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.

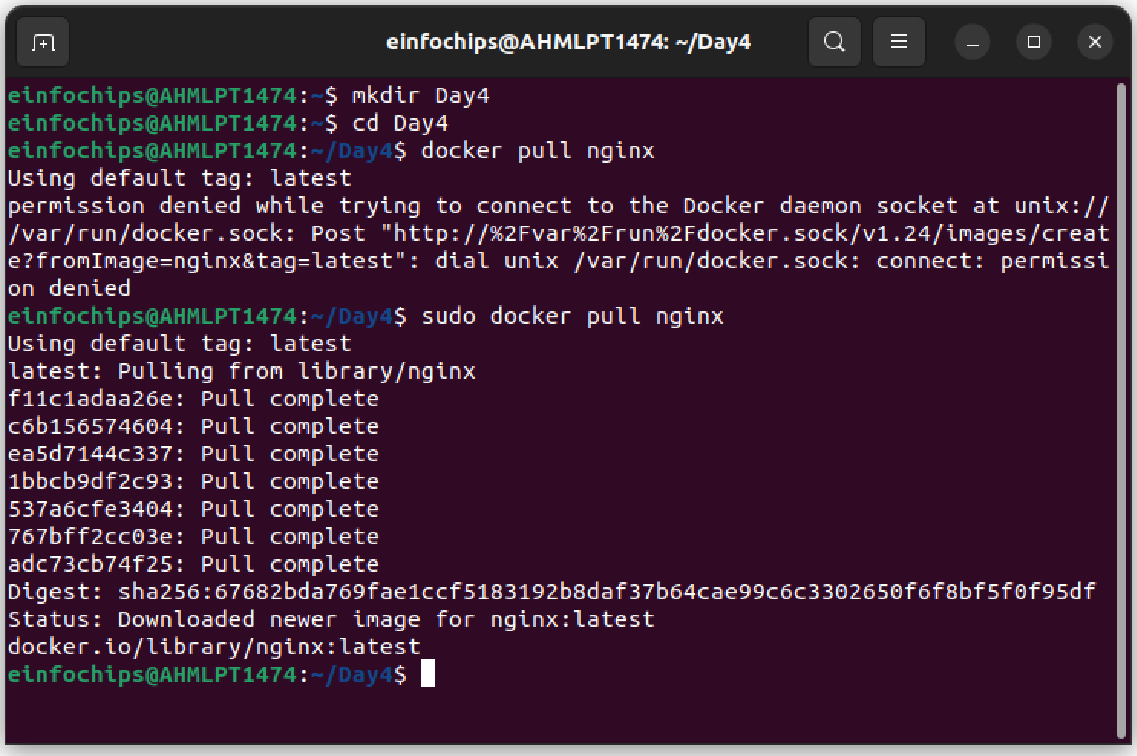
Part 1: Creating a Container from a Pulled Image

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

Steps:

Pull the Nginx Image:

```
docker pull nginx
```



```
einfochips@AHMLPT1474:~$ mkdir Day4
einfochips@AHMLPT1474:~$ cd Day4
einfochips@AHMLPT1474:~/Day4$ docker pull nginx
Using default tag: latest
permission denied while trying to connect to the Docker daemon socket at unix://
/var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/creat
e?fromImage=nginx&tag=latest": dial unix /var/run/docker.sock: connect: permissi
on denied
einfochips@AHMLPT1474:~/Day4$ 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: sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
einfochips@AHMLPT1474:~/Day4$
```

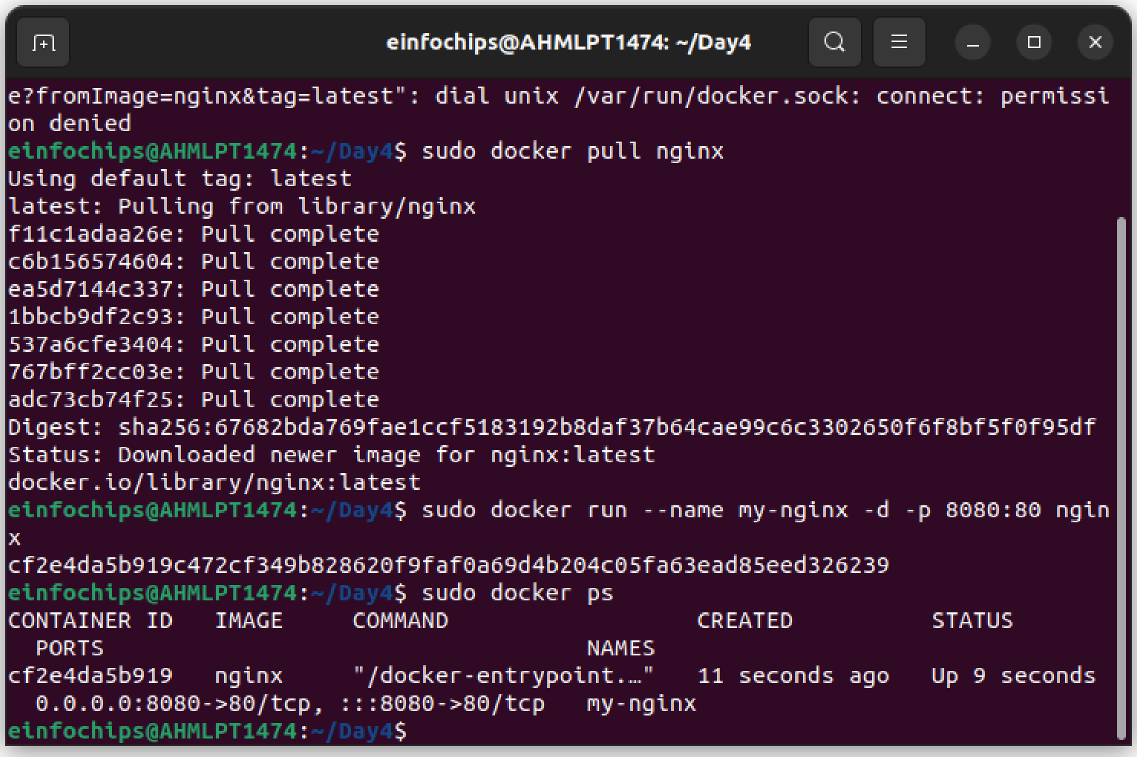
Run the Nginx Container:

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

1.
   - `--name my-nginx`: Assigns a name to the container.
   - `-d`: Runs the container in detached mode.
   - `-p 8080:80`: Maps port 8080 on your host to port 80 in the container.

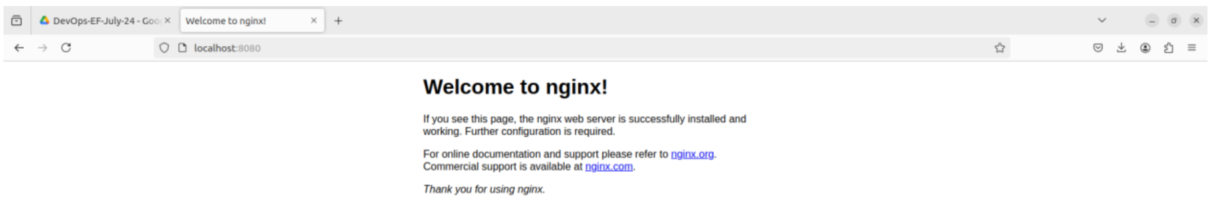
Verify the Container is Running:

docker ps



```
e?fromImage=nginx&tag=latest": dial unix /var/run/docker.sock: connect: permissi
on denied
einfochips@AHMLPT1474:~/Day4$ 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: sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
einfochips@AHMLPT1474:~/Day4$ sudo docker run --name my-nginx -d -p 8080:80 ngin
x
cf2e4da5b919c472cf349b828620f9faf0a69d4b204c05fa63ead85eed326239
einfochips@AHMLPT1474:~/Day4$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS
  PORTS                                   NAMES
cf2e4da5b919   nginx     "/docker-entrypoint.…"   11 seconds ago   Up 9 seconds
  0.0.0.0:8080->80/tcp, :::8080->80/tcp   my-nginx
einfochips@AHMLPT1474:~/Day4$
```

2.
   - Visit http://localhost:8080 in your browser. You should see 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.

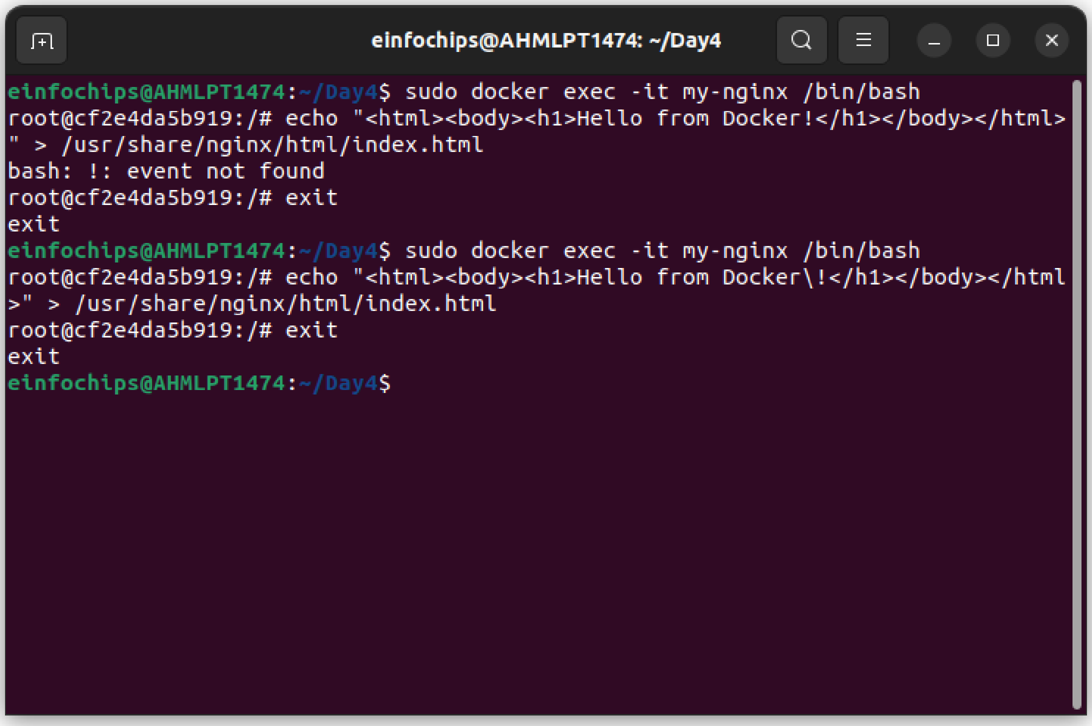
Create a Custom HTML Page:

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

2.

