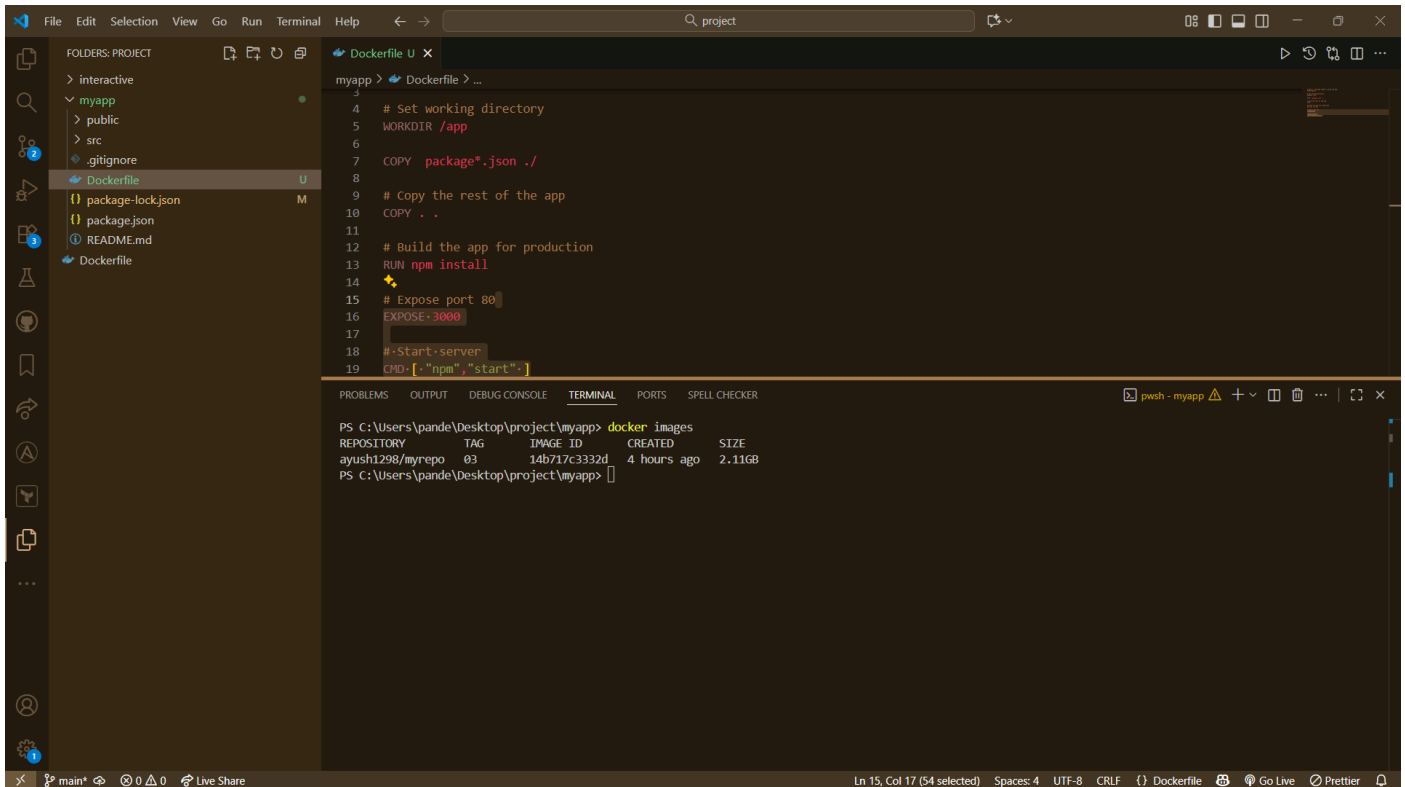


# Steps to Pull/Push react app Image from Repository in Docker hub and Run it on Remote Ubuntu Server

## ⚙️ Step-by-Step Setup

### 1. Create a Image of React app : -

Docker Image: -



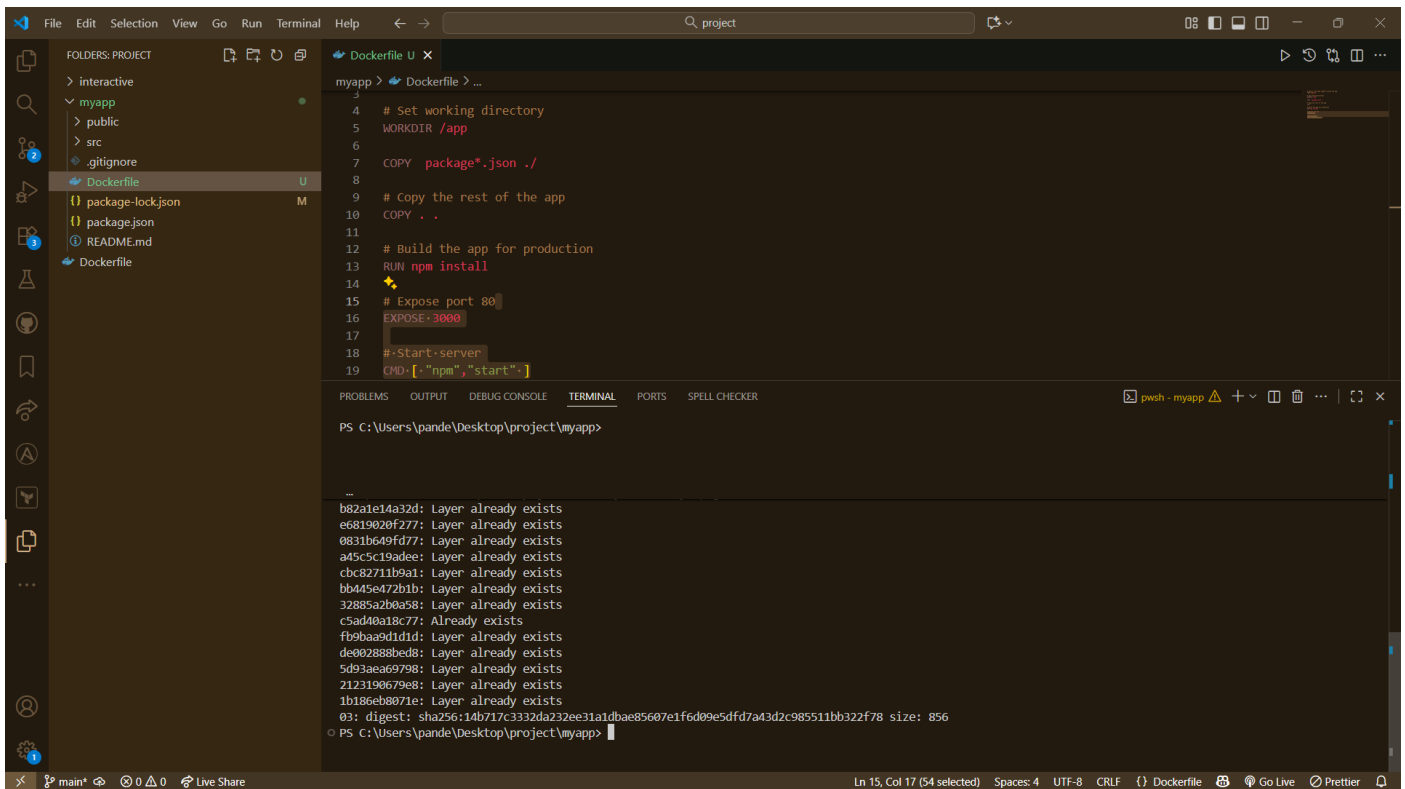
The screenshot shows a Visual Studio Code editor with a Dockerfile open. The Dockerfile contains the following instructions:

```
3
4 # Set working directory
5 WORKDIR /app
6
7 COPY package*.json ./
8
9 # Copy the rest of the app
10 COPY . .
11
12 # Build the app for production
13 RUN npm install
14
15 # Expose port 80
16 EXPOSE 3000
17
18 # Start server
19 CMD ["npm", "start"]
```

