www.linkedin.com/in/arun-kumar-akula

Dockerized & Deployed Food-App on AWS EC2 Using Docker.!

Overview :

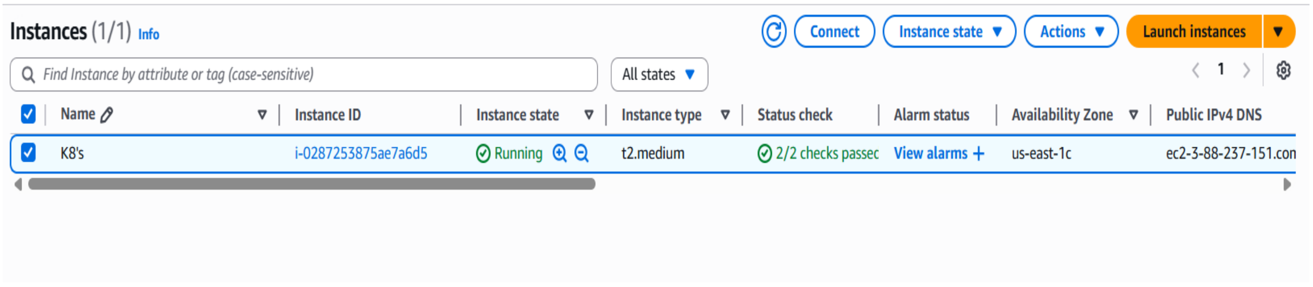
This task involves cloning a web application from GitHub and containerizing it using Docker. The process includes exploring the docker commands & creating a Dockerfile following best practices, and building a lightweight image. The application is then deployed on an AWS EC2 instance and made accessible via the browser.

Step-by-Step Guide:

- Start by launching a new EC2 instance from the AWS Management Console:

Step 1: Launch EC2 Instance

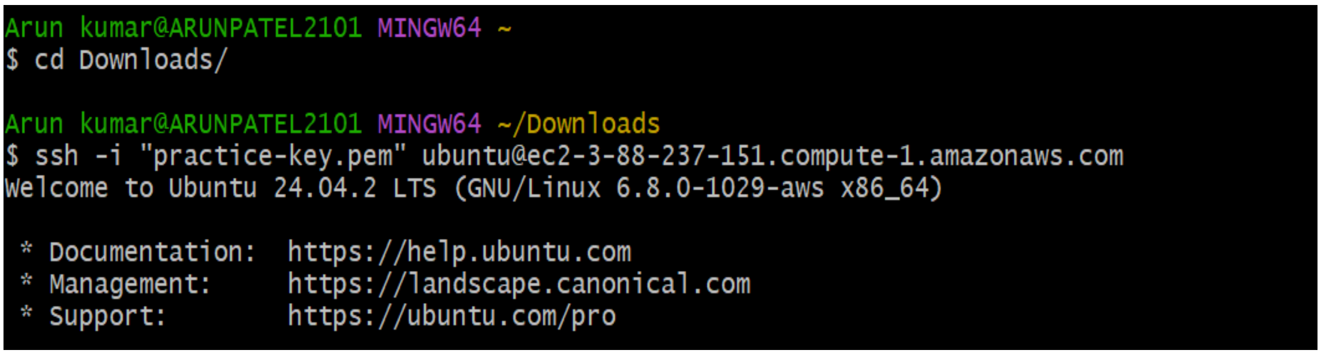
- Go to the AWS EC2 console.
- Select the instance type(t2.micro).
- Launch a new Ubuntu or linux server.
- Allow ports 22 (SSH) and 80 (HTTP) in the security group.
- Click on launch instance



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
K8's	i-0287253875ae7a6d5	Running	t2.medium	2/2 checks passec	View alarms +	us-east-1c	ec2-3-88-237-151.con

Step 2: Connect to your EC2 instance using SSH

```
ssh -i "your-key.pem" ubuntu@<EC2-Public-IP>
```