Exit the Container:

```
exit
```



```
einfochips@AHMLPT1474:~/Day4$ sudo docker exec -it my-nginx /bin/bash
root@cf2e4da5b919:/# echo "<html><body><h1>Hello from Docker!</h1></body></html>
" > /usr/share/nginx/html/index.html
bash: !: event not found
root@cf2e4da5b919:/# exit
exit
einfochips@AHMLPT1474:~/Day4$ sudo docker exec -it my-nginx /bin/bash
root@cf2e4da5b919:/# echo "<html><body><h1>Hello from Docker\!</h1></body></html
>" > /usr/share/nginx/html/index.html
root@cf2e4da5b919:/# exit
exit
einfochips@AHMLPT1474:~/Day4$
```

3.

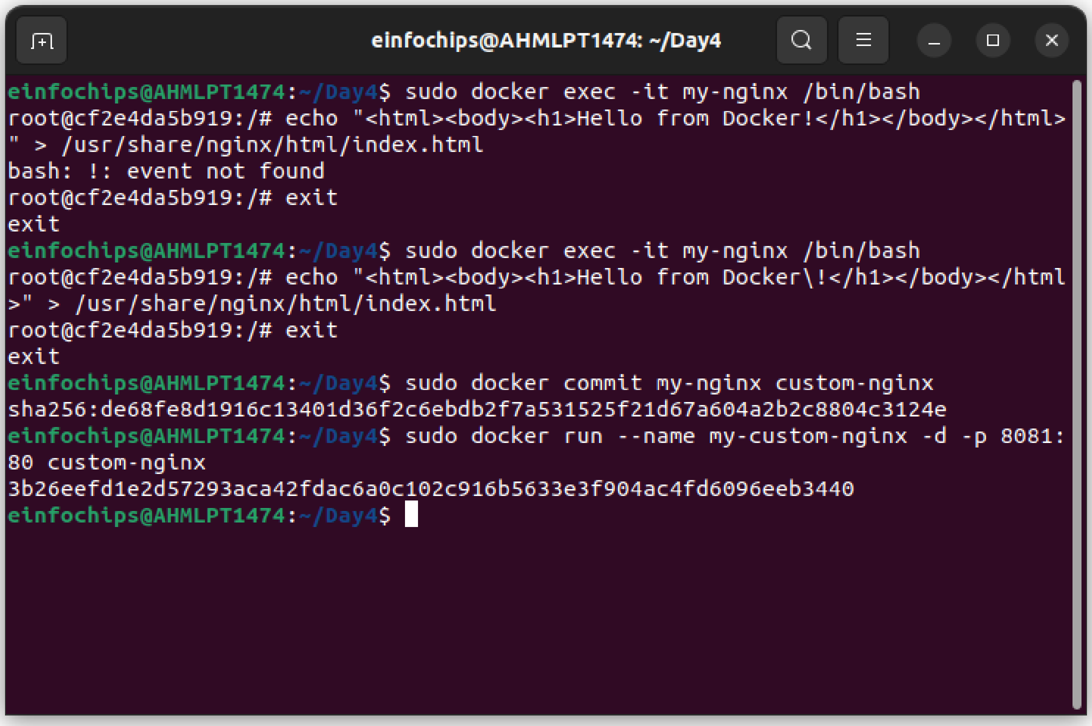
Commit the Changes to Create a New Image:

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

4.

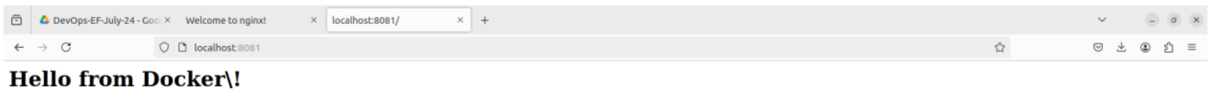
Run a Container from the New Image:

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



```
einfochips@AHMLPT1474:~/Day4$ sudo docker exec -it my-nginx /bin/bash
root@cf2e4da5b919:/# echo "<html><body><h1>Hello from Docker!</h1></body></html>
" > /usr/share/nginx/html/index.html
bash: !: event not found
root@cf2e4da5b919:/# exit
exit
einfochips@AHMLPT1474:~/Day4$ sudo docker exec -it my-nginx /bin/bash
root@cf2e4da5b919:/# echo "<html><body><h1>Hello from Docker\!</h1></body></html
>" > /usr/share/nginx/html/index.html
root@cf2e4da5b919:/# exit
exit
einfochips@AHMLPT1474:~/Day4$ sudo docker commit my-nginx custom-nginx
sha256:de68fe8d1916c13401d36f2c6ebdb2f7a531525f21d67a604a2b2c8804c3124e
einfochips@AHMLPT1474:~/Day4$ sudo docker run --name my-custom-nginx -d -p 8081:
80 custom-nginx
3b26eefd1e2d57293aca42fdac6a0c102c916b5633e3f904ac4fd6096eeb3440
einfochips@AHMLPT1474:~/Day4$ 
```

5.
6. **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:

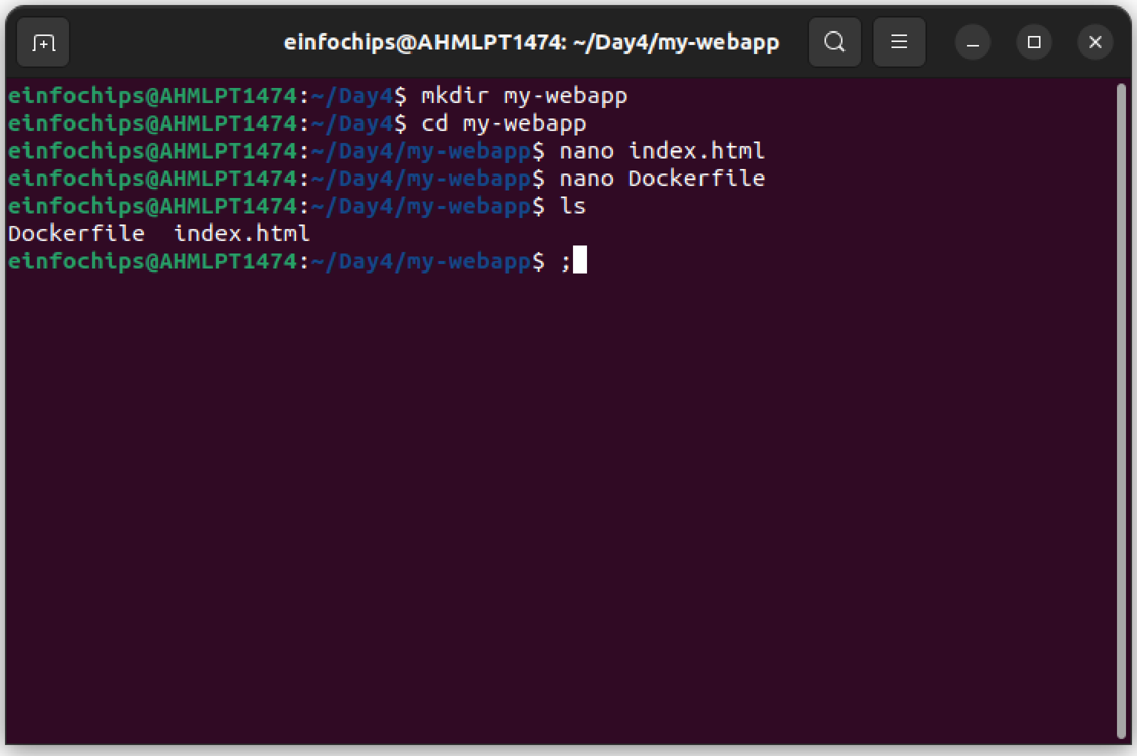
```
<!DOCTYPE html>
<html>
<body>
   <h1>Hello from My Web App!</h1>
</body>
</html>
```

   -
   - 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@AHMLPT1474:~/Day4$ mkdir my-webapp
einfochips@AHMLPT1474:~/Day4$ cd my-webapp
einfochips@AHMLPT1474:~/Day4/my-webapp$ nano index.html
einfochips@AHMLPT1474:~/Day4/my-webapp$ nano Dockerfile
einfochips@AHMLPT1474:~/Day4/my-webapp$ ls
Dockerfile  index.html
einfochips@AHMLPT1474:~/Day4/my-webapp$ ;
```

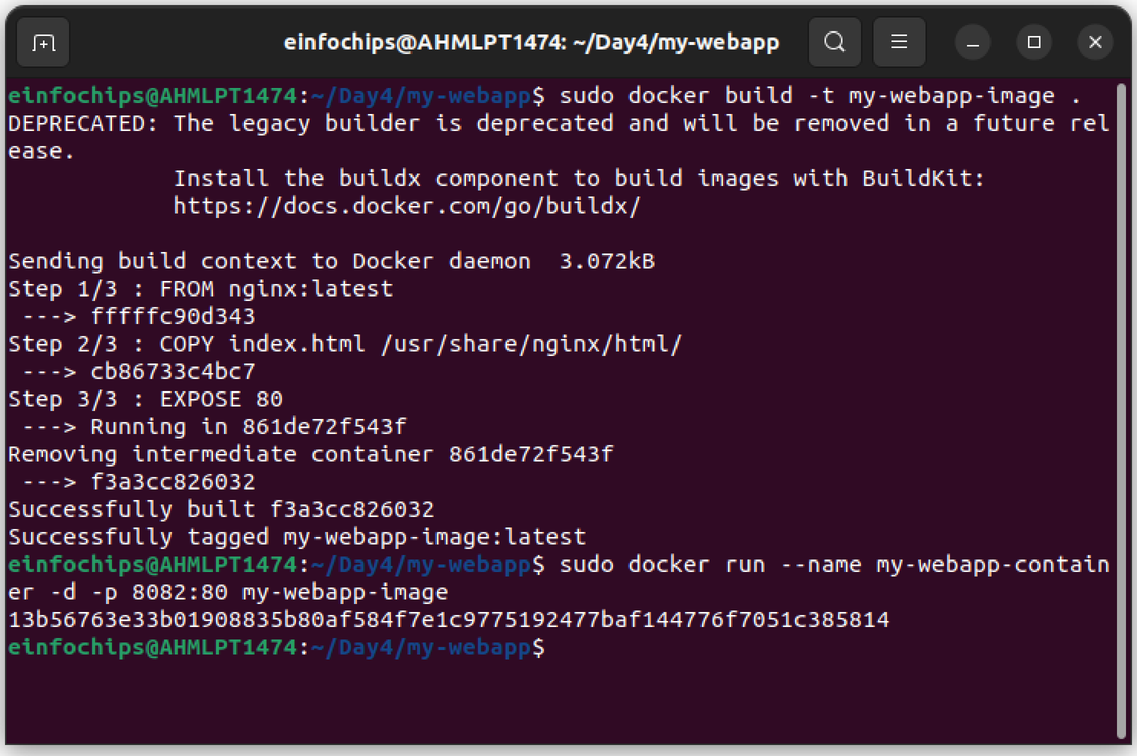
Build the Docker Image:

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

4.

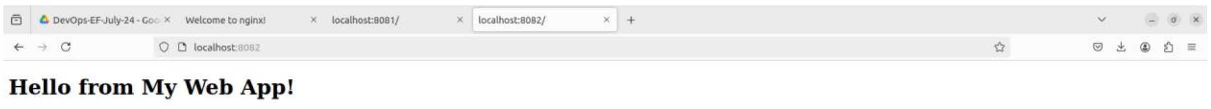
Run a Container from the Built Image:

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



