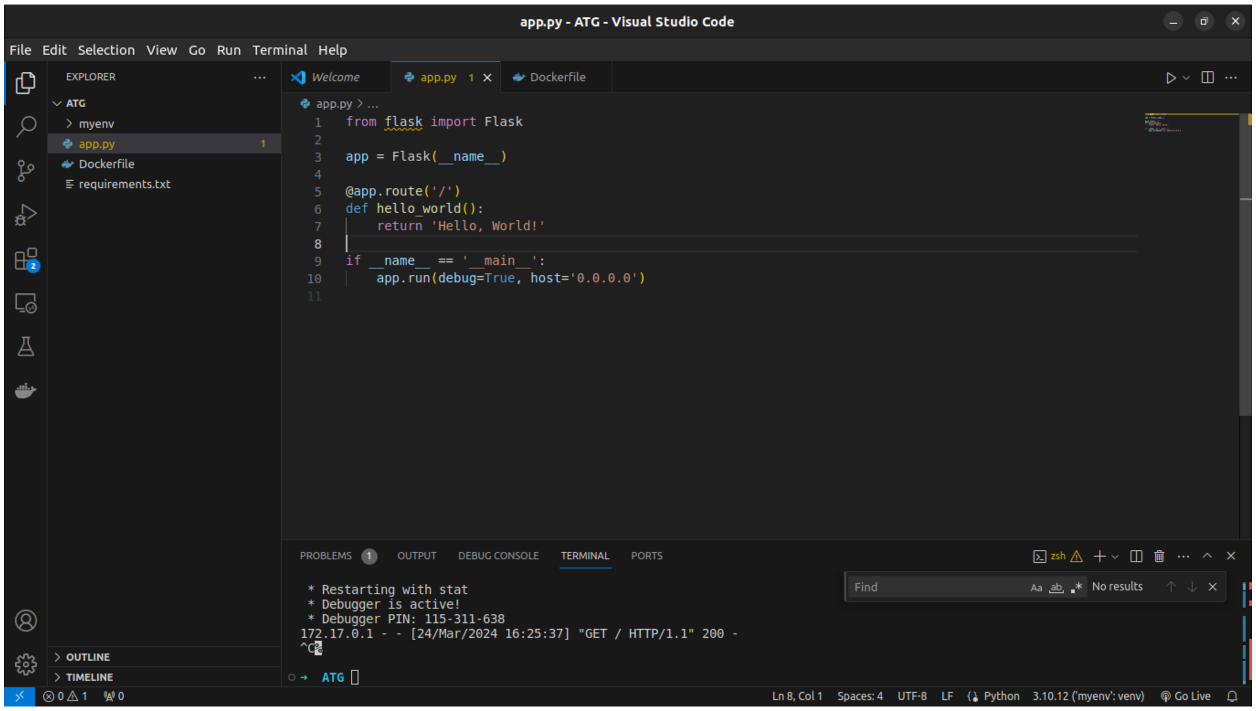
Assignment

Python app with hello World configuration


app.py - ATG - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
ATG
myenv
app.py
Dockerfile
requirements.txt
Welcome
app.py
Dockerfile
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Find
No results
OUTLINE
TIMELINE
Ln 8, Col 1 Spaces: 4 UTF-8 LF Python 3.10.12 ('myenv': venv) Go Live

