

Git Cherry Pick

Scenario:

- You have two branches: **branch-A** and **branch-B**.
- You made a bug fix commit on **branch-A** that you now want to apply to **branch-B** without merging all changes from **branch-A** into **branch-B**.

Steps:

Identify the Commit:

First, find the commit hash of the bug fix commit on **branch-A**:

```
git log --oneline branch-A
```

1. Suppose the commit hash is **abcdef1234567890**.

Switch to **branch-B**:

Ensure you are on **branch-B** where you want to apply the bug fix:

```
git checkout branch-B
```

- 2.

Cherry-pick the Commit:

Apply the bug fix commit from **branch-A** to **branch-B**:

```
git cherry-pick abcdef1234567890
```

3. This command applies the changes introduced by the commit **abcdef1234567890** onto **branch-B**.

Resolve Conflicts (if any):

```
git cherry-pick --continue
```

- 4.

Commit the Cherry-picked Changes:

After resolving conflicts (if any), commit the cherry-picked changes on **branch-B**:

```
git commit
```

5. This creates a new commit on **branch-B** that includes the changes from **branch-A**'s selected commit.

Git Stash

Step 1: Initialize a Git Repository

First, create a new directory for your project and initialize a Git repository:

```
mkdir git-stash-example  
cd git-stash-example  
git init
```

Step 2: Add and Commit Files

Create some files and add content to them:

```
echo "This is file1.txt" > file1.txt  
echo "This is file2.txt" > file2.txt
```

Add these files to the staging area and commit them:

```
git add file1.txt file2.txt  
git commit -m "Initial commit - Added file1.txt and file2.txt"
```

Step 3: Modify Files

Make some changes to `file1.txt`:

```
echo "Updated content in file1.txt" >> file1.txt
```

Step 4: Use `git stash`

Now, let's use `git stash` to temporarily store the changes in `file1.txt` without committing them:

```
git stash save "WIP: Work in progress changes"
```

This command saves your changes (in this case, the update to `file1.txt`) to a stash with a message "WIP: Work in progress changes".

Step 5: Verify Stash

You can verify the stash list using:

```
git stash list
```

It should show something like:

```
stash@{0}: On master: WIP: Work in progress changes
```

Step 6: Check Working Directory Status

Check the status of your working directory:

```
git status
```

It should indicate that your working directory is clean (no changes).

Step 7: Apply Stashed Changes

Let's apply the stashed changes back into your working directory:

```
git stash pop
```

Step 8: Verify Changes

Check the changes in `file1.txt`:

```
cat file1.txt
```

Step 9: Commit Stashed Changes

If you are satisfied with the changes, commit them:

```
git add file1.txt  
git commit -m "Updated file1.txt with stashed changes"
```

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`

1.

Run the Nginx Container:

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

2.

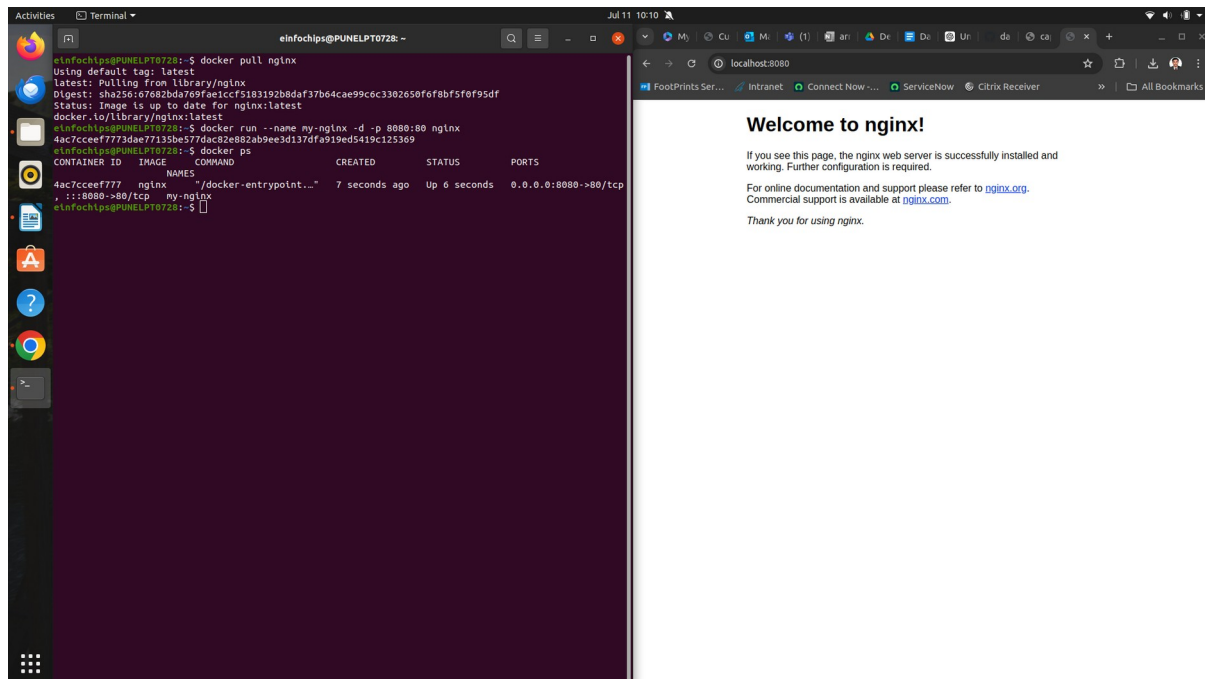
- `--name my-nginx`: Assigns a name to the container.
- `-d`: Runs the container in detached mode.
- `-p 8080:80`: Maps port 8080 on your host to port 80 in the container.