```
einfochips@AHMLPT1474:~/Day4/my-webapp$ sudo docker build -t my-webapp-image .
DEPRECATED: The legacy builder is deprecated and will be removed in a future rel
ease.
            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/
 ---> cb86733c4bc7
Step 3/3 : EXPOSE 80
 ---> Running in 861de72f543f
Removing intermediate container 861de72f543f
 ---> f3a3cc826032
Successfully built f3a3cc826032
Successfully tagged my-webapp-image:latest
einfochips@AHMLPT1474:~/Day4/my-webapp$ sudo docker run --name my-webapp-contain
er -d -p 8082:80 my-webapp-image
13b56763e33b01908835b80af584f7e1c9775192477baf144776f7051c385814
einfochips@AHMLPT1474:~/Day4/my-webapp$
```

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



Hello from My Web App!

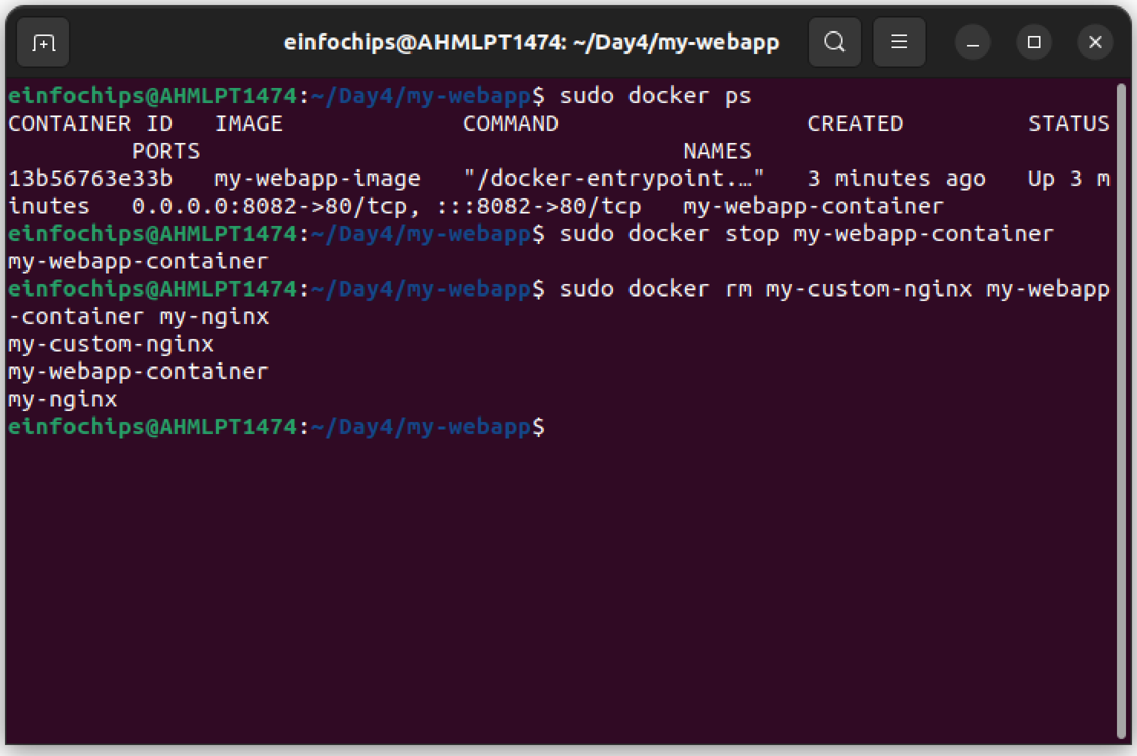
Part 4: Cleaning Up

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
```

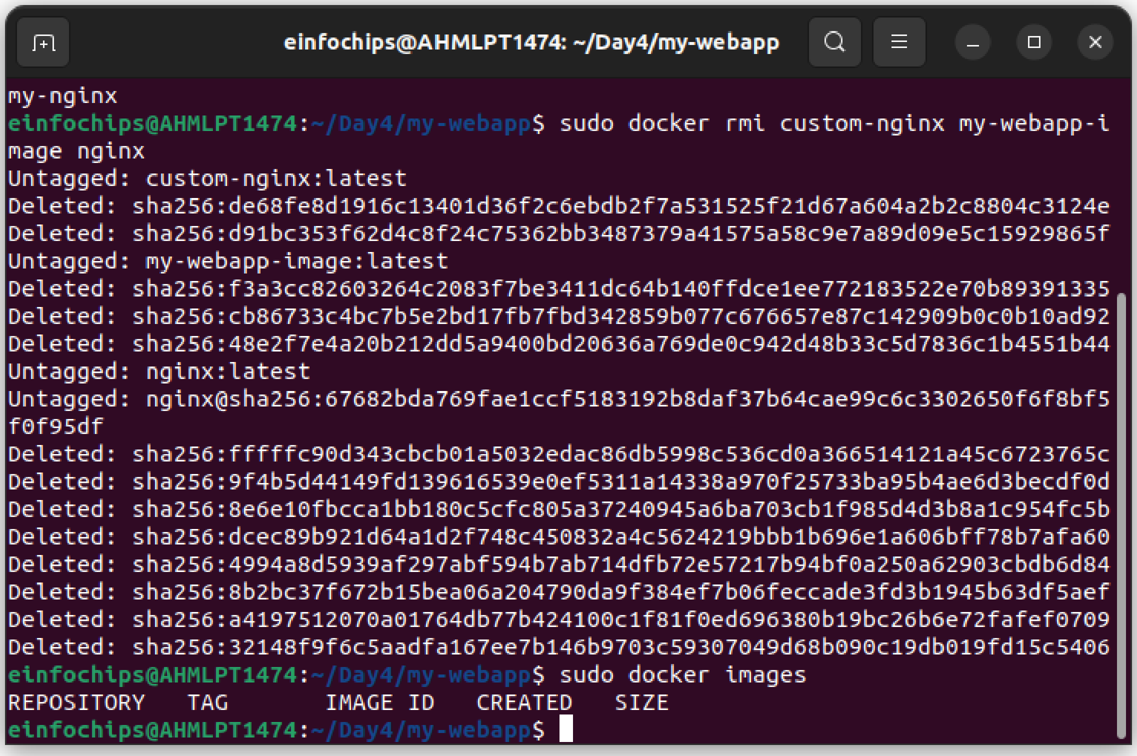


einfochips@AHMLPT1474: ~/Day4/my-webapp

```
einfochips@AHMLPT1474:~/Day4/my-webapp$ sudo docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS
         PORTS                                   NAMES
13b56763e33b   my-webapp-image    "/docker-entrypoint.…"   3 minutes ago    Up 3 m
inutes   0.0.0.0:8082->80/tcp, :::8082->80/tcp   my-webapp-container
einfochips@AHMLPT1474:~/Day4/my-webapp$ sudo docker stop my-webapp-container
my-webapp-container
einfochips@AHMLPT1474:~/Day4/my-webapp$ sudo docker rm my-custom-nginx my-webapp
-container my-nginx
my-custom-nginx
my-webapp-container
my-nginx
einfochips@AHMLPT1474:~/Day4/my-webapp$
```

1. **Remove the Images:**

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



```
einfochips@AHMLPT1474: ~/Day4/my-webapp
my-nginx
einfochips@AHMLPT1474:~/Day4/my-webapp$ sudo docker rmi custom-nginx my-webapp-i
mage nginx
Untagged: custom-nginx:latest
Deleted: sha256:de68fe8d1916c13401d36f2c6ebdb2f7a531525f21d67a604a2b2c8804c3124e
Deleted: sha256:d91bc353f62d4c8f24c75362bb3487379a41575a58c9e7a89d09e5c15929865f
Untagged: my-webapp-image:latest
Deleted: sha256:f3a3cc82603264c2083f7be3411dc64b140ffdce1ee772183522e70b89391335
Deleted: sha256:cb86733c4bc7b5e2bd17fb7fbd342859b077c676657e87c142909b0c0b10ad92
Deleted: sha256:48e2f7e4a20b212dd5a9400bd20636a769de0c942d48b33c5d7836c1b4551b44
Untagged: nginx:latest
Untagged: nginx@sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5
f0f95df
Deleted: sha256:fffffc90d343cbcb01a5032edac86db5998c536cd0a366514121a45c6723765c
Deleted: sha256:9f4b5d44149fd139616539e0ef5311a14338a970f25733ba95b4ae6d3becdf0d
Deleted: sha256:8e6e10fbcca1bb180c5cfc805a37240945a6ba703cb1f985d4d3b8a1c954fc5b
Deleted: sha256:dcec89b921d64a1d2f748c450832a4c5624219bbb1b696e1a606bff78b7afa60
Deleted: sha256:4994a8d5939af297abf594b7ab714dfb72e57217b94bf0a250a62903cbdb6d84
Deleted: sha256:8b2bc37f672b15bea06a204790da9f384ef7b06feccade3fd3b1945b63df5aef
Deleted: sha256:a4197512070a01764db77b424100c1f81f0ed696380b19bc26b6e72fafef0709
Deleted: sha256:32148f9f6c5aadfa167ee7b146b9703c59307049d68b090c19db019fd15c5406
einfochips@AHMLPT1474:~/Day4/my-webapp$ sudo docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
einfochips@AHMLPT1474:~/Day4/my-webapp$ 
```

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:

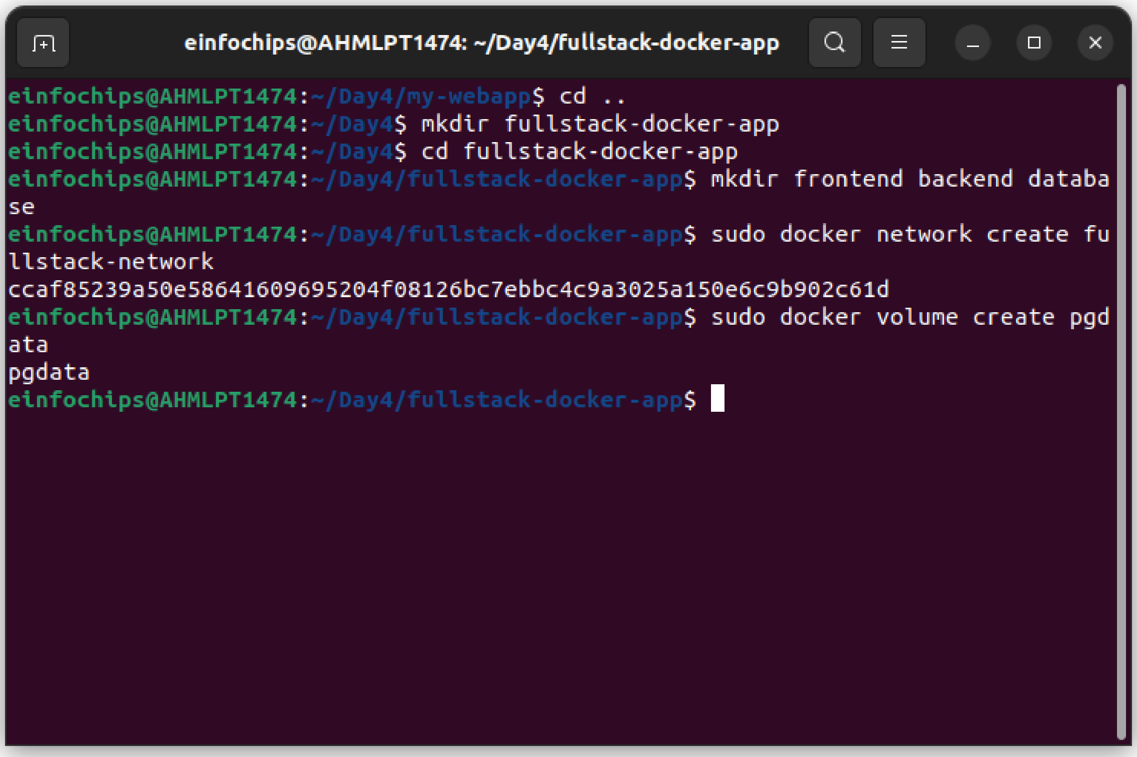
```
mkdir frontend backend database
```

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@AHMLPT1474: ~/Day4/fullstack-docker-app

```
einfochips@AHMLPT1474:~/Day4/my-webapp$ cd ..
einfochips@AHMLPT1474:~/Day4$ mkdir fullstack-docker-app
einfochips@AHMLPT1474:~/Day4$ cd fullstack-docker-app
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ mkdir frontend backend databa
se
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ sudo docker network create fu
llstack-network
ccaf85239a50e58641609695204f08126bc7ebbc4c9a3025a150e6c9b902c61d
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ sudo docker volume create pgd
ata
pgdata
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ 
```

Part 2: Setting Up the Database

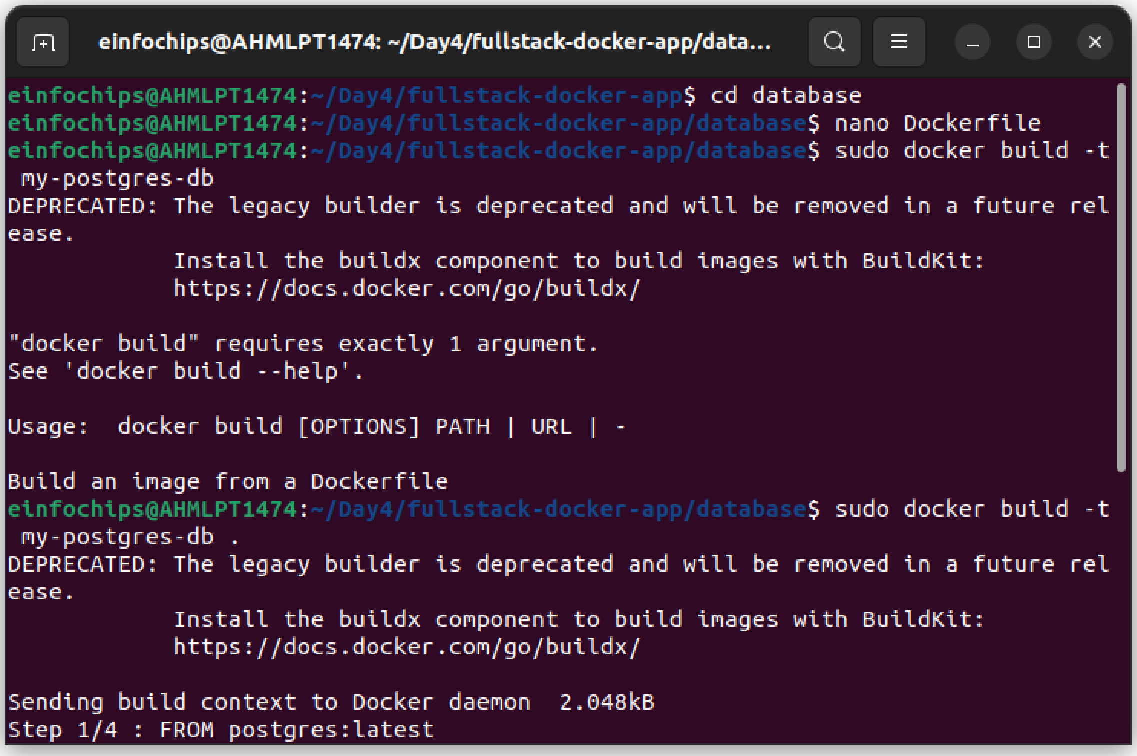
Objective: Set up a PostgreSQL database with Docker.

Steps:

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

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

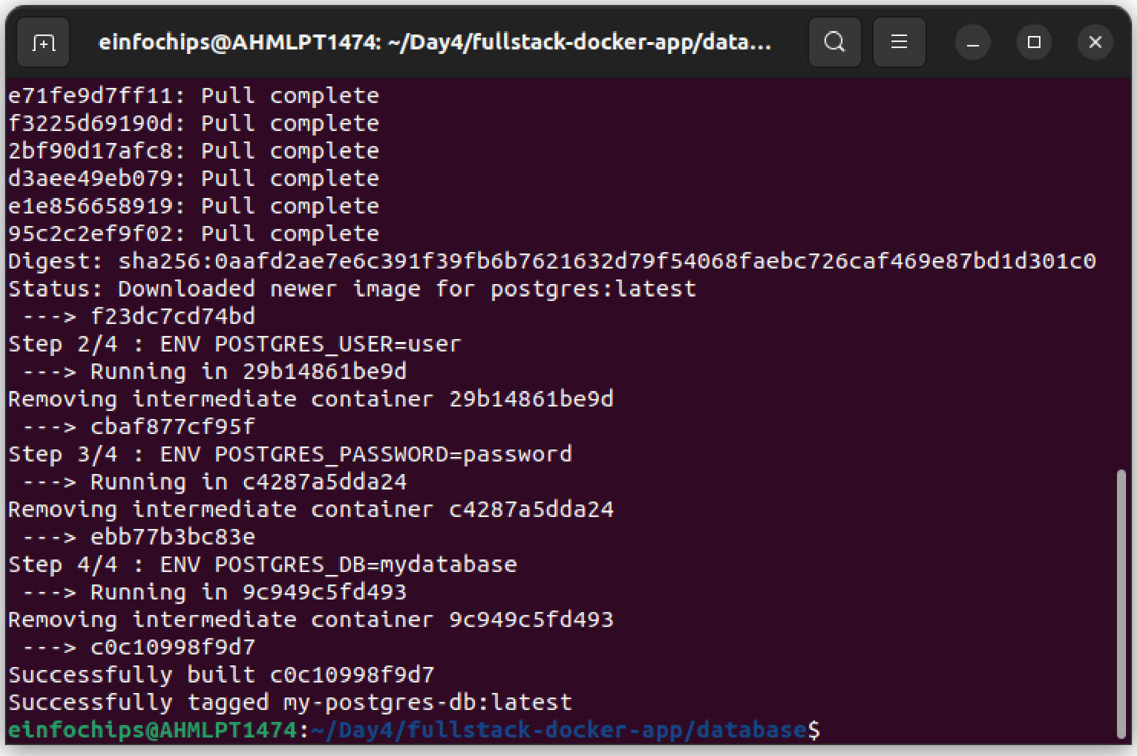
```
FROM postgres:latest
ENV POSTGRES_USER=user
ENV POSTGRES_PASSWORD=password
ENV POSTGRES_DB=mydatabase
```



```
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ cd database
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/database$ nano Dockerfile
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/database$ sudo docker build -t
 my-postgres-db
DEPRECATED: The legacy builder is deprecated and will be removed in a future rel
ease.
            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
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/database$ sudo docker build -t
 my-postgres-db .
DEPRECATED: The legacy builder is deprecated and will be removed in a future rel
ease.
            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
```

Build the PostgreSQL Image:

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

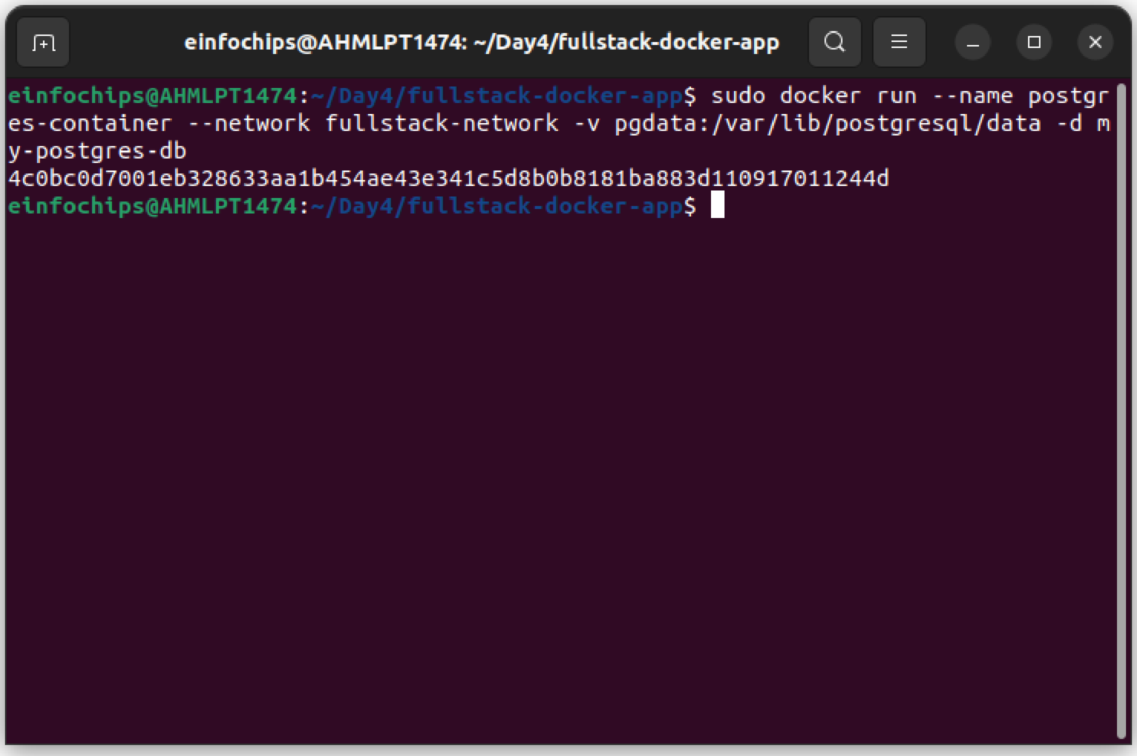


```
einfochips@AHMLPT1474: ~/Day4/fullstack-docker-app/data...
e71fe9d7ff11: Pull complete
f3225d69190d: Pull complete
2bf90d17afc8: Pull complete
d3aee49eb079: Pull complete
e1e856658919: Pull complete
95c2c2ef9f02: Pull complete
Digest: sha256:0aafd2ae7e6c391f39fb6b7621632d79f54068faebc726caf469e87bd1d301c0
Status: Downloaded newer image for postgres:latest
 ---> f23dc7cd74bd
Step 2/4 : ENV POSTGRES_USER=user
 ---> Running in 29b14861be9d
Removing intermediate container 29b14861be9d
 ---> cbaf877cf95f
Step 3/4 : ENV POSTGRES_PASSWORD=password
 ---> Running in c4287a5dda24
Removing intermediate container c4287a5dda24
 ---> ebb77b3bc83e
Step 4/4 : ENV POSTGRES_DB=mydatabase
 ---> Running in 9c949c5fd493
Removing intermediate container 9c949c5fd493
 ---> c0c10998f9d7
Successfully built c0c10998f9d7
Successfully tagged my-postgres-db:latest
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/database$
```

2.

Run the PostgreSQL Container:

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



```
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ sudo docker run --name postgr
es-container --network fullstack-network -v pgdata:/var/lib/postgresql/data -d m
y-postgres-db
4c0bc0d7001eb328633aa1b454ae43e341c5d8b0b8181ba883d110917011244d
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ 
```

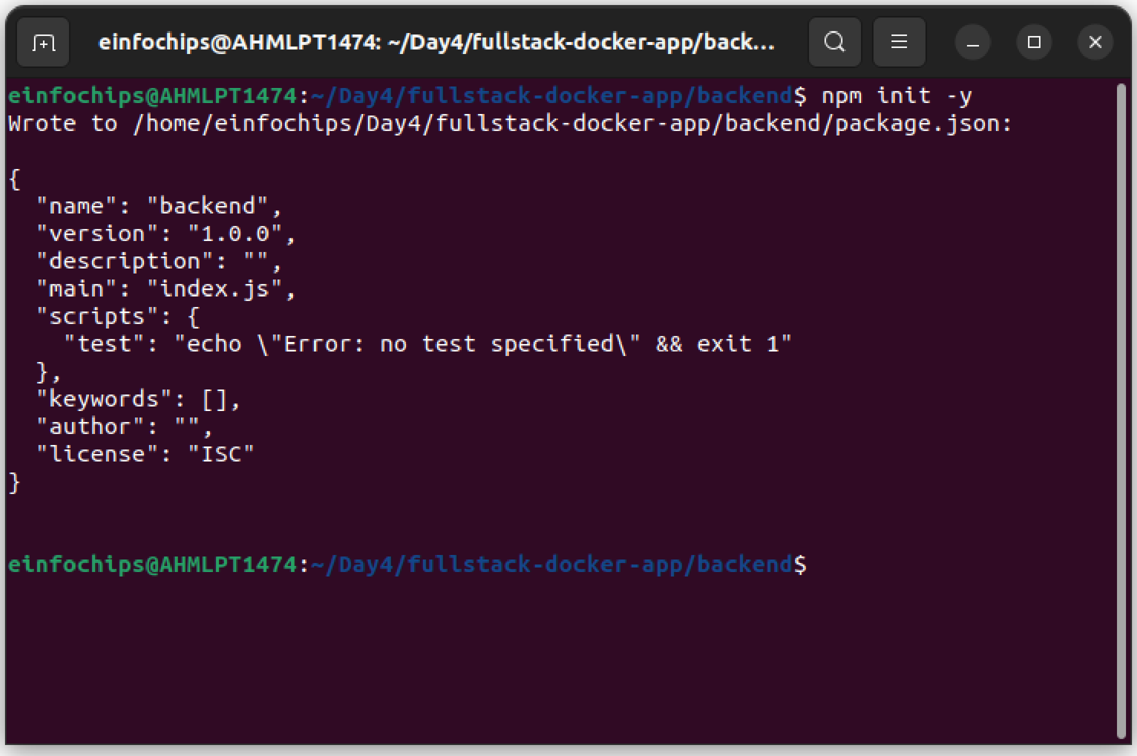
Part 3: Setting Up the Backend (Node.js with Express)

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
```

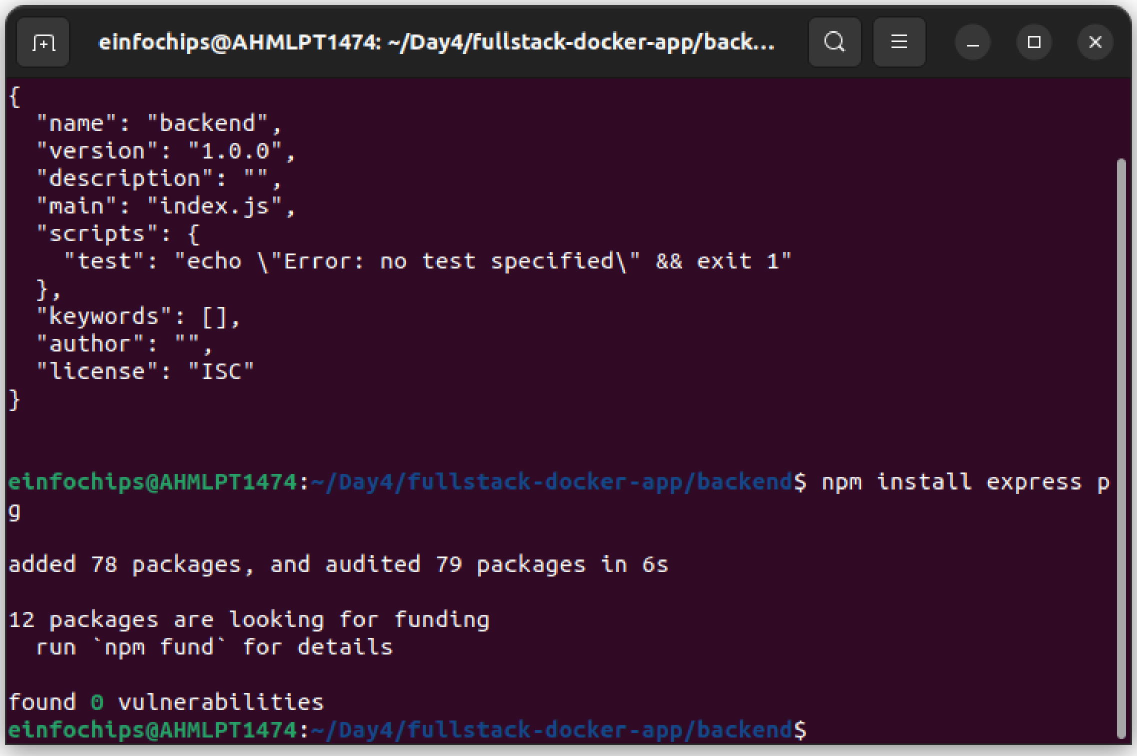


```
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$ npm init -y
Wrote to /home/einfochips/Day4/fullstack-docker-app/backend/package.json:

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


einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$
```

1.

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

```
npm install express pg
```


einfochips@AHMLPT1474: ~/Day4/fullstack-docker-app/back...

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


einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$ npm install express p
g

added 78 packages, and audited 79 packages in 6s

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

found 0 vulnerabilities
einfochips@AHMLPT1474:~/Day4/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()');
```