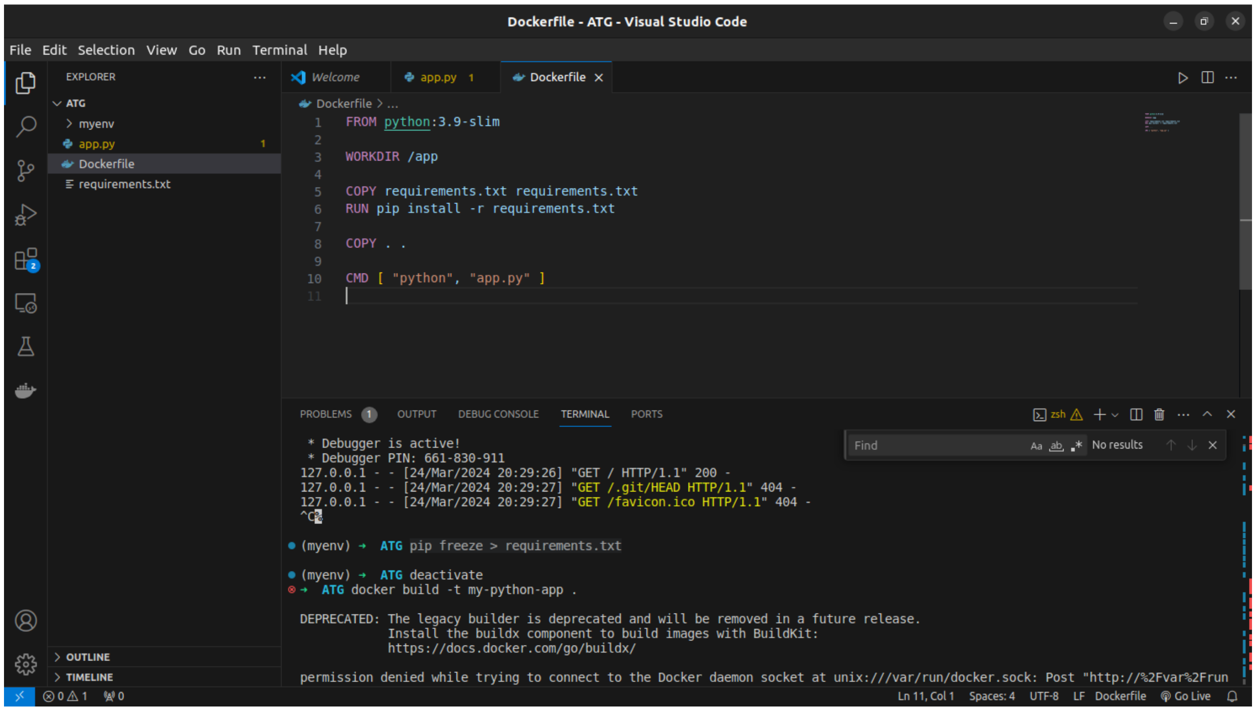
```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

```
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 115-311-638
172.17.0.1 - - [24/Mar/2024 16:25:37] "GET / HTTP/1.1" 200 -
^C
ATG
```

```
python3 -m venv myenv // virtual Enviorment

source myenv/bin/activate // activate

pip install flask // flask install

pip freeze > requirements.txt // all requires dependicies
```

Docker file for building Docker Images

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt

COPY . .

CMD [ "python", "app.py" ]
```