Verify the Container is Running:

`docker ps`

3.

- Visit <http://localhost:8080> in your browser. You should see the Nginx welcome page.



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
```

4.

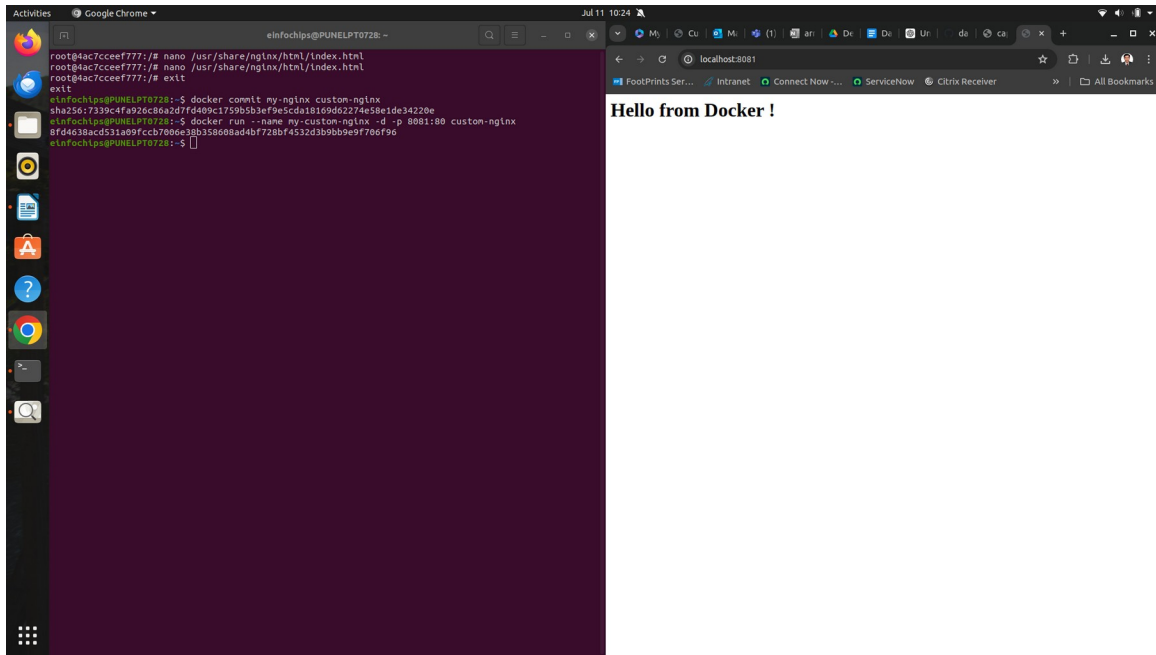
Run a Container from the New Image:

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

5.

6. Verify the New Container:

- Visit <http://localhost:8081> in your browser. You should see your custom HTML page.



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

○

Build the Docker Image:

docker build -t my-webapp-image .

4.

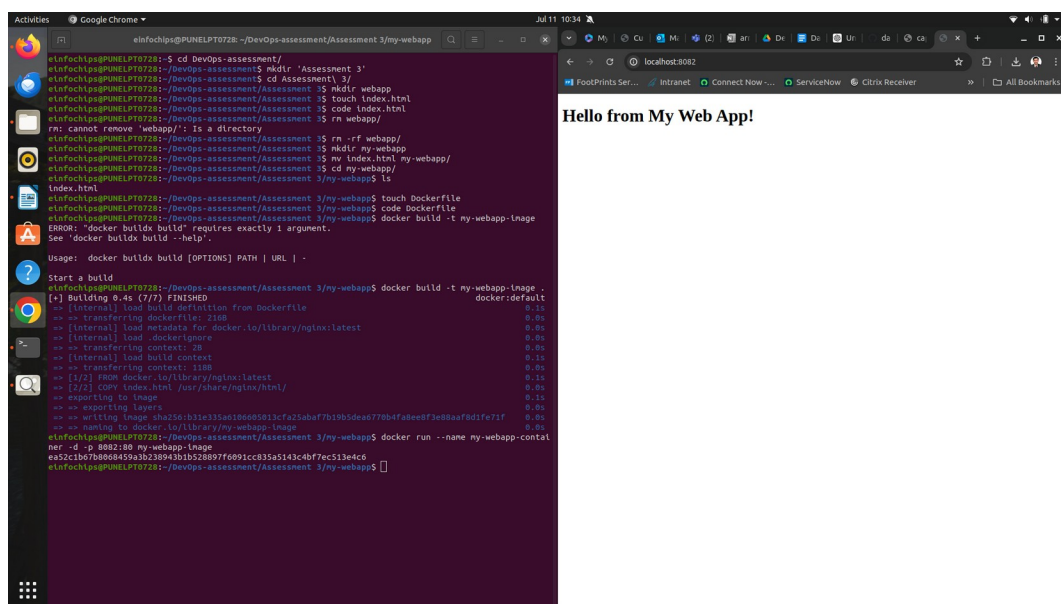
Run a Container from the Built Image:

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

5.

6. Verify the Web Application:

- Visit <http://localhost:8082> in your browser. You should see your custom web application.



The screenshot shows a terminal window with the following commands and output:

```
elinfochips@PUNELPT0728: ~/DevOps-assessment/Assessment 3/my-webapp
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ cd DevOps-assessment/
elinfochips@PUNELPT0728:~/DevOps-assessment$ mkdir 'Assessment 3'
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ cd Assessment 3/
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ mkdir webapp
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ touch index.html
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ code index.html
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ cd my-webapp/
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ rm webapp/
rm: cannot remove 'webapp/': is a directory
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ rm -rf webapp/
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ mkdir my-webapp/
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ mv index.html my-webapp/
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ cd my-webapp/
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ ls
index.html
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ touch Dockerfile
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ my-webapp$ code Dockerfile
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ my-webapp$ docker build -t my-webapp-image
ERROR: "docker buildx build" requires exactly 1 argument.
See "docker buildx build --help".

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

Start a build
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ my-webapp$ docker build -t my-webapp-image .
[+] Building 0.4s (7/7) FINISHED
    docker:default
    => [internal] load build definition from Dockerfile
    ==> transferring dockerfile: 210B
    ==> [internal] load metadata for docker.io/library/nginx:latest
    ==> [internal] load .dockerignore
    ==> transferring context: 2B
    ==> [internal] load build context
    ==> transferring context: 118B
    ==> [2/2] FROM docker.io/library/nginx:latest
    ==> [2/2] COPY index.html /usr/share/nginx/html/
    ==> exporting to image
    ==> writing image sha256:b31e33a6180005913fa25abaf7b19b5de6770b4fabe9f3e8baaf8d1fe71f
    ==> naming to docker.io/library/my-webapp-image
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ my-webapp$ docker run --name my-webapp-container -d -p 8082:80 my-webapp-image
e2821c1b70b084f9a3b230943b1b5218897f6091cc835a5143c4bf7ec513e4c6
elinfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ my-webapp$
```

The browser window shows the URL localhost:8082 and the message "Hello from My Web App!"

Part 4: Cleaning Up

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

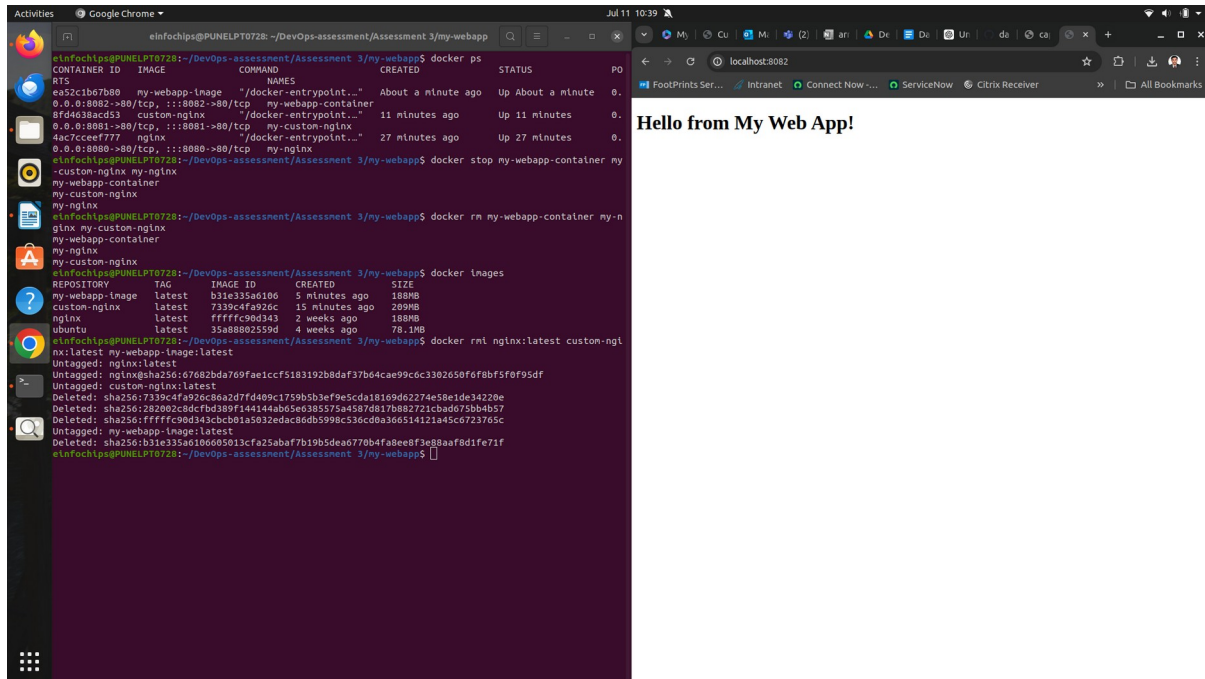
Steps:

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
```