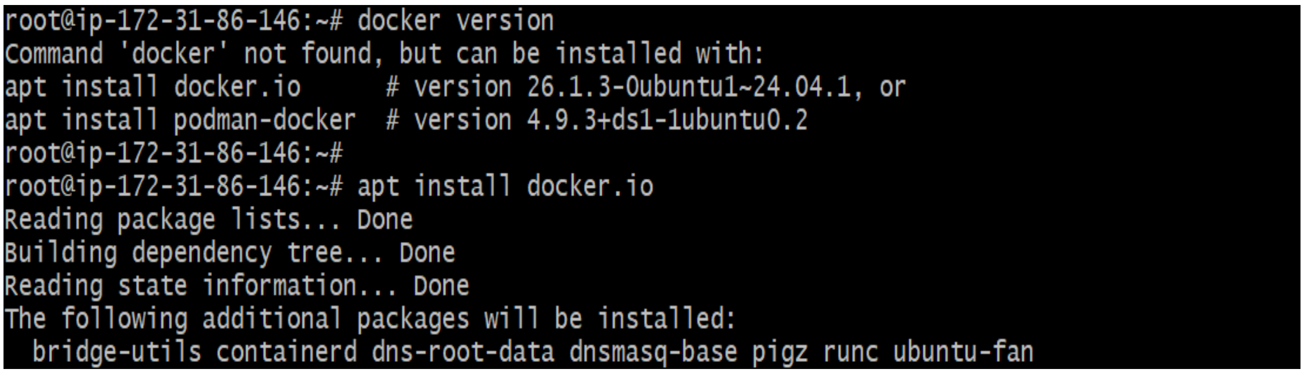


```
Arun kumar@ARUNPATEL2101 MINGW64 ~
$ cd Downloads/

Arun kumar@ARUNPATEL2101 MINGW64 ~/Downloads
$ ssh -i "practice-key.pem" ubuntu@ec2-3-88-237-151.compute-1.amazonaws.com
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro
```

Step 3: Install Docker on the Server

- Once logged into the EC2 instance, install Docker:

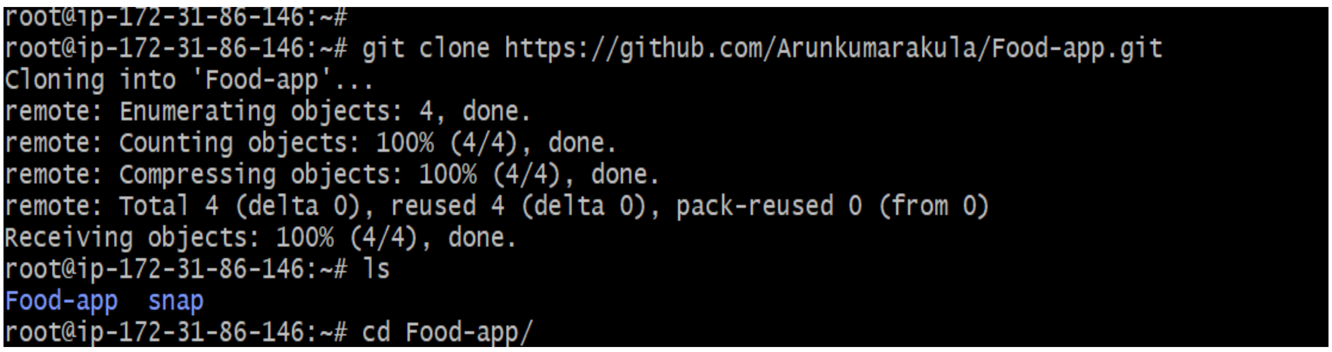
```
sudo apt update

sudo apt install -y docker.io

docker version
```



```
root@ip-172-31-86-146:~# docker version
Command 'docker' not found, but can be installed with:
apt install docker.io      # version 26.1.3-0ubuntu1~24.04.1, or
apt install podman-docker  # version 4.9.3+ds1-1ubuntu0.2
root@ip-172-31-86-146:~#
root@ip-172-31-86-146:~# apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
```

Step 4: Clone the GitHub Repository

```
git clone https://github.com/Arunkumarakula/Food-app.git

cd Food-app
```