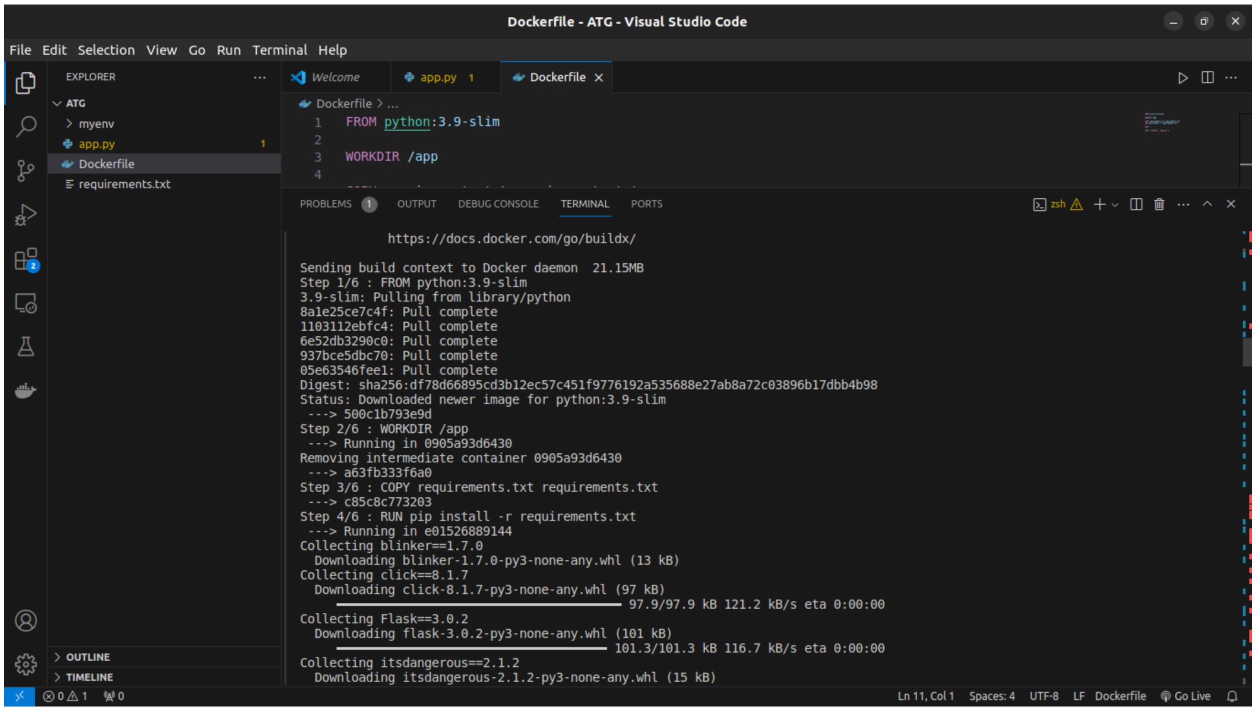

Dockerfile - ATG - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
ATG
myenv
app.py
Dockerfile
requirements.txt
Welcome
app.py
Dockerfile
OUTLINE
TIMELINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Ln 11, Col 1 Spaces: 4 UTF-8 LF Dockerfile Go Live

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt

COPY . .

CMD [ "python", "app.py" ]
```

```
 * Debugger is active!
 * Debugger PIN: 661-830-911
127.0.0.1 - - [24/Mar/2024 20:29:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Mar/2024 20:29:27] "GET /.git/HEAD HTTP/1.1" 404 -
127.0.0.1 - - [24/Mar/2024 20:29:27] "GET /favicon.ico HTTP/1.1" 404 -
^C%

(myenv) → ATG pip freeze > requirements.txt

(myenv) → ATG deactivate
→ ATG docker build -t my-python-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/

permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun
```

command to build it

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

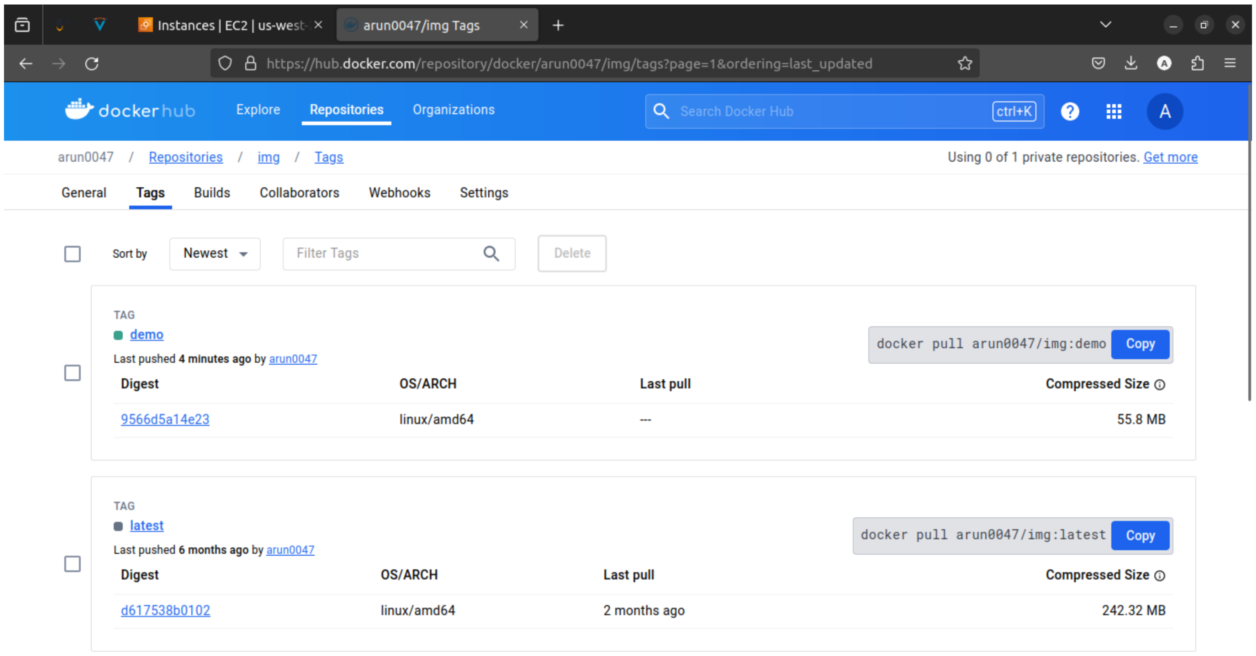