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:**

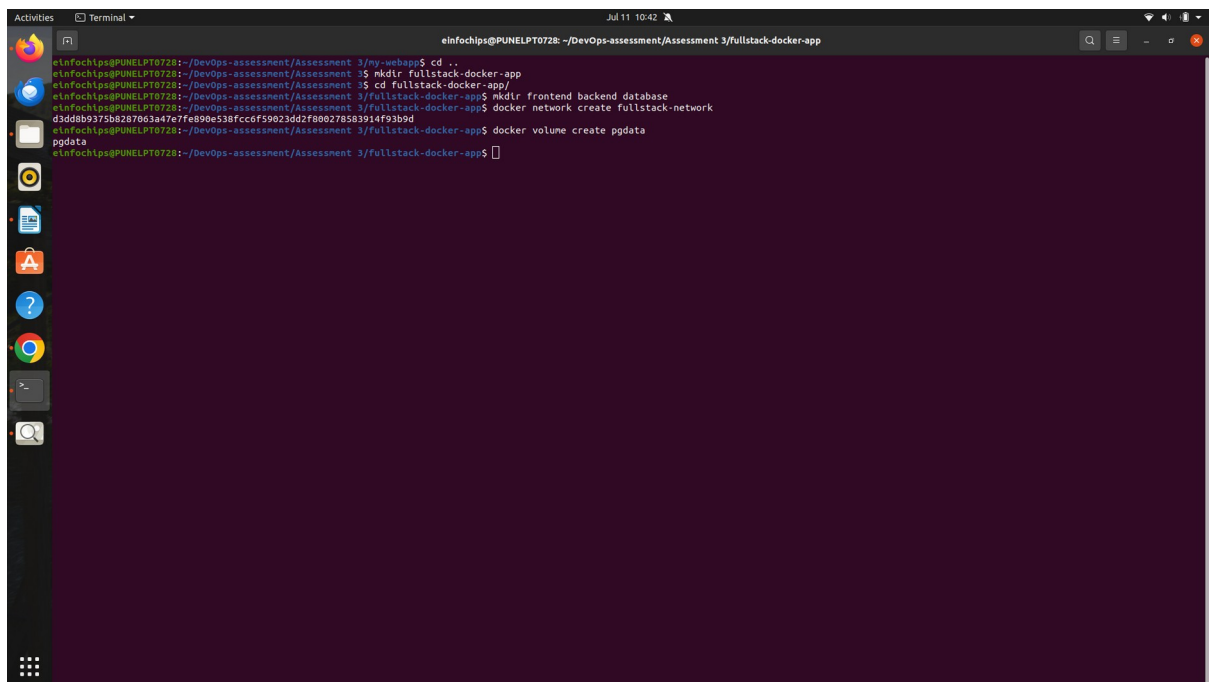
- Docker allows communication between containers through a shared network.

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

3.

- Create a volume for the PostgreSQL database.

```
docker volume create pgdata
```



```
einfochips@PUNELPT0728: ~/DevOps-assessment/Assessment 3/fullstack-docker-app  
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ cd ..  
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ mkdir fullstack-docker-app  
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3$ cd fullstack-docker-app/  