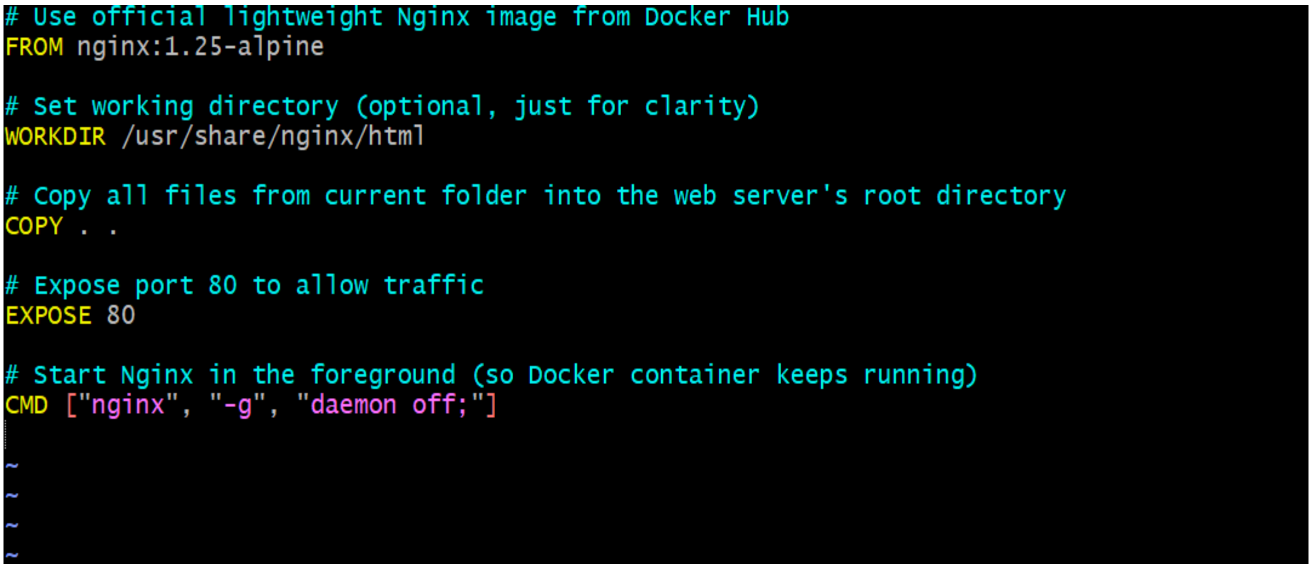
```
root@ip-172-31-86-146:~#
root@ip-172-31-86-146:~# git clone https://github.com/Arunkumarakula/Food-app.git
Cloning into 'Food-app'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.
root@ip-172-31-86-146:~# ls
Food-app  snap
root@ip-172-31-86-146:~# cd Food-app/
```

Step 5: Create a Dockerfile

A Dockerfile is a text file that contains a set of instructions to build a Docker image. It defines everything needed to set up your application inside a container like the base image, files to copy, commands to run, and exposed ports.

- Command to create a dockerfile **vi Dockerfile**

```
# Use official lightweight Nginx image from Docker Hub

FROM nginx:1.25-alpine

# Set working directory (optional, just for clarity)

WORKDIR /usr/share/nginx/html

# Copy all files from current folder into the web server's root directory

COPY . .

# Expose port 80 to allow traffic

EXPOSE 80

# Start Nginx in the foreground (so Docker container keeps running)

CMD ["nginx", "-g", "daemon off;"]
```