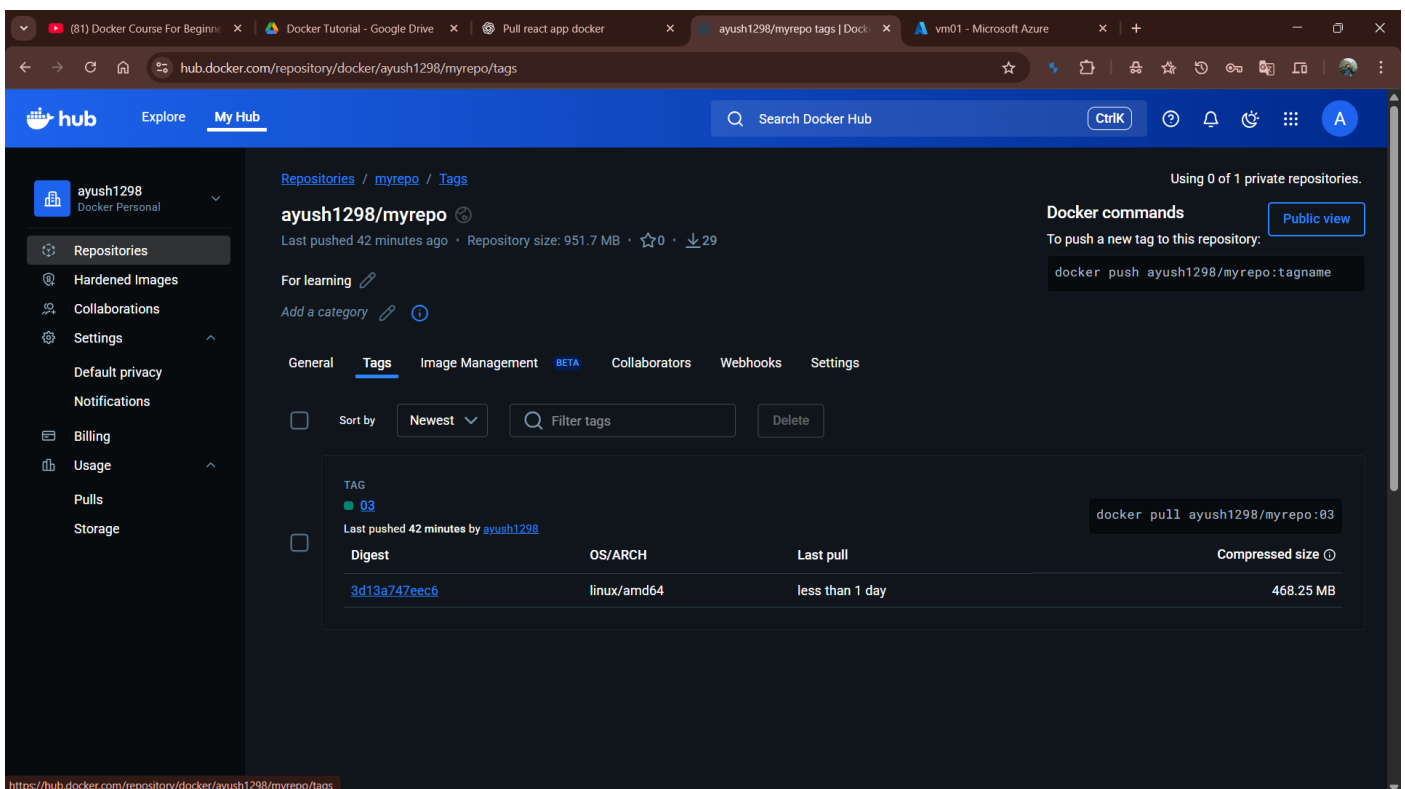
The terminal at the bottom shows the command `docker images` being executed, resulting in the following output:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ayush1298/myrepo	03	14b717c3332d	4 hours ago	2.11GB