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ mkdir frontend backend database  
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker network create fullstack-network  
d3ddb9375b8287663a47e7fe890e538fccc59023dd2f800278583914f93b9d  
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker volume create pgdata  
pgdata  
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/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
```

○

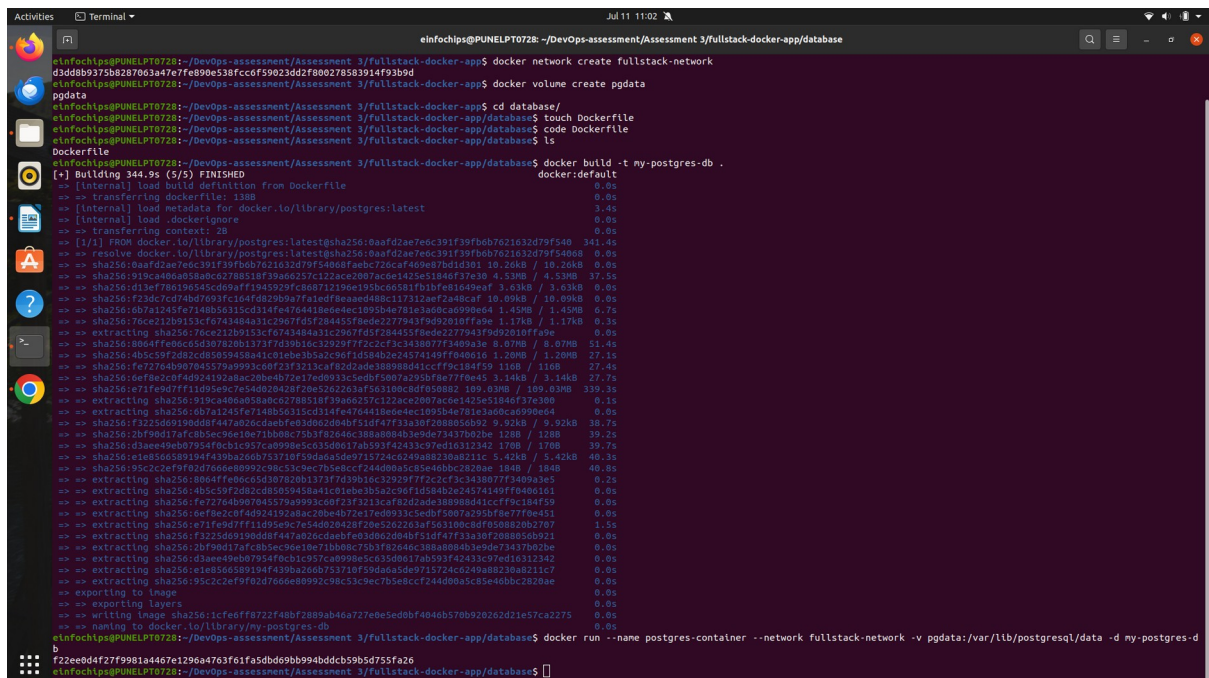
Build the PostgreSQL Image:

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

2.