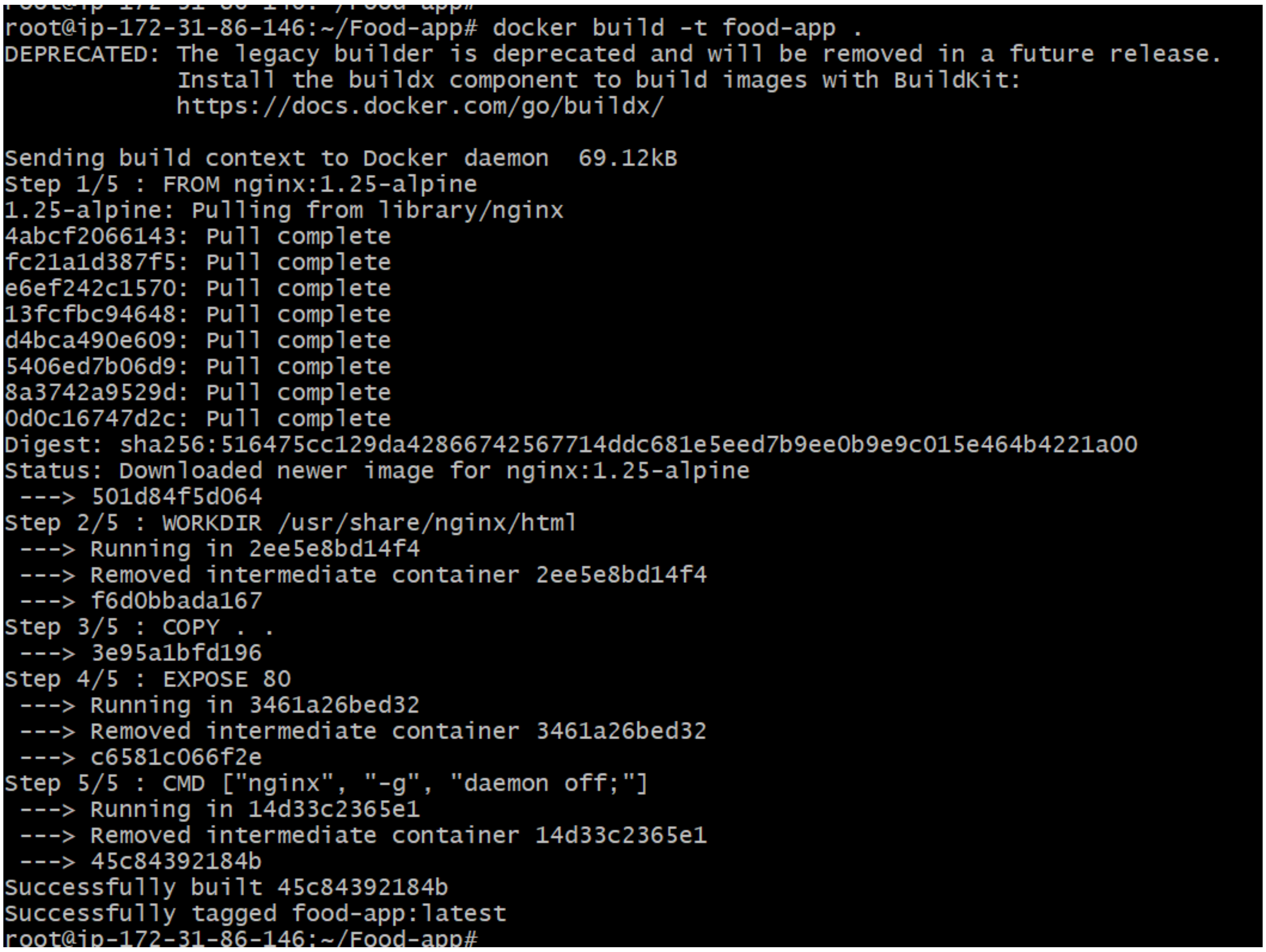


```
# Use official lightweight Nginx image from Docker Hub
FROM nginx:1.25-alpine

# Set working directory (optional, just for clarity)
WORKDIR /usr/share/nginx/html

# Copy all files from current folder into the web server's root directory
COPY . .

# Expose port 80 to allow traffic
EXPOSE 80

# Start Nginx in the foreground (so Docker container keeps running)
CMD ["nginx", "-g", "daemon off;"]

~
~
~
~
```

- Close file press (shift : wq!) it will save the file.

Step 6: Build the Docker Image

- Run the following command inside the project directory
  docker build -t food-app .
- This command Builds your Docker image from the Dockerfile and names it food-app.