```
FROM python:3.9-slim

WORKDIR /app
```

```
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  21.15MB
Step 1/6 : FROM python:3.9-slim
3.9-slim: Pulling from library/python
8a1e25ce7c4f: Pull complete
1103112ebfc4: Pull complete
6e52db3290c0: Pull complete
937bce5dbc70: Pull complete
05e63546fee1: Pull complete
Digest: sha256:df78d66895cd3b12ec57c451f9776192a535688e27ab8a72c03896b17dbb4b98
Status: Downloaded newer image for python:3.9-slim
 ---> 500c1b793e9d
Step 2/6 : WORKDIR /app
 ---> Running in 0905a93d6430
Removing intermediate container 0905a93d6430
 ---> a63fb333f6a0
Step 3/6 : COPY requirements.txt requirements.txt
 ---> c85c8c773203
Step 4/6 : RUN pip install -r requirements.txt
 ---> Running in e01526889144
Collecting blinker==1.7.0
  Downloading blinker-1.7.0-py3-none-any.whl (13 kB)
Collecting click==8.1.7
  Downloading click-8.1.7-py3-none-any.whl (97 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 97.9/97.9 kB 121.2 kB/s eta 0:00:00
Collecting Flask==3.0.2
  Downloading flask-3.0.2-py3-none-any.whl (101 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.3/101.3 kB 116.7 kB/s eta 0:00:00
Collecting itsdangerous==2.1.2
  Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
```

After building the image push it to docker hub so that we can download from docker hub on ec2 sever

```
sudo docker tag my-python-app arun0047/img:demo # images

sudo docker push arun0047/img:demo # pushing to docker hub
```


Instances | EC2 | us-west-
arun0047/img Tags
https://hub.docker.com/repository/docker/arun0047/img/tags?page=1&ordering=last_updated
docker hub
Explore
Repositories
Organizations
Search Docker Hub
ctrl+K
A
arun0047 / Repositories / img / Tags
Using 0 of 1 private repositories. Get more
General
Tags
Builds
Collaborators
Webhooks
Settings
Sort by
Newest
Filter Tags
Delete

TAG
demo
Last pushed **4 minutes ago** by arun0047

```
docker pull arun0047/img:demo
```

Copy

Digest	OS/ARCH	Last pull	Compressed Size
9566d5a14e23	linux/amd64	---	55.8 MB

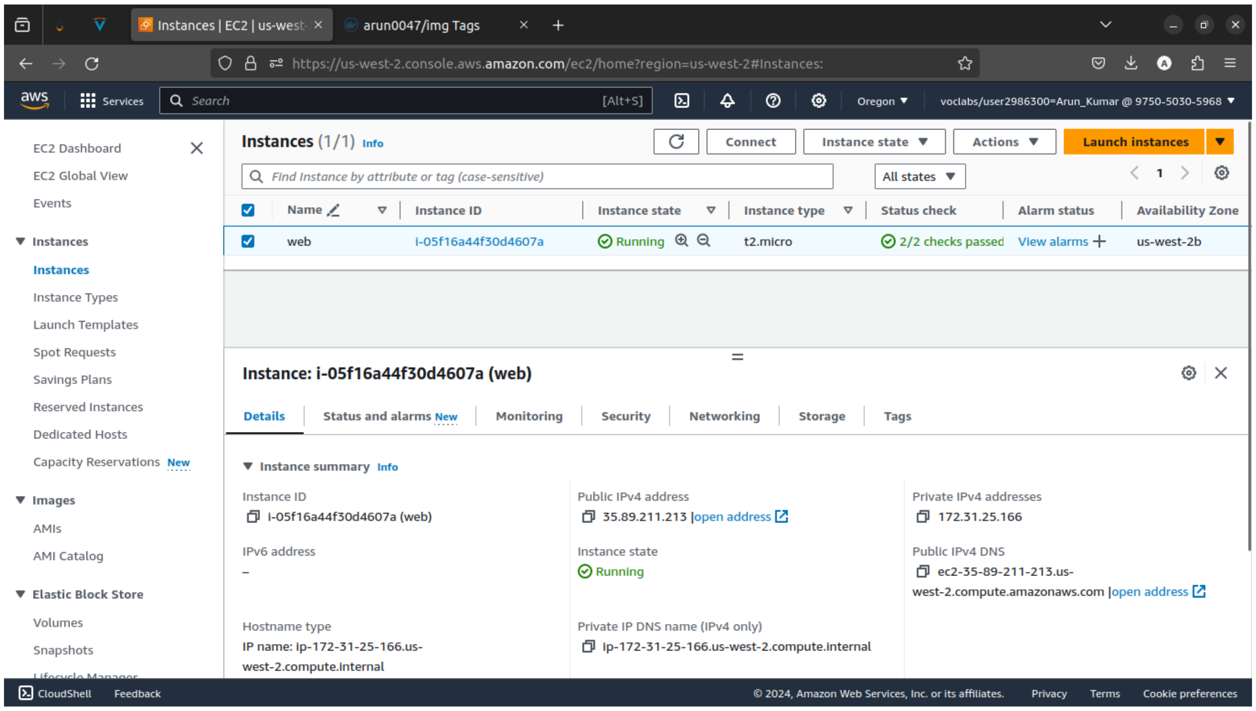
TAG
latest
Last pushed **6 months ago** by arun0047