Run the PostgreSQL Container:

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



```
Activities Terminal
einfochips@PUNELPT0728: ~/DevOps-assessment/Assessment 3/fullstack-docker-app/database
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker network create fullstack-network
d3d4db9375b287663a4e7fe890e538fccc59023dd2f800278583914f93b9d
pgdata
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ cd database/
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/database$ touch Dockerfile
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/database$ code Dockerfile
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/database$ ls
Dockerfile
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/database$ docker build -t my-postgres-db .
[+] Building 344.9s (5/5) FINISHED docker:default
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 130B
=> [internal] load metadata for docker.io/library/postgres:latest
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/1] FROM docker.io/library/postgres:latest@sha256:0aaf2ae7ec391f39fbb7621632d79f5a0 341.4s
=> resolve docker.io/library/postgres:latest@sha256:0aaf2ae7ec391f39fbb7621632d79f5a0
=> sha256:0aaf2ae7ec391f39fbb7621632d79f5a08faebc726caf469eb7bd1d301 10.26kB / 10.26kB
=> sha256:d19ca48083806c2788518f3860d237c122cc2807acee1425e53840f3730 4.53MB / 4.53MB
=> sha256:d13ef780196545cd9aaff1945929fc86071219ae195bc6581fb1bfed1649eaf 3.63kB / 3.63kB
=> sha256:f23dc7cd74bd7093fc164f4829b9a7faedf8eaaed48cc17312aef2a48caf 10.09kB / 10.09kB
=> sha256:6b7a1245f7148b5631cd3147e476418e0e4ec1095b4e761e3a0ca990e64 1.45MB / 1.45MB
=> sha256:76cc212b9153cf674348a31c2967f5f284455f8ede2277943f9d92810f9ae 0.0s
=> extracting sha256:76cc212b9153cf674348a31c2967f5f284455f8ede2277943f9d92810f9ae 0.0s
=> sha256:8064ff6e0c5d307820b13737f439b16c32929f7f2c2c3c3438077f3409ae 8.07MB / 8.07MB
=> sha256:405c9f2d02d80594c584c1e1eb3b32cefd594b2e45741d9f04016 9.20MB / 9.20MB
=> sha256:fe727649b07045579a993c60f23f321cafa82d2ade38898bd41ccff9c184f59 116B / 116B
=> sha256:6ef8e2c0f4d924192a8ac20be4b72e17ed0933c5ebf5807a295b8e77f0e45 3.14kB / 3.14kB
=> extracting sha256:1919ca40e0a08ba0c2788518f39a60257c122ace2007ac0e1425e51846f37e300 0.1s
=> extracting sha256:6b7a1245f7148b5631cd3147e476418e0e4ec1095b4e761e3a0ca990e64 0.0s
=> sha256:f325d09196d8f447a26cdaebf63062d04bf51d4f7f33a30f208050b92 9.92kB / 9.92kB
=> sha256:2bf0e017afcb3c9e1b71b00873d3f824dc38a8a08b3e0de7937802b6 128B / 128B
=> sha256:d3aea49eb7954f0cb1c957ca0998e5c35d0617ab593f4243c97ed1631242 170B / 170B
=> sha256:e1e856589194f439ba260b753710f59da6a5de9715724c249a88230a8211c 5.42kB / 5.42kB
=> sha256:b95c2cef902f0d6e0e992c98c53c9c7b5e8cc744d00a5c5e40bcb2820ae 184B / 184B
=> extracting sha256:8064ff6e0c5d307820b13737f439b16c32929f7f2c2c3c3438077f3409ae5 0.2s
=> extracting sha256:4b5c59f2d82cd85859458a4c1e1eb3b32cefd594b2e45741d9f040161 0.0s
=> extracting sha256:fe727649b07045579a993c60f23f321cafa82d2ade38898bd41ccff9c184f59 0.0s
=> extracting sha256:6ef8e2c0f4d924192a8ac20be4b72e17ed0933c5ebf5807a295b8e77f0e451 0.0s
=> sha256:e71fe9d7ff1d95e9c7e540b2428f20e5262263af563100c8df0508820b2707 1.5s
=> extracting sha256:f325d09196d8f447a26cdaebf63062d04bf51d4f7f33a30f208050b921 0.0s
=> extracting sha256:d390d172fcd85c9ee1b71b00873d3f824dc38a8a08b3e0de7937802b6 0.0s
=> sha256:d3aea49eb7954f0cb1c957ca0998e5c35d0617ab593f4243c97ed1631242 0.0s
=> extracting sha256:e1e856589194f439ba260b753710f59da6a5de9715724c249a88230a8211c7 0.0s
=> extracting sha256:95c2cef902f0d6e0e992c98c53c9c7b5e8cc744d00a5c5e40bcb2820ae 0.0s
=> exporting to image
=> exporting layers
=> writing image sha256:1cfe0f0722f40bf2089ab46a727e8e5d0bf4040b570b920262d21e57ca2275 0.0s
=> naming to docker.io/library/my-postgres-db
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/database$ docker run --name postgres-container --network fullstack-network -v pgdata:/var/lib/postgresql/data -d my-postgres-db
f2eeb4f27f901a4467e1296a4763f1fa5dbdb9b94bdcdb59b5d75fa2e
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/database$
```