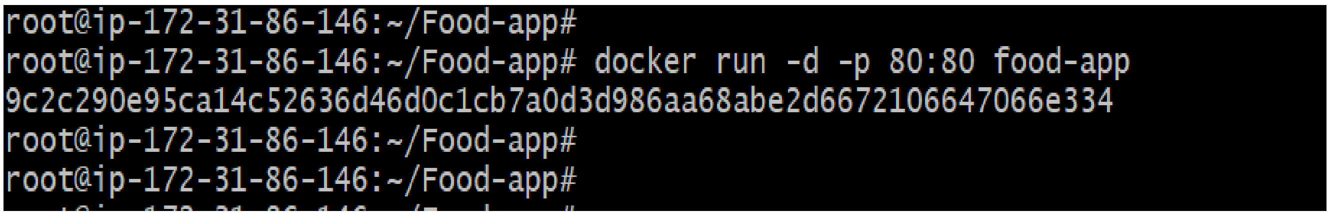


```
root@ip-172-31-86-146:~/Food-app# docker build -t food-app .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  69.12kB
Step 1/5 : FROM nginx:1.25-alpine
1.25-alpine: Pulling from library/nginx
4abcf2066143: Pull complete
fc21a1d387f5: Pull complete
e6ef242c1570: Pull complete
13fcfbc94648: Pull complete
d4bca490e609: Pull complete
5406ed7b06d9: Pull complete
8a3742a9529d: Pull complete
0d0c16747d2c: Pull complete
Digest: sha256:516475cc129da42866742567714ddc681e5eed7b9ee0b9e9c015e464b4221a00
Status: Downloaded newer image for nginx:1.25-alpine
 ---> 501d84f5d064
Step 2/5 : WORKDIR /usr/share/nginx/html
 ---> Running in 2ee5e8bd14f4
 ---> Removed intermediate container 2ee5e8bd14f4
 ---> f6d0bbada167
Step 3/5 : COPY . .
 ---> 3e95a1bfd196
Step 4/5 : EXPOSE 80
 ---> Running in 3461a26bed32
 ---> Removed intermediate container 3461a26bed32
 ---> c6581c066f2e
Step 5/5 : CMD ["nginx", "-g", "daemon off;"]
 ---> Running in 14d33c2365e1
 ---> Removed intermediate container 14d33c2365e1
 ---> 45c84392184b
Successfully built 45c84392184b
Successfully tagged food-app:latest
root@ip-172-31-86-146:~/Food-app#
```

Step 7: Run the Docker Container

- This command starts a container that serves your web app and makes it accessible publicly via port 80 on your EC2.

```
docker run -d -p 80:80 food-app
```



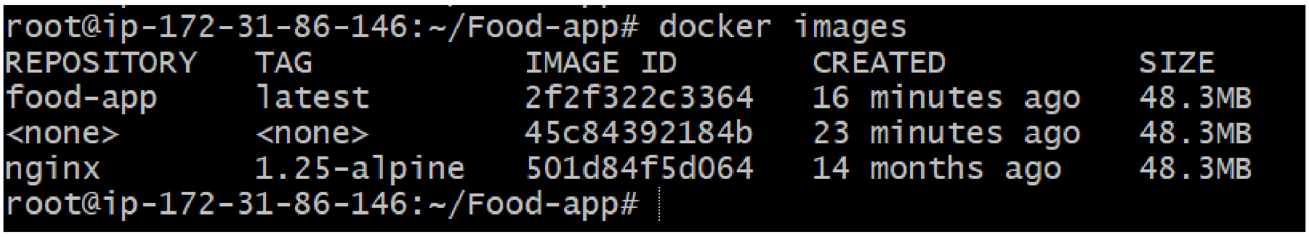
```
root@ip-172-31-86-146:~/Food-app#
root@ip-172-31-86-146:~/Food-app# docker run -d -p 80:80 food-app
9c2c290e95ca14c52636d46d0c1cb7a0d3d986aa68abe2d6672106647066e334
root@ip-172-31-86-146:~/Food-app#
root@ip-172-31-86-146:~/Food-app#
```

- Now The application is running inside a Docker container on port 80.