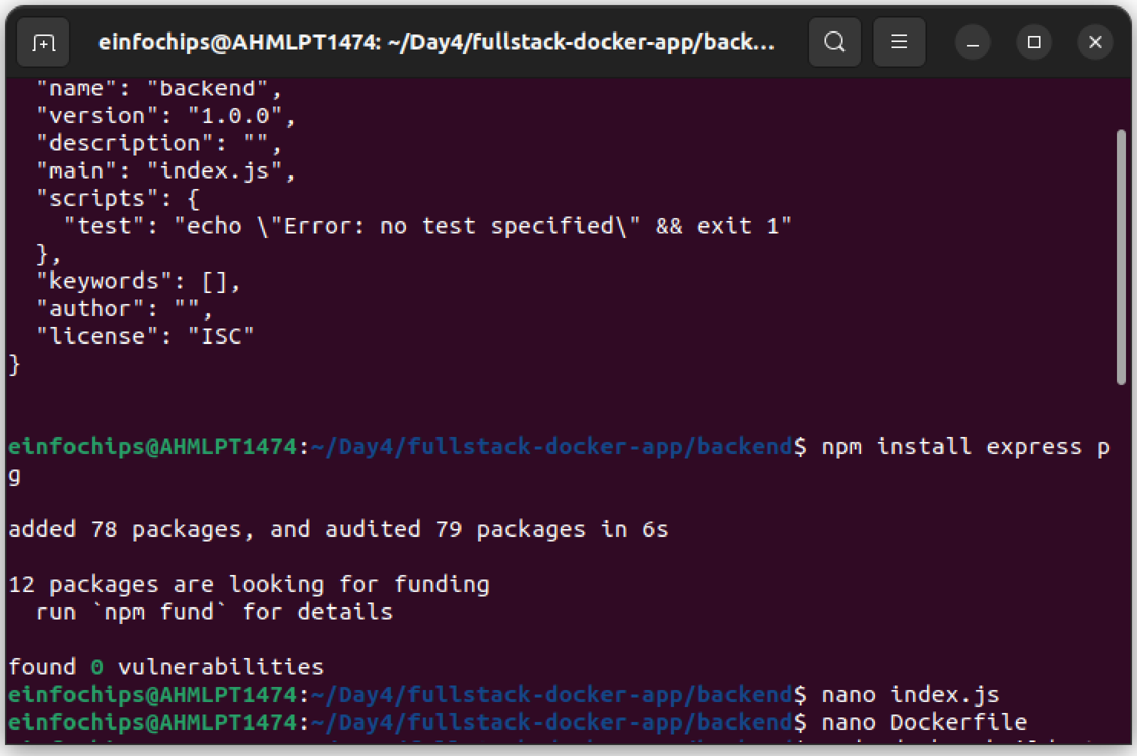
```
    client.release();
    res.send(result.rows);
});

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:

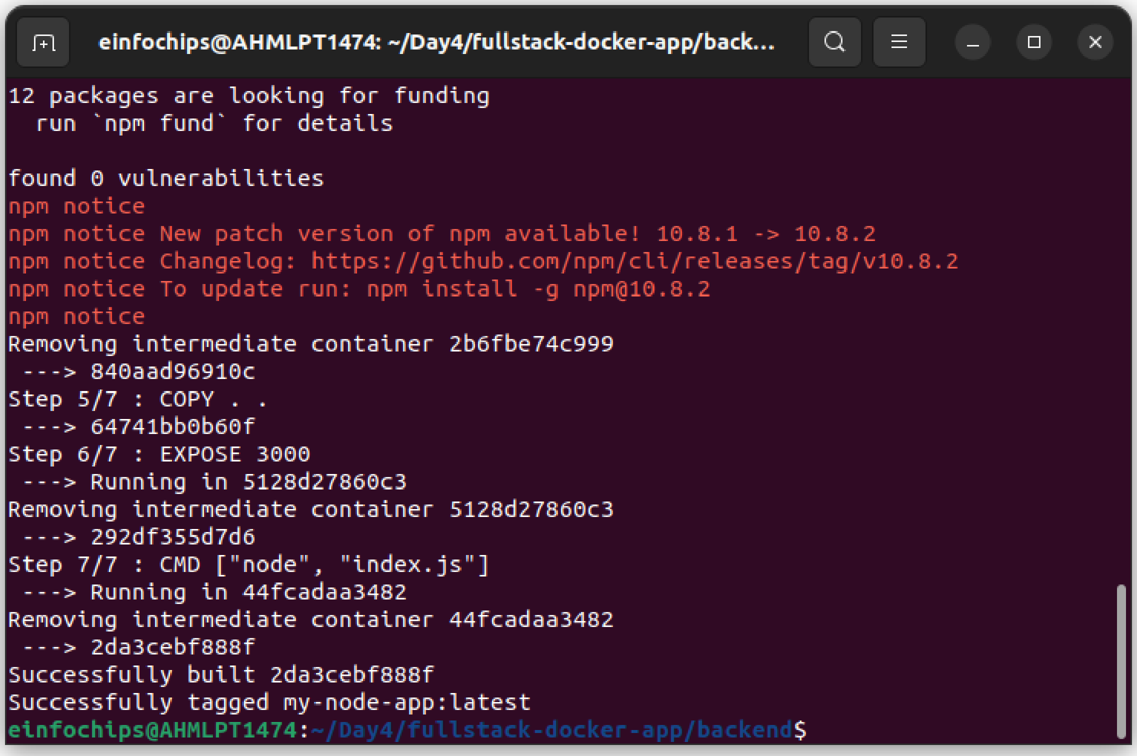
```
FROM node:latest

WORKDIR /usr/src/app

COPY package*.json ./
RUN npm install

COPY . .

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



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


einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$ npm install express p
g

added 78 packages, and audited 79 packages in 6s

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

found 0 vulnerabilities
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$ nano index.js
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$ nano Dockerfile
```

Build the Backend Image:

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

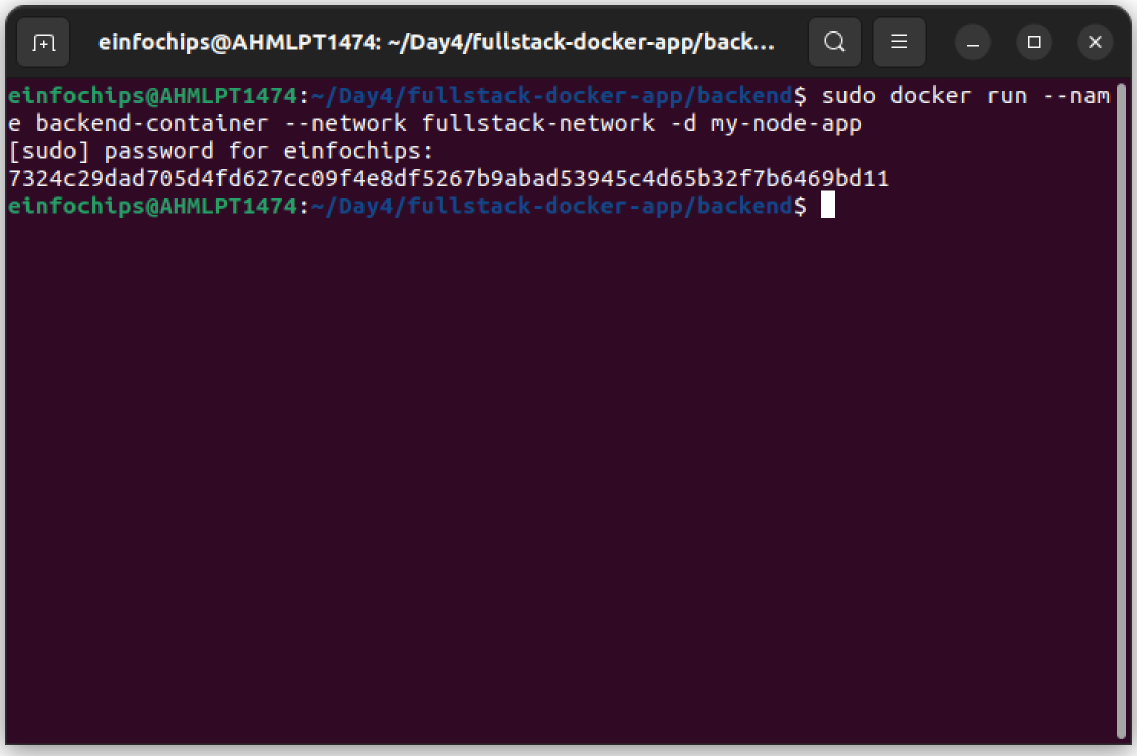

einfochips@AHMLPT1474: ~/Day4/fullstack-docker-app/back...

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

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 2b6fbe74c999
 ---> 840aad96910c
Step 5/7 : COPY . .
 ---> 64741bb0b60f
Step 6/7 : EXPOSE 3000
 ---> Running in 5128d27860c3
Removing intermediate container 5128d27860c3
 ---> 292df355d7d6
Step 7/7 : CMD ["node", "index.js"]
 ---> Running in 44fcadaa3482
Removing intermediate container 44fcadaa3482
 ---> 2da3cebf888f
Successfully built 2da3cebf888f
Successfully tagged my-node-app:latest
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$
```

5.

Run the Backend Container:

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



```
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$ sudo docker run --nam
e backend-container --network fullstack-network -d my-node-app
[sudo] password for einfochips:
7324c29dad705d4fd627cc09f4e8df5267b9abad53945c4d65b32f7b6469bd11
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/backend$
```

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:

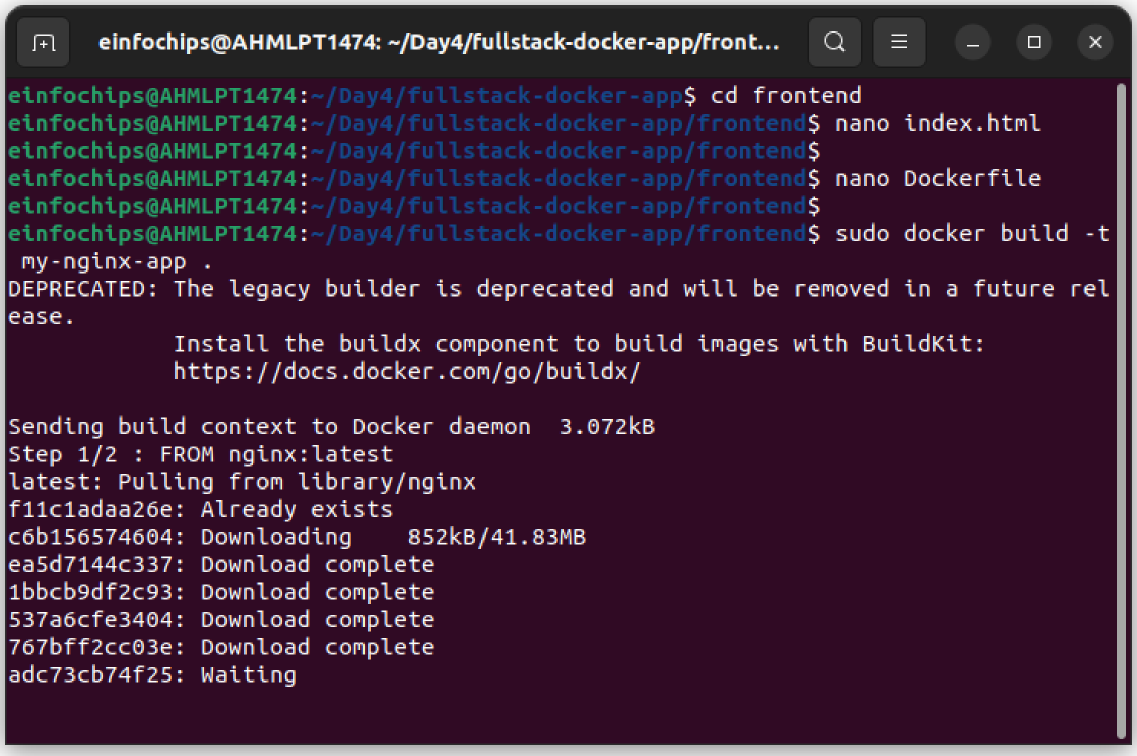
```
<!DOCTYPE html>
<html>
<body>
  <h1>Hello from Nginx and Docker!</h1>
  <p>This is a simple static front-end served by Nginx.</p>
</body>
</html>
```