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
```

1.

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

```
npm install express pg
```

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}`);  
});
```

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"]

O

Build the Backend Image:

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

```
cd ..
```

5.

Run the Backend Container:

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

[illegible]

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 ..
```

3.

Run the Frontend Container:

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

```
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/backend$ cd ../
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ cd frontend/
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ touch index.html
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ code index.html
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ touch Dockerfile
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ code Dockerfile
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ cat Dockerfile
FROM nginx:latest
COPY index.html /usr/share/nginx/html/index.html
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ docker build -t my-nginx-app .
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

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

Start a build
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ docker build -t my-nginx-app .
[+] Building 5.5s (7/7) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 103B                             0.0s
=> [internal] load metadata for docker.io/library/nginx:latest  5.1s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 188B                                  0.0s
=> CACHED [1/2] FROM docker.io/library/nginx:latest@sha256:67682bda769fae1ccf5183192b8daf37b6  0.0s
=> => resolve docker.io/library/nginx:latest@sha256:67682bda769fae1ccf5183192b8daf37b6:caef9c  0.0s
=> [2/2] COPY index.html /usr/share/nginx/html/index.html      0.1s
=> => exporting layers                                           0.1s
=> => writing image sha256:39fc0867bf96ec7d70178681d9f5bd1c245abe8e6783d022d9428283d423ba96  0.0s
=> => naming to docker.io/library/my-nginx-app                 0.0s
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$ docker run --name frontend-container --network fullstack-network -p 8080:80 -d my-nginx-app
25f2dbf9dfe86bf46330d1d3d36e55b9244ff6c126b8f3862994cb7f4a4af9f9
einfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/frontend$
```

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
```

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.

```

$lnfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/backend$ ls
Dockerfile index.js node_modules package.json package-lock.json
$lnfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app/backend$ docker exec -it backend-container /bin/bash
root@6222bf8f1682:/usr/src/app# sudo apt-get update
bash: sudo: command not found
root@6222bf8f1682:/usr/src/app# apt-get update
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 [9788 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 Packages [168 kB]
Fetched 9224 kB in 26s (356 kB/s)
Reading package lists... Done
root@6222bf8f1682:/usr/src/app# apt-get install -y postgresql-client
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
0 upgraded, 3 newly installed, 0 to remove and 10 not upgraded.
Need to get 1744 kB of archives.
After this operation, 8244 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 postgresql-client-common all 248 [35.1 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 postgresql-client-15 amd64 15.7-0+deb12u1 [1699 kB]
Get:3 http://deb.debian.org/debian bookworm/main amd64 postgresql-client all 15+248 [10.1 kB]
Fetched 1744 kB in 10s (169 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package postgresql-client-common.
(Reading database ... 23243 files and directories currently installed.)
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 provide /usr/share/man/man1/psql.1.gz (psql.1.gz) in auto mode
Setting up postgresql-client (15+248) ...
root@6222bf8f1682:/usr/src/app# psql -h postgres-container -U user -d mydatabase -c "SELECT NOW();"
Password for user user:
      now
-----
2024-07-11 06:35:07.42013+00
(1 row)

root@6222bf8f1682:/usr/src/app# exit