Step 8: Verify Image & Containers

- Check all Docker images.

```
docker images
```

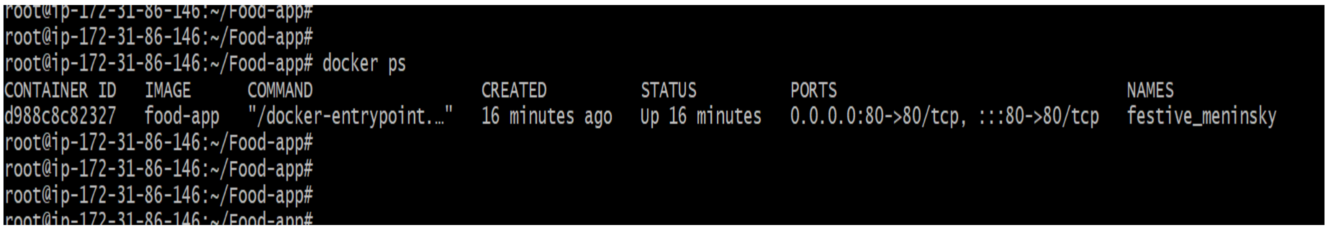


```
root@ip-172-31-86-146:~/Food-app# docker images
REPOSITORY    TAG             IMAGE ID       CREATED          SIZE
food-app      latest          2f2f322c3364   16 minutes ago   48.3MB
<none>        <none>          45c84392184b   23 minutes ago   48.3MB
nginx         1.25-alpine     501d84f5d064   14 months ago    48.3MB
root@ip-172-31-86-146:~/Food-app#
```

Docker image food-app was built successfully with a final size of **48.3MB** using dockerfile build.

- **Check running containers:**

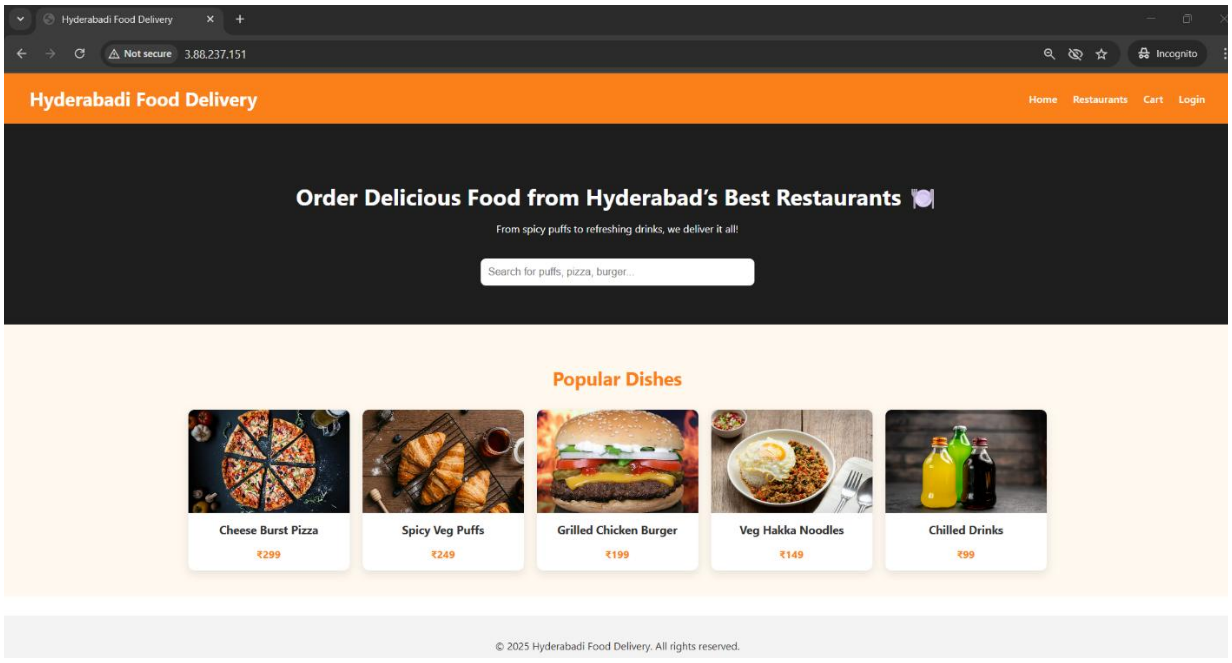
```
docker ps
```



```
root@ip-172-31-86-146:~/Food-app#
root@ip-172-31-86-146:~/Food-app#
root@ip-172-31-86-146:~/Food-app# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS                               NAMES
d988c8c82327   food-app   "/docker-entrypoint.…"   16 minutes ago   Up 16 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp   festive_meninsky
root@ip-172-31-86-146:~/Food-app#
root@ip-172-31-86-146:~/Food-app#
root@ip-172-31-86-146:~/Food-app#
root@ip-172-31-86-146:~/Food-app#
```