○

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
```



```
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ cd frontend
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ nano index.html
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ nano Dockerfile
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ sudo docker build -t
 my-nginx-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future rel
ease.
            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: Downloading    852kB/41.83MB
ea5d7144c337: Download complete
1bbcb9df2c93: Download complete
537a6cfe3404: Download complete
767bff2cc03e: Download complete
adc73cb74f25: Waiting
```

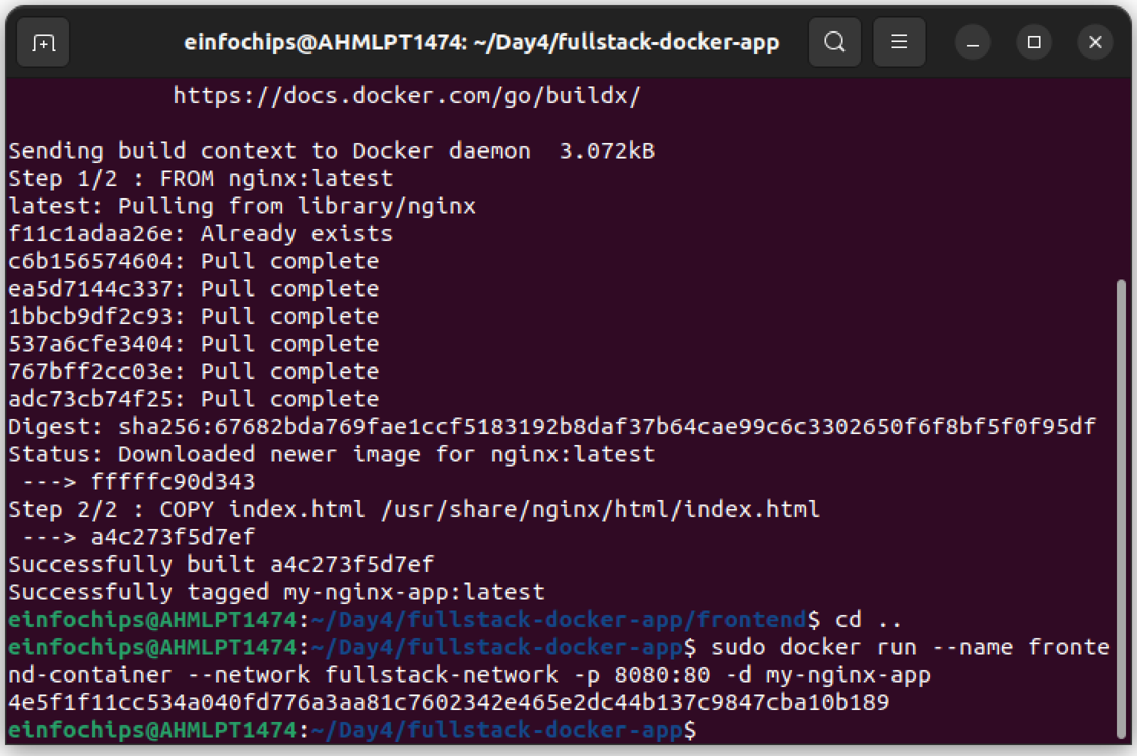
Build the Frontend Image:

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

3.

Run the Frontend Container:

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



einfochips@AHMLPT1474: ~/Day4/fullstack-docker-app

```
            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: sha256:67682bda769fae1ccf5183192b8daf37b64cae99c6c3302650f6f8bf5f0f95df
Status: Downloaded newer image for nginx:latest
 ---> fffffc90d343
Step 2/2 : COPY index.html /usr/share/nginx/html/index.html
 ---> a4c273f5d7ef
Successfully built a4c273f5d7ef
Successfully tagged my-nginx-app:latest
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ cd ..
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ sudo docker run --name fronte
nd-container --network fullstack-network -p 8080:80 -d my-nginx-app
4e5f1f11cc534a040fd776a3aa81c7602342e465e2dc44b137c9847cba10b189
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$
```

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:

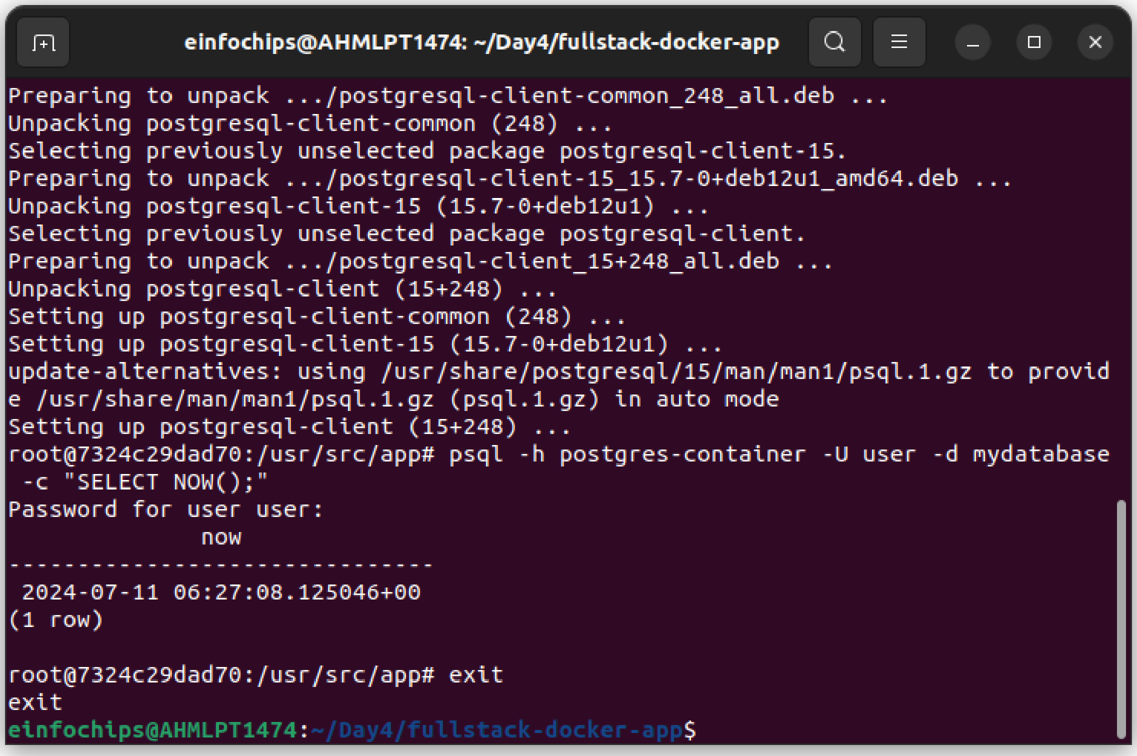
```
docker exec -it backend-container /bin/bash
```

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();"
```

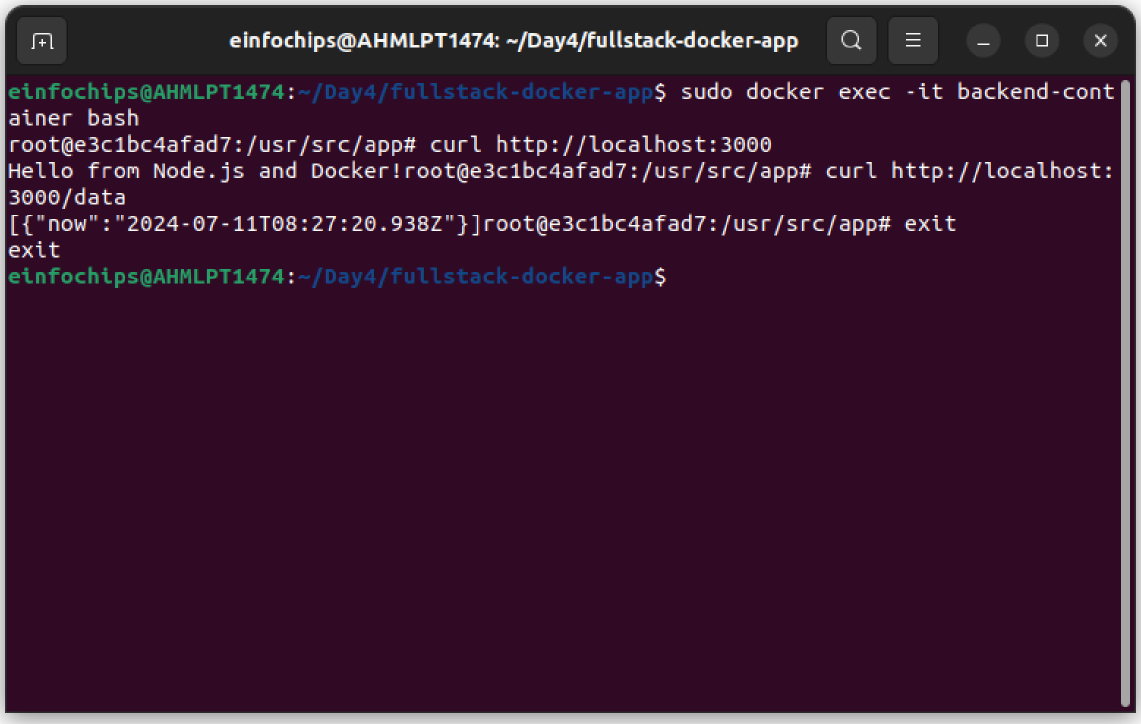
Exit the container:

exit



```
Preparing to unpack .../postgresql-client-common_248_all.deb ...
Unpacking postgresql-client-common (248) ...
Selecting previously unselected package postgresql-client-15.
Preparing to unpack .../postgresql-client-15_15.7-0+deb12u1_amd64.deb ...
Unpacking postgresql-client-15 (15.7-0+deb12u1) ...
Selecting previously unselected package postgresql-client.
Preparing to unpack .../postgresql-client_15+248_all.deb ...
Unpacking postgresql-client (15+248) ...
Setting up postgresql-client-common (248) ...
Setting up postgresql-client-15 (15.7-0+deb12u1) ...
update-alternatives: using /usr/share/postgresql/15/man/man1/psql.1.gz to provid
e /usr/share/man/man1/psql.1.gz (psql.1.gz) in auto mode
Setting up postgresql-client (15+248) ...
root@7324c29dad70:/usr/src/app# psql -h postgres-container -U user -d mydatabase
 -c "SELECT NOW();"
Password for user user:
              now
-------------------------------
 2024-07-11 06:27:08.125046+00
(1 row)

root@7324c29dad70:/usr/src/app# exit
exit
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$
```

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.



