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  postgresql-client-15 postgresql-client-common
Suggested packages:
  postgresql-15 postgresql-doc-15
The following NEW packages will be installed:
  postgresql-client postgresql-client-15 postgresql-client-common
```

3. **Test the Backend API:**
    - Visit `http://localhost:3000` to see the basic message.

- Visit `http://localhost:3000/data` to see the current date and time fetched from PostgreSQL.


einfochips@AHMLPT2484: ~/fullstack-docker-app

```
Hit:3 http://deb.debian.org/debian-security bookworm-security InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
postgresql-client is already the newest version (15+248).
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
root@7dd3ba36210f:/usr/src/app# psql -h postgres-container -U user -d mydataba
se -c "SELECT NOW();"
Password for user user:
              now
-------------------------------
 2024-07-11 08:27:46.558082+00
(1 row)

root@7dd3ba36210f:/usr/src/app# exit
exit
einfochips@AHMLPT2484:~/fullstack-docker-app$ curl http://localhost:3000
curl: (7) Failed to connect to localhost port 3000 after 0 ms: Connection refu
sed
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker exec -it backend-containe
r /bin/bash
root@7dd3ba36210f:/usr/src/app# curl http://localhost:3080
Hello from Node.js and Docker!root@7dd3ba36210f:/usr/src/app# 
```

Part 6: Final Integration and Testing

Objective: Ensure all components are working together and verify the full-stack application.

Steps:

1. **Access the Frontend:**
   - Visit `http://localhost:8080` in your browser. You should see the Nginx welcome page with the custom HTML.
2. **Verify Full Integration:**

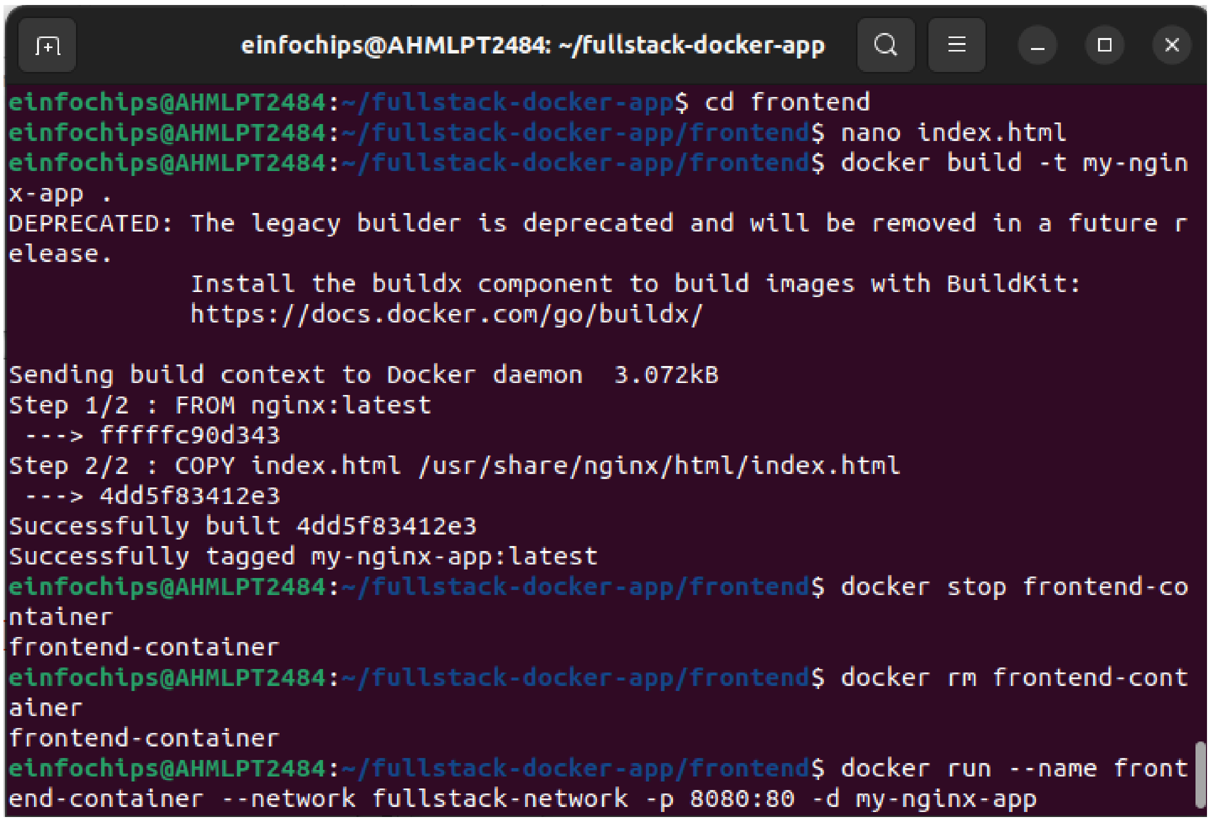
Update the `index.html` to include a link to the backend:

```
<!DOCTYPE html>
<html>
<body>
    <h1>Hello from Nginx and Docker!</h1>
    <p>This is a simple static front-end served by Nginx.</p>
    <a href="http://localhost:3000/data">Fetch Data from Backend</a>
</body>
</html>
```

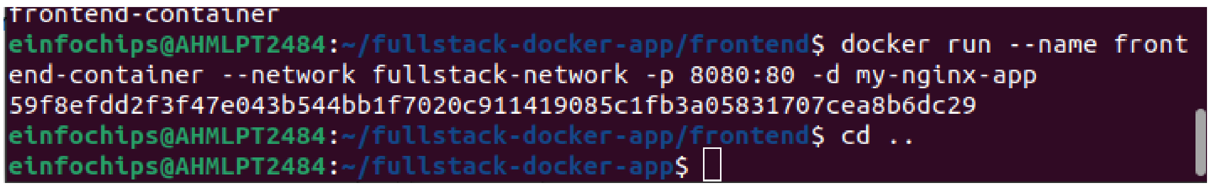
○

Rebuild and Run the Updated Frontend Container:

```
cd frontend
docker build -t my-nginx-app .
docker stop frontend-container
docker rm frontend-container
docker run --name frontend-container --network fullstack-network -p
8080:80 -d my-nginx-app
cd ..
```

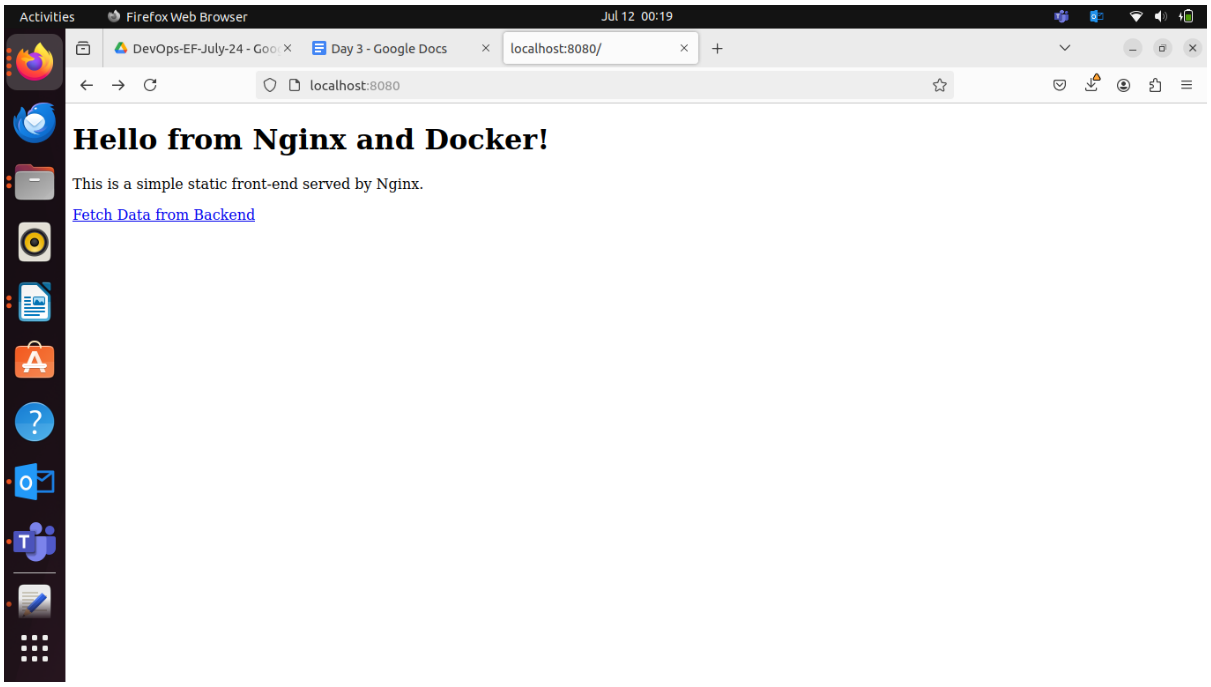