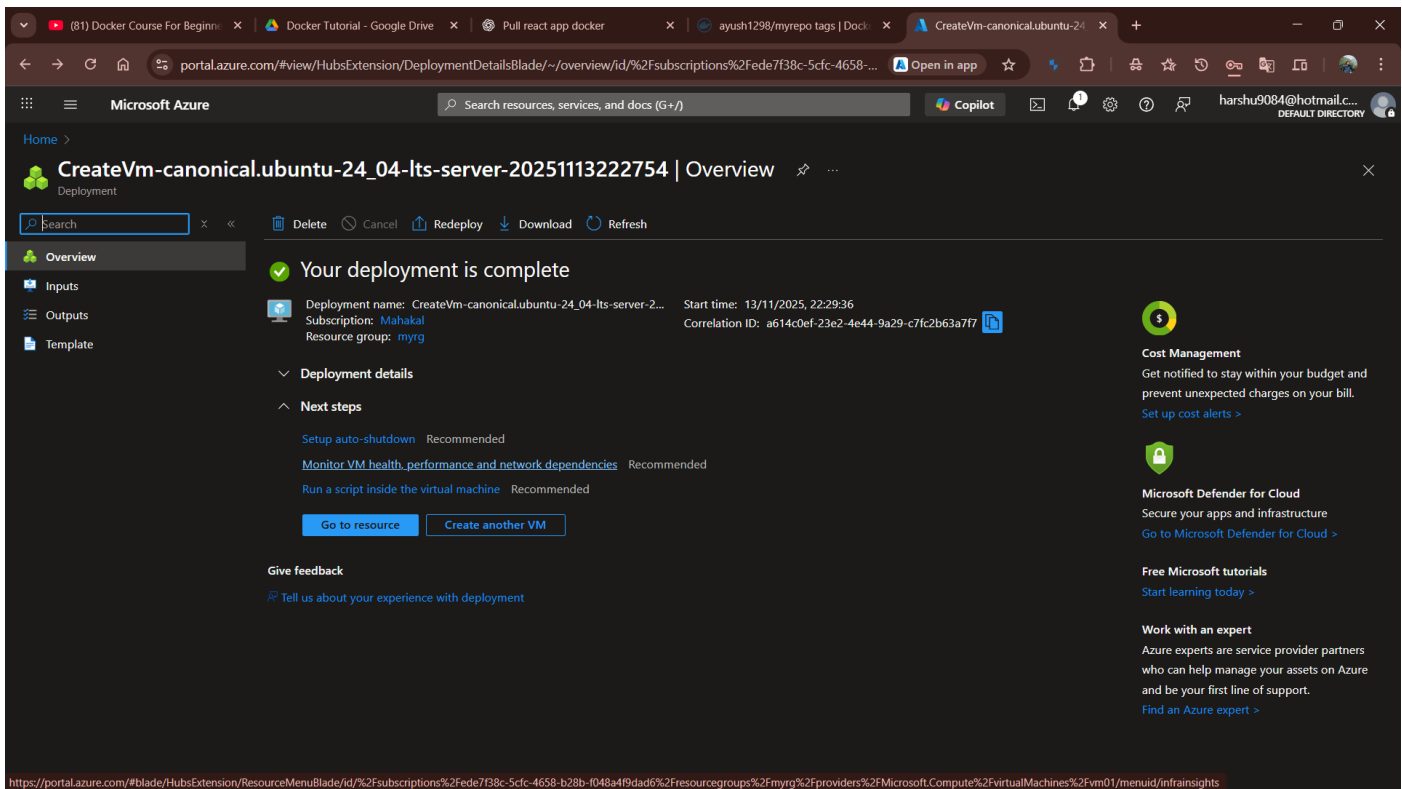
### 2. Login on DockerHub And push this Image on DockerHub in your repo: -