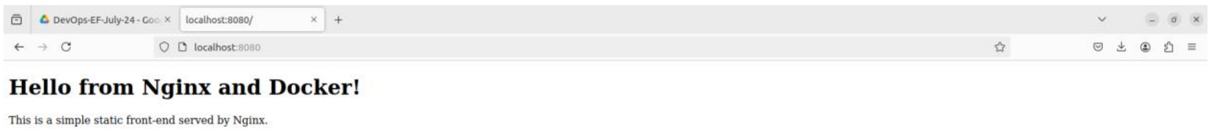
```
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app$ sudo docker exec -it backend-cont
ainer bash
root@e3c1bc4afad7:/usr/src/app# curl http://localhost:3000
Hello from Node.js and Docker!root@e3c1bc4afad7:/usr/src/app# curl http://localhost:
3000/data
[{"now":"2024-07-11T08:27:20.938Z"}]root@e3c1bc4afad7:/usr/src/app# exit
exit
einfochips@AHMLPT1474:~/Day4/fullstack-docker-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.



Hello from Nginx and Docker!

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

2. **Verify Full Integration:**

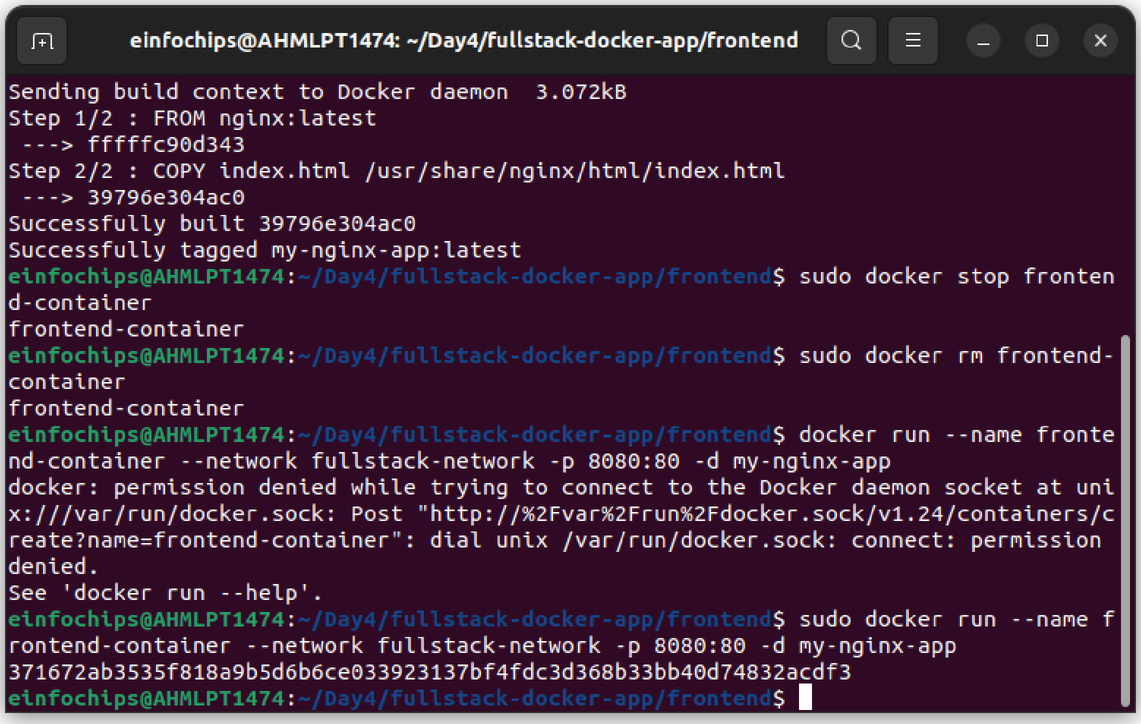
Update the index.html to include a link to the backend:

```
<!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 ..
```



```
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
 ---> 39796e304ac0
Successfully built 39796e304ac0
Successfully tagged my-nginx-app:latest
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ sudo docker stop fronten
d-container
frontend-container
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ sudo docker rm frontend-
container
frontend-container
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ docker run --name fronte
nd-container --network fullstack-network -p 8080:80 -d my-nginx-app
docker: permission denied while trying to connect to the Docker daemon socket at uni
x:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/c
reate?name=frontend-container": dial unix /var/run/docker.sock: connect: permission
denied.
See 'docker run --help'.
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ sudo docker run --name f
rontend-container --network fullstack-network -p 8080:80 -d my-nginx-app
371672ab3535f818a9b5d6b6ce033923137bf4fdc3d368b33bb40d74832acdf3
einfochips@AHMLPT1474:~/Day4/fullstack-docker-app/frontend$ 
```

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

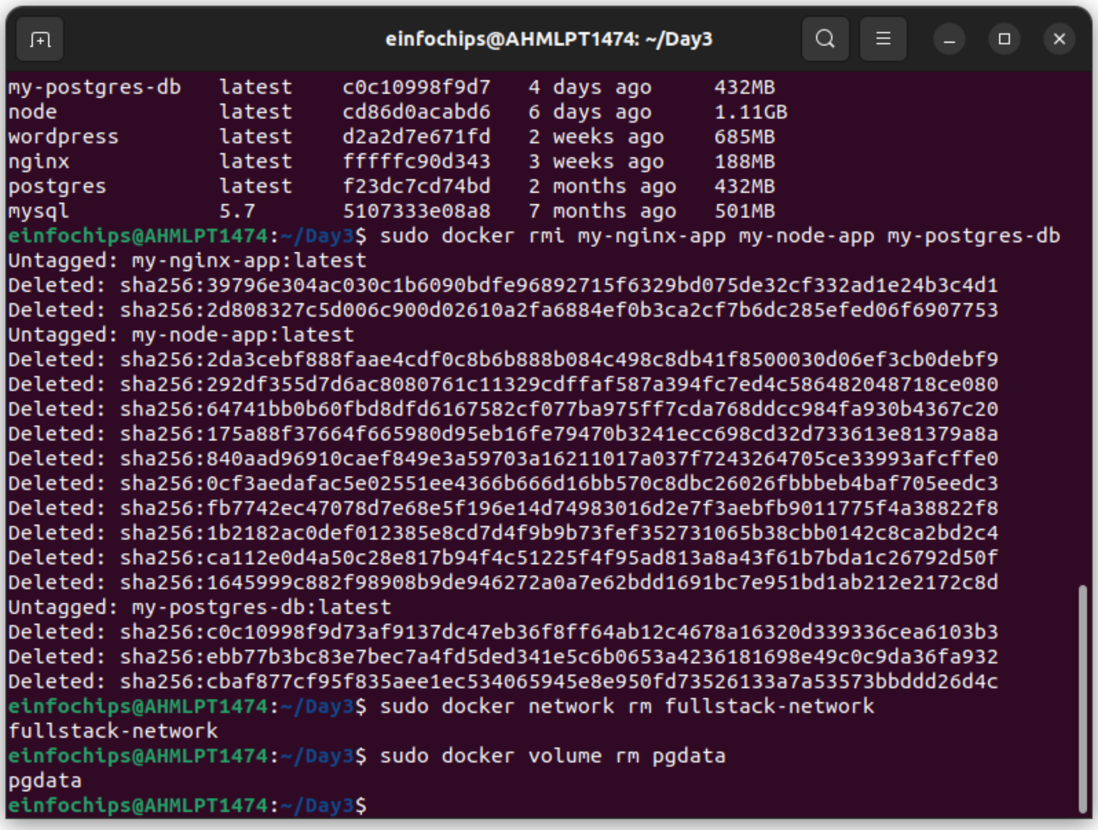
Remove the Images:

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

2.

Remove the Network and Volume:

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



```
my-postgres-db   latest    c0c10998f9d7   4 days ago     432MB
node             latest    cd86d0acabd6   6 days ago     1.11GB
wordpress        latest    d2a2d7e671fd   2 weeks ago    685MB
nginx            latest    fffffc90d343   3 weeks ago    188MB
postgres         latest    f23dc7cd74bd   2 months ago   432MB
mysql            5.7       5107333e08a8   7 months ago   501MB
einfochips@AHMLPT1474:~/Day3$ sudo docker rmi my-nginx-app my-node-app my-postgres-db
Untagged: my-nginx-app:latest
Deleted: sha256:39796e304ac030c1b6090bdfe96892715f6329bd075de32cf332ad1e24b3c4d1
Deleted: sha256:2d808327c5d006c900d02610a2fa6884ef0b3ca2cf7b6dc285efed06f6907753
Untagged: my-node-app:latest
Deleted: sha256:2da3cebf888faae4cdf0c8b6b888b084c498c8db41f8500030d06ef3cb0debf9
Deleted: sha256:292df355d7d6ac8080761c11329cdffaf587a394fc7ed4c586482048718ce080
Deleted: sha256:64741bb0b60fbd8dfd6167582cf077ba975ff7cda768ddcc984fa930b4367c20
Deleted: sha256:175a88f37664f665980d95eb16fe79470b3241ecc698cd32d733613e81379a8a
Deleted: sha256:840aad96910caef849e3a59703a16211017a037f7243264705ce33993afcffe0
Deleted: sha256:0cf3aedafac5e02551ee4366b666d16bb570c8dbc26026fbbbeb4baf705eedc3
Deleted: sha256:fb7742ec47078d7e68e5f196e14d74983016d2e7f3aebfb9011775f4a38822f8
Deleted: sha256:1b2182ac0def012385e8cd7d4f9b9b73fef352731065b38cbb0142c8ca2bd2c4
Deleted: sha256:ca112e0d4a50c28e817b94f4c51225f4f95ad813a8a43f61b7bda1c26792d50f
Deleted: sha256:1645999c882f98908b9de946272a0a7e62bdd1691bc7e951bd1ab212e2172c8d
Untagged: my-postgres-db:latest
Deleted: sha256:c0c10998f9d73af9137dc47eb36f8ff64ab12c4678a16320d339336cea6103b3
Deleted: sha256:ebb77b3bc83e7bec7a4fd5ded341e5c6b0653a4236181698e49c0c9da36fa932
Deleted: sha256:cbaf877cf95f835aee1ec534065945e8e950fd73526133a7a53573bbddd26d4c
einfochips@AHMLPT1474:~/Day3$ sudo docker network rm fullstack-network
fullstack-network
einfochips@AHMLPT1474:~/Day3$ sudo docker volume rm pgdata
pgdata
einfochips@AHMLPT1474:~/Day3$
```