```
einfochips@AHMLPT2484:~/fullstack-docker-app$ cd frontend
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ nano index.html
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ docker build -t my-ngin
x-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future r
elease.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx:latest
 ---> fffffc90d343
Step 2/2 : COPY index.html /usr/share/nginx/html/index.html
 ---> 4dd5f83412e3
Successfully built 4dd5f83412e3
Successfully tagged my-nginx-app:latest
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ docker stop frontend-co
ntainer
frontend-container
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ docker rm frontend-cont
ainer
frontend-container
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ docker run --name front
end-container --network fullstack-network -p 8080:80 -d my-nginx-app
```



```
frontend-container
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ docker run --name front
end-container --network fullstack-network -p 8080:80 -d my-nginx-app
59f8efdd2f3f47e043b544bb1f7020c911419085c1fb3a05831707cea8b6dc29
einfochips@AHMLPT2484:~/fullstack-docker-app/frontend$ cd ..
einfochips@AHMLPT2484:~/fullstack-docker-app$ 
```

3. **Final Verification:**
    - Visit `http://localhost:8080` and click the link to fetch data from the backend.



Hello from Nginx and Docker!

This is a simple static front-end served by Nginx.

Fetch Data from Backend

Part 7: Cleaning Up

Objective: Remove all created containers, images, networks, and volumes to clean up your environment.

Steps:

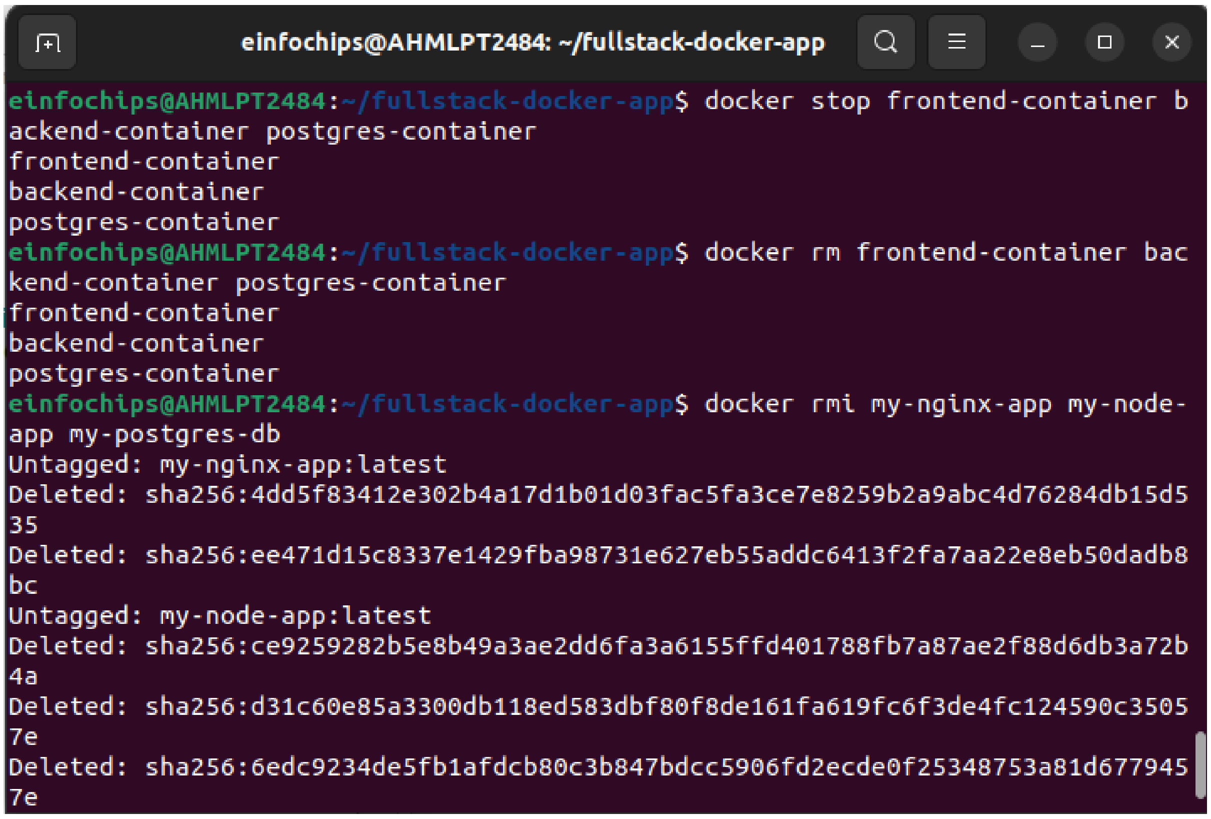
Stop and Remove the Containers:

```
docker stop frontend-container backend-container postgres-container
docker rm frontend-container backend-container postgres-container
```

1.

Remove the Images:

```
docker rmi my-nginx-app my-node-app my-postgres-db
```

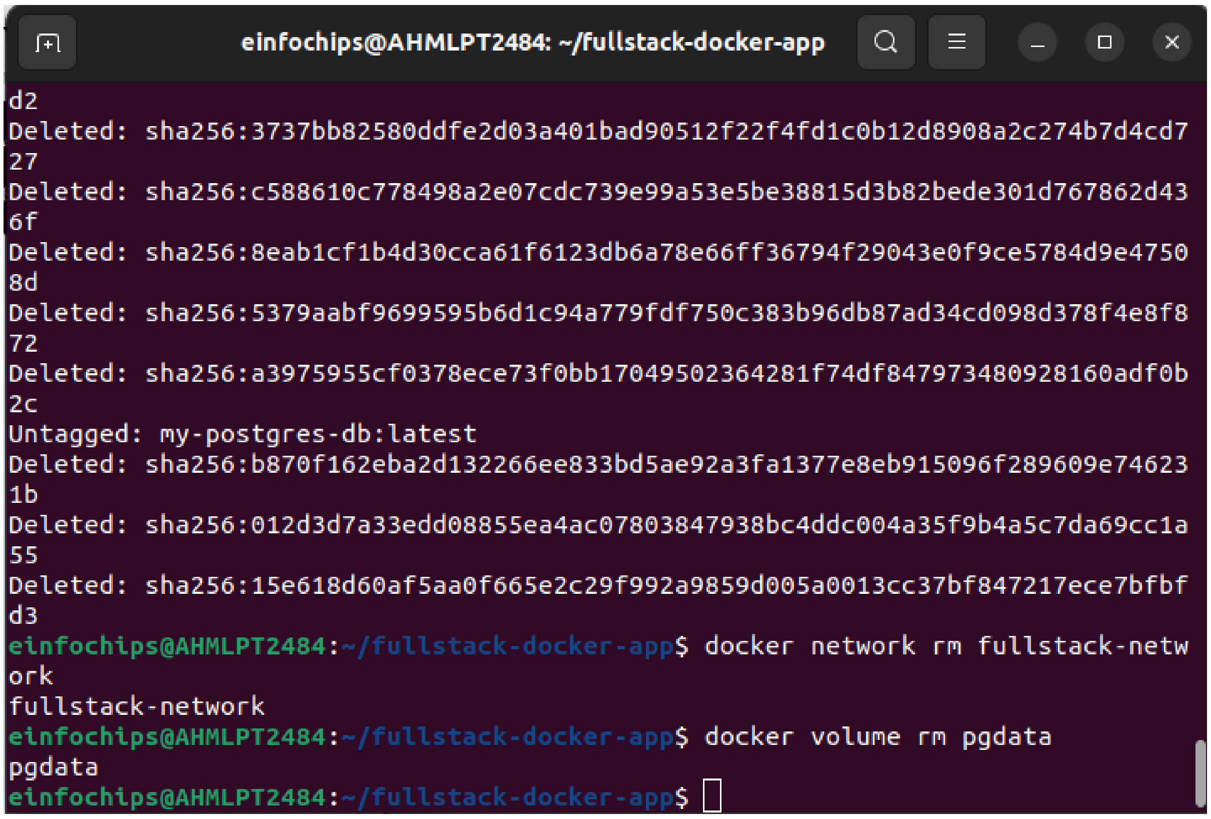