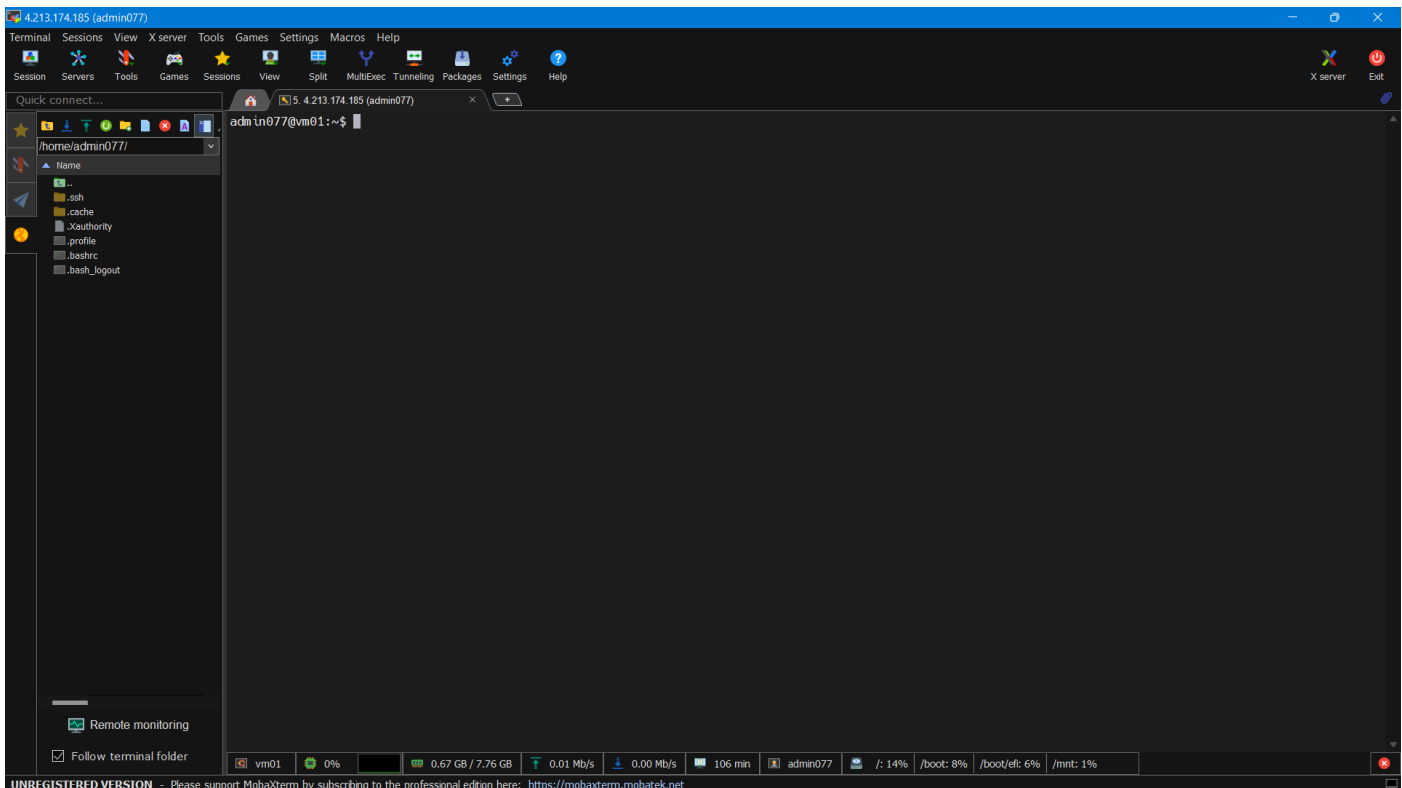
**Docker Image Name: - ayush1298/myrepo:03**



### 3. Create Ubuntu Server on Azure: -



Login: -



Steps to Install Docker in Remote Server: -

## 2. Update the system

```
bash
```

```
sudo apt update && sudo apt upgrade -y
```

## 3. Install Docker (if not already installed)

```
bash
```

```
sudo apt install docker.io -y
```

Enable and start Docker:

```
bash
```

```
sudo systemctl enable docker
```

```
sudo systemctl start docker
```

Verify installation:

```
bash
```

```
docker --version
```



### Enable and start Docker:

```
bash  
  
sudo systemctl enable docker  
sudo systemctl start docker
```

### Verify installation:

```
bash  
  
docker --version
```

## 4. Login to Docker Hub

If your image is private, login with your credentials:

```
bash  
  
sudo docker login
```