```
docker pull arun0047/img:latest
```

Copy

Digest	OS/ARCH	Last pull	Compressed Size
d617538b0102	linux/amd64	2 months ago	242.32 MB

AWS Ec2 server creation



Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
web	i-05f16a44f30d4607a	Running	t2.micro	2/2 checks passed	View alarms +	us-west-2b

Instance: i-05f16a44f30d4607a (web)

Instance ID i-05f16a44f30d4607a (web)	Public IPv4 address 35.89.211.213 |open address	Private IPv4 addresses 172.31.25.166
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-35-89-211-213.us-west-2.compute.amazonaws.com |open address
Hostname type IP name: ip-172-31-25-166.us-west-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-25-166.us-west-2.compute.internal	

public ip : 35.89.211.213

private ip : 172.31.25.166

used key pair to connect to the instance using ssh commad

```
ssh -i key.pem ubuntu@35.89.211.213
```

To run an EC2 instance, you'll need to perform the following steps:

1. **Sign in to the AWS Management Console**: Go to the AWS Management Console at https://console.aws.amazon.com and sign in to your AWS account.
2. **Navigate to the EC2 Dashboard**: Once you're logged in, navigate to the EC2 dashboard by searching for "EC2" in the AWS services search bar or by clicking on "Services" and selecting "EC2" from the list.
3. **Launch Instance**: On the EC2 dashboard, click on the "Launch Instance" button to start the instance creation wizard.
4. **Choose an Amazon Machine Image (AMI)**: Select an AMI from the provided list. An AMI is a pre-configured template for your instance. You can choose from various operating systems and configurations.
5. **Choose an Instance Type**: Select an instance type based on your requirements. Instance types vary in terms of CPU, memory, storage, and networking capacity.
6. **Configure Instance**: Configure instance details such as the number of instances, network settings, subnet, IAM role, etc. You can leave most of the options as default if you're unsure.
7. **Add Storage**: Specify the size and type of storage for your instance. You can add additional EBS volumes if needed.
8. **Add Tags** (Optional): Add tags to your instance for easier identification and management.

9. **Configure Security Group**: Configure security group settings to control inbound and outbound traffic to your instance. At a minimum, you'll need to allow SSH (port 22) access for remote administration.