Docker container food-app is running successfully and serving on port 80, accessible via the EC2 public IP.

Step 9 : Open your browser and visit **http://<EC2-Public-IP>** to access the running application.



Hyderabadi Food Delivery

Order Delicious Food from Hyderabad's Best Restaurants 🍽️

From spicy puffs to refreshing drinks, we deliver it all!

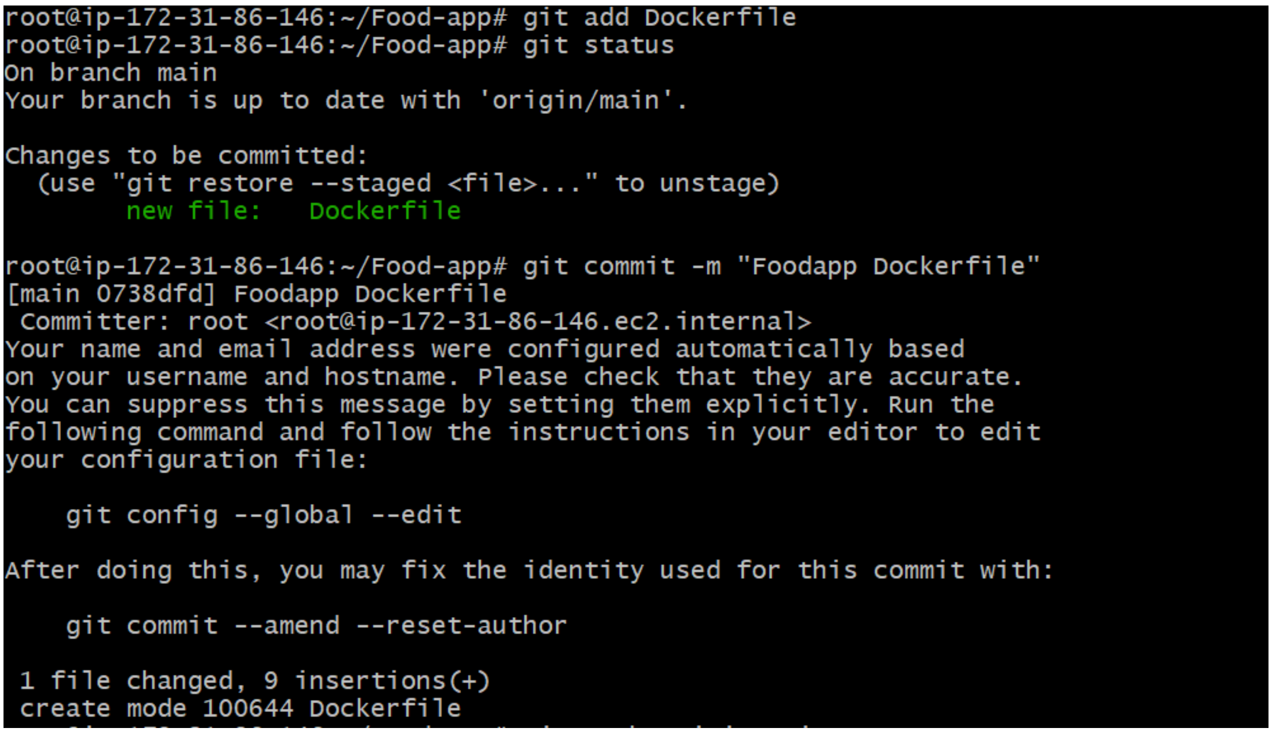
Popular Dishes

Cheese Burst Pizza	₹299
Spicy Veg Puffs	₹249
Grilled Chicken Burger	₹199
Veg Hakka Noodles	₹149
Chilled Drinks	₹99

© 2025 Hyderabadi Food Delivery. All rights reserved.

Step 10: Push Dockerfile to GitHub (optinal)

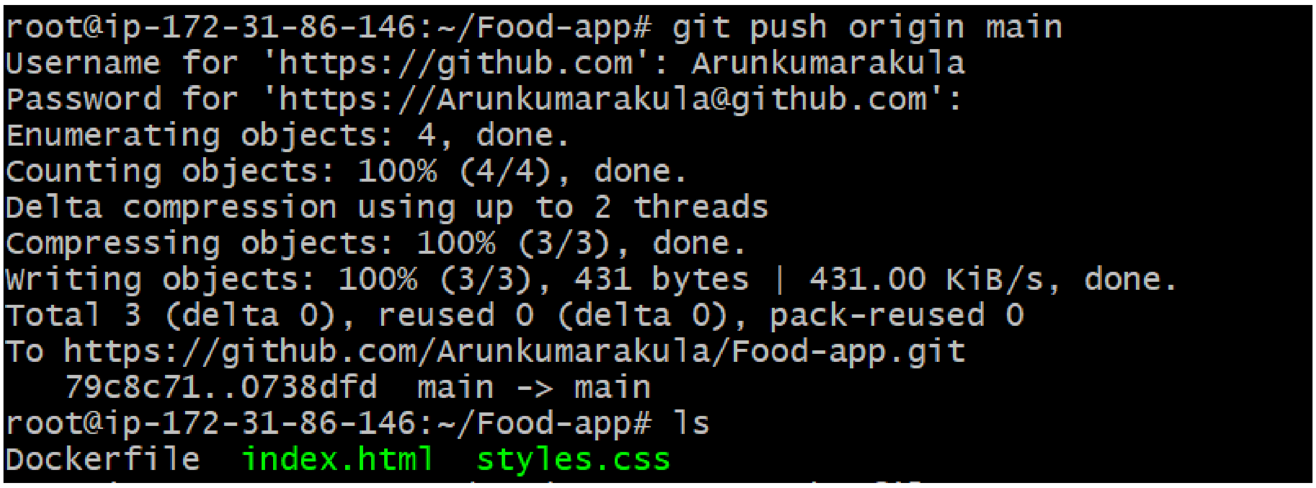
```
git add Dockerfile

git commit -m "Food-app Dockerfile"
```



```
root@ip-172-31-86-146:~/Food-app# git add Dockerfile
root@ip-172-31-86-146:~/Food-app# git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Dockerfile

root@ip-172-31-86-146:~/Food-app# git commit -m "Foodapp Dockerfile"
[main 0738dfd] Foodapp Dockerfile
 Committer: root <root@ip-172-31-86-146.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 9 insertions(+)
 create mode 100644 Dockerfile
```

- **Push the docker file**

```
git push origin main
```



```
root@ip-172-31-86-146:~/Food-app# git push origin main
Username for 'https://github.com': Arunkumarakula
Password for 'https://Arunkumarakula@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 431 bytes | 431.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Arunkumarakula/Food-app.git
   79c8c71..0738dfd  main -> main
root@ip-172-31-86-146:~/Food-app# ls
Dockerfile  index.html  styles.css
```

- GitHub requires a Personal Access Token (PAT) for secure authentication when performing certain actions over HTTPS, especially from the command line or scripts (like git push, git clone), after generating pass the username and password .