```

Part 6: Final Integration and Testing

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

Steps:

- Access the Frontend:**
 - Visit <http://localhost:8080> in your browser. You should see the Nginx welcome page with the custom HTML.
- 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 ..
```

```
inf@chips@PUNELPT0728:~/devOps-assessment/Assessment 3/fullstack-docker-app/backend$ ls
Dockerfile  index.js  node_modules  package.json  package-lock.json
inf@chips@PUNELPT0728:~/devOps-assessment/Assessment 3/fullstack-docker-app/backend$ docker exec -it backend-container /bin/bash
root@6222bf8f1682:/usr/src/app# sudo apt-get update
bash: sudo: command not found
root@6222bf8f1682:/usr/src/app# apt-get update
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 Packages [168 kB]
Fetched 9224 kB in 26s (356 kB/s)
Reading package lists... Done
root@6222bf8f1682:/usr/src/app# apt-get install -y postgresql-client
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
0 upgraded, 3 newly installed, 0 to remove and 10 not upgraded.
Need to get 1744 kB of archives.
After this operation, 8244 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bookworm/main amd64 postgresql-client-common all 248 [35.1 kB]
Get:2 http://deb.debian.org/debian bookworm/main amd64 postgresql-client-15 amd64 15.7-0+deb12u1 [1699 kB]
Get:3 http://deb.debian.org/debian bookworm/main amd64 postgresql-client all 15+248 [10.1 kB]
Fetched 1744 kB in 10s (169 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package postgresql-client-common.
(Reading database ... 23243 files and directories currently installed.)
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 provide /usr/share/man/man1/psql.1.gz (psql.1.gz) in auto mode
Setting up postgresql-client (15+248) ...
root@6222bf8f1682:/usr/src/app# psql -h postgres-container -U user -d mydatabase -c "SELECT NOW();"
Password for user user:
      now
-----
2024-07-11 06:35:07.42013+00
(1 row)
```

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
```

```
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker stop frontend-container backend-container postgres-container
frontend-container
backend-container
postgres-container
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker rm frontend-container backend-container postgres-container
frontend-container
backend-container
postgres-container
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker rmi my-nginx-app my-node-app my-postgres-db
Untagged: my-nginx-app:latest
Deleted: sha256:3ced0f6fc975f87b96333c06d4cb30a80c23a7a907b14f39e94d6fcb0ff522492
Untagged: my-node-app:latest
Deleted: sha256:5603111d74d62571d2473c4862e0be32b20588efcb9afef577ec9c5f272c9cbd
Untagged: my-postgres-db:latest
Deleted: sha256:1cfe6ff8722f48bf2889ab46a727e0e5ed0bf4046b570b920262d21e57ca2275
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker network rm fullstack-network
fullstack-network
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker volume rm pgdata
pgdata
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker image

Usage: docker image COMMAND

Manage images

Commands:
  build      Build an image from a Dockerfile
  history    Show the history of an image
  import     Import the contents from a tarball to create a filesystem image
  inspect    Display detailed information on one or more images
  load       Load an image from a tar archive or STDIN
  ls         List images
  prune      Remove unused images
  pull       Download an image from a registry
  push       Upload an image to a registry
  rm         Remove one or more images
  save       Save one or more images to a tar archive (streamed to STDOUT by default)
  tag        Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
<none>        <none>    B9fc0867bf9e   2 hours ago   188MB
ubuntu        latest    35a88802559d   4 weeks ago   78.1MB
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker rmi 39fc0867bf9e
Deleted: sha256:39fc0867bf9ec7d78178681d9f5bd1c245abe8e6783d022d9428283d423ba96
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    35a88802559d   4 weeks ago   78.1MB
minfochips@PUNELPT0728:~/DevOps-assessment/Assessment 3/fullstack-docker-app$
```