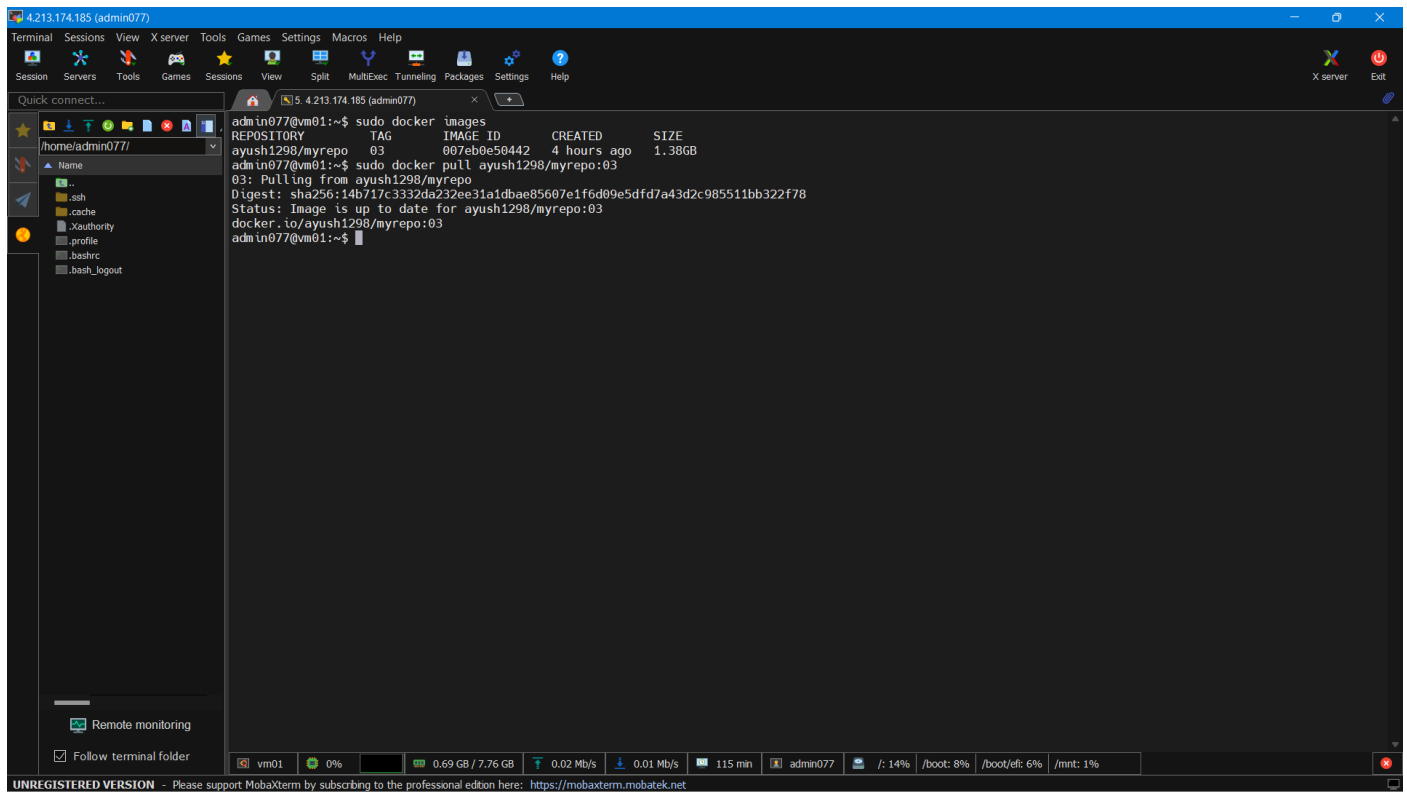


(Enter your Docker Hub username and password or token)

If it is a public image, you can skip this step.

### 4. Pull your React app image from Docker Hub: -

**Command:** - sudo docker pull ayush1298/myrepo:03



## 5. Run your React app container

Run the container and map ports:

**Command:** - `sudo docker run -d -p 80:3000 ayush1298/myrepo:03`

## 6. Verify the container is running

**Command:** - `sudo docker ps`

```
admin077@vm01:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ayush1298/myrepo    03                 007eb0e50442       4 hours ago        1.38GB
admin077@vm01:~$ sudo docker pull ayush1298/myrepo:03
03: Pulling from ayush1298/myrepo
Digest: sha256:14b717c3332da232ee31a1dbae85607e1fed09e5dfd7a43d2c985511bb322f78
Status: Image is up to date for ayush1298/myrepo:03
docker.io/ayush1298/myrepo:03
admin077@vm01:~$ sudo docker run -d -p 80:3000 ayush1298/myrepo:03
22783e310898bc024a00f07c59edc6de69797fc70f9f31c218ffbf9a99cd477ad
admin077@vm01:~$ sudo docker ps
sudo: command not found
admin077@vm01:~$ sudo docker ps
Command 'sudo' not found, did you mean:
  command 'sudo' from deb sudo (1.9.15p5-3ubuntu5.24.04.1)
  command 'sudo' from deb sudo-ldap (1.9.15p5-3ubuntu5.24.04.1)
Try: sudo apt install <deb name>
admin077@vm01:~$ sudo docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                               NAMES
22783e310898   ayush1298/myrepo:03   "docker-entrypoint.s..." About a minute ago Up About a minute   0.0.0.0:80->3000/tcp, [::]:80->3000/tcp   focused_bartik
admin077@vm01:~$
```

You should see your container listed.