10. **Review and Launch**: Review your instance configuration and click on the "Launch" button if everything looks good.
11. **Select Key Pair**: Choose an existing key pair or create a new one. Key pairs are used for SSH authentication to connect to your instance securely.
12. **Launch Instance**: Click on the "Launch Instances" button to start the instance.

Once you've launched the instance, you can monitor its status on the EC2 dashboard. Once the instance is in the "running" state, you can connect to it using SSH or other remote access methods.

1. **Update the Package Repository**: Update the package repository to ensure you're installing the latest version of Docker and its dependencies.

   ```
   sudo apt update
   ```

2. **Install Docker Engine**: Install Docker Engine using the following command:

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

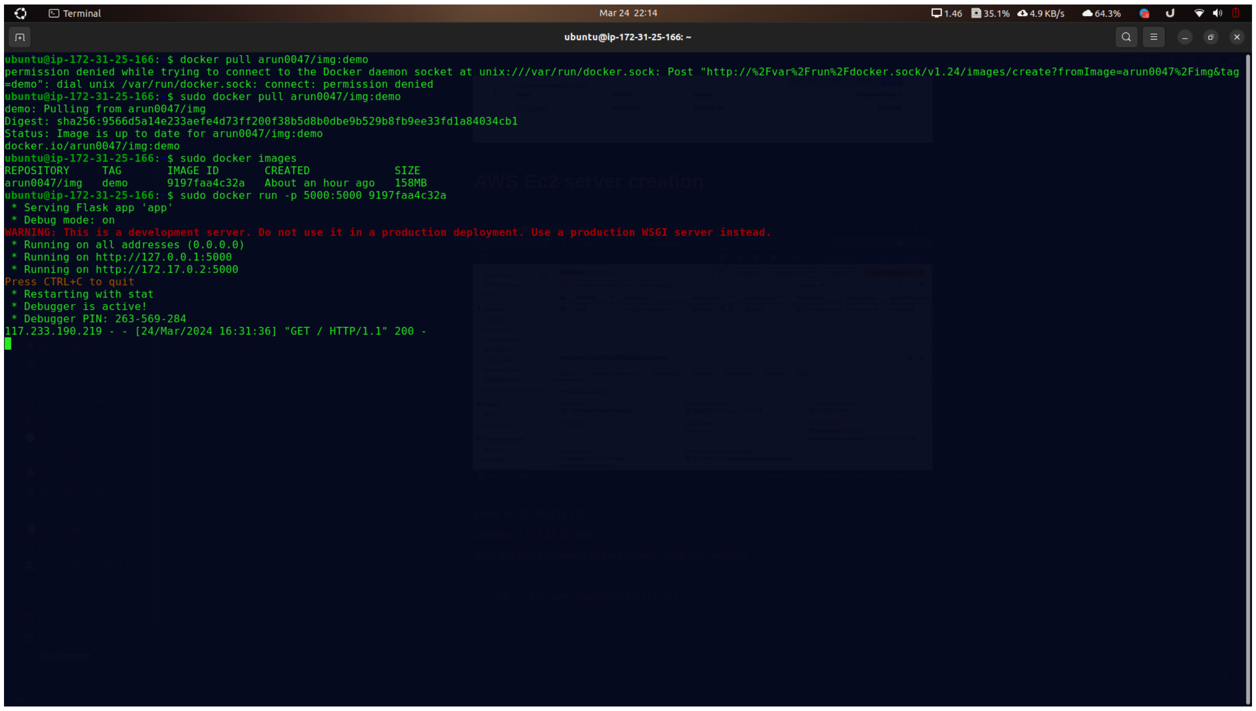
   This command installs Docker Engine, the Docker command-line tool, and the Docker container runtime.

3. **Start Docker Service**: Start the Docker service:

   ```
   sudo systemctl start docker
   ```

   You can also enable Docker to start on boot:

   ```
   sudo systemctl enable docker
   ```



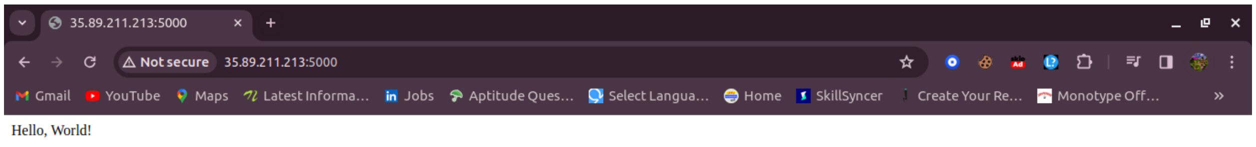
```
ubuntu@ip-172-31-25-166:~$ docker pull arun0047/img:demo
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/images/create?fromImage=arun0047%2Fimg&tag
=demo": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-25-166:~$ sudo docker pull arun0047/img:demo
demo: Pulling from arun0047/img
Digest: sha256:9566d5a14e233aefe4d73ff200f38b5d8b0dbe9b529b8fb9ee33fd1a84034cb1
Status: Image is up to date for arun0047/img:demo
docker.io/arun0047/img:demo
ubuntu@ip-172-31-25-166:~$ sudo docker images
REPOSITORY     TAG       IMAGE ID       CREATED              SIZE
arun0047/img   demo      9197faa4c32a   About an hour ago    158MB
ubuntu@ip-172-31-25-166:~$ sudo docker run -p 5000:5000 9197faa4c32a
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 263-569-284
117.233.190.219 - - [24/Mar/2024 16:31:36] "GET / HTTP/1.1" 200 -
```

pulled the docker images from docker hub and the run docker image

Then run your Docker container:

```
docker run -p 5000:5000 image
```

This will ensure that all the dependencies required by our Flask application are installed inside the Docker container when it is built and run.



Hello, World!

video： POC

```
https://drive.google.com/file/d/1f9xNbKDkyqnywoXiXQPxY69DuThNfrvP/view?usp=sharing
```