```
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker stop frontend-container b
ackend-container postgres-container
frontend-container
backend-container
postgres-container
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker rm frontend-container bac
kend-container postgres-container
frontend-container
backend-container
postgres-container
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker rmi my-nginx-app my-node-
app my-postgres-db
Untagged: my-nginx-app:latest
Deleted: sha256:4dd5f83412e302b4a17d1b01d03fac5fa3ce7e8259b2a9abc4d76284db15d5
35
Deleted: sha256:ee471d15c8337e1429fba98731e627eb55addc6413f2fa7aa22e8eb50dadb8
bc
Untagged: my-node-app:latest
Deleted: sha256:ce9259282b5e8b49a3ae2dd6fa3a6155ffd401788fb7a87ae2f88d6db3a72b
4a
Deleted: sha256:d31c60e85a3300db118ed583dbf80f8de161fa619fc6f3de4fc124590c3505
7e
Deleted: sha256:6edc9234de5fb1afdcb80c3b847bdcc5906fd2ecde0f25348753a81d677945
7e
```

2.

Remove the Network and Volume:

```
docker network rm fullstack-network
docker volume rm pgdata
```



```
d2
Deleted: sha256:3737bb82580ddfe2d03a401bad90512f22f4fd1c0b12d8908a2c274b7d4cd7
27
Deleted: sha256:c588610c778498a2e07cdc739e99a53e5be38815d3b82bede301d767862d43
6f
Deleted: sha256:8eab1cf1b4d30cca61f6123db6a78e66ff36794f29043e0f9ce5784d9e4750
8d
Deleted: sha256:5379aabf9699595b6d1c94a779fdf750c383b96db87ad34cd098d378f4e8f8
72
Deleted: sha256:a3975955cf0378ece73f0bb17049502364281f74df847973480928160adf0b
2c
Untagged: my-postgres-db:latest
Deleted: sha256:b870f162eba2d132266ee833bd5ae92a3fa1377e8eb915096f289609e74623
1b
Deleted: sha256:012d3d7a33edd08855ea4ac07803847938bc4ddc004a35f9b4a5c7da69cc1a
55
Deleted: sha256:15e618d60af5aa0f665e2c29f992a9859d005a0013cc37bf847217ece7bfbf
d3
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker network rm fullstack-netw
ork
fullstack-network
einfochips@AHMLPT2484:~/fullstack-docker-app$ docker volume rm pgdata
pgdata
einfochips@AHMLPT2484:~/fullstack-docker-app$
```