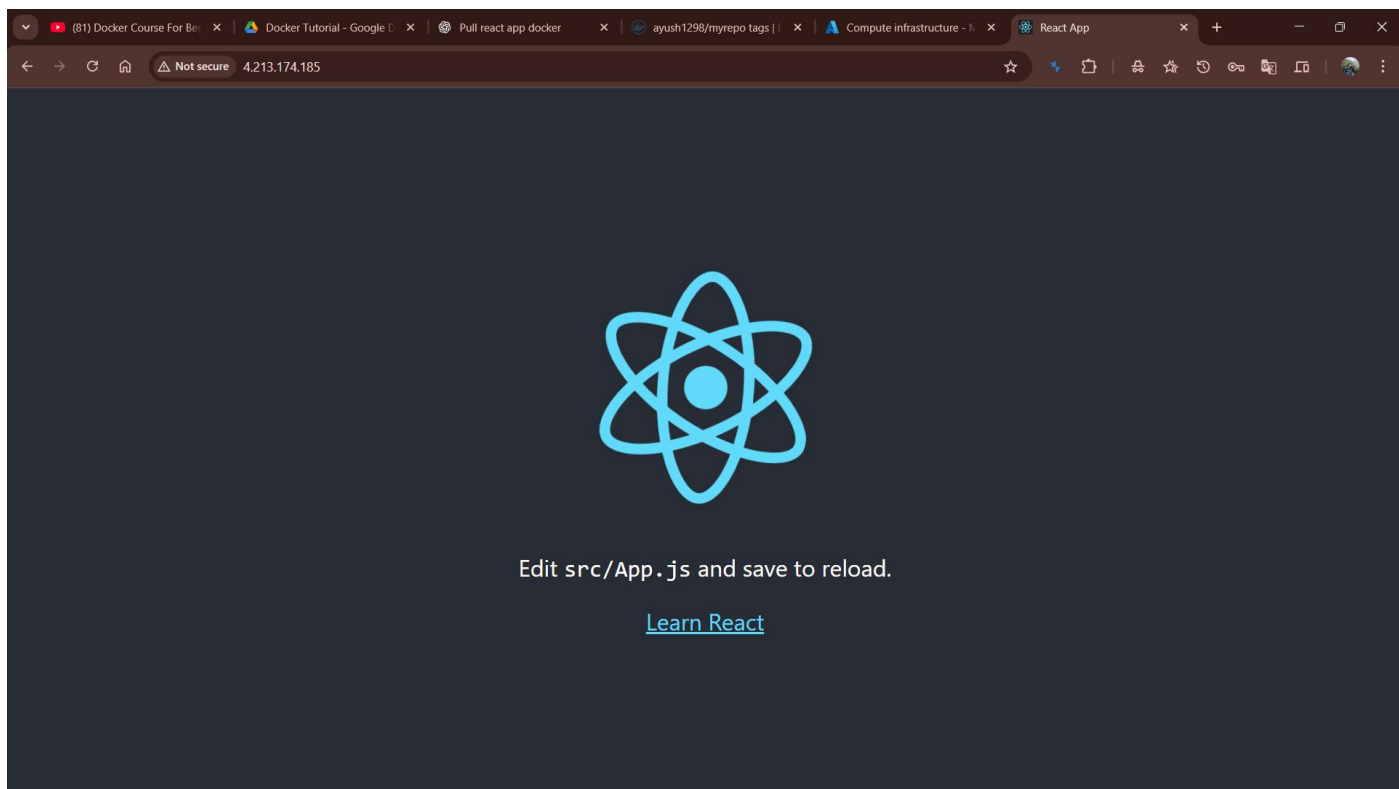
## 7. Open port 80 in Azure (important)

Go to your **Azure Portal** → **Your VM** → **Networking** → **Inbound Port Rules**  
Add a rule to **Allow TCP on port 80**.

## 8. Test the app in your browser

<http://<your-vm-public-ip>>

**Output: -**



React app running 🎉

---