

Deployment of ML on AWS





STEPS



1) Create a ML Model



ML
Model



2) Create an API – FastAPI

Request



FastAPI

Response



3) Containerize – Docker

4) Push the Docker image to Docker hub



5) Create an AWS EC2 instance



6) Pull the Docker image from Docker hub to AWS EC2

7) Start the Docker and access the API endpoint

Create a ML Model

1) pip install scikit-learn pandas joblib

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
import joblib
import numpy as np

# Load data
iris = load_iris()
X, y = iris.data, iris.target

# Train model
model = LogisticRegression(max_iter=200)
model.fit(X, y)

# Test to see if prediction works
X_test = [5.1, 3.5, 1.4, 0.2]
X_test = np.array([X_test]) # reshape for sklearn

prediction = model.predict(X_test)
predicted_class_name = load_iris().target_names[prediction[0]]
print("Predicted class:", predicted_class_name)

# Save model
joblib.dump(model, "iris_model.pkl")
print("Model saved as iris_model.pkl")
```

Create a FastAPI and dockerize it

Create docker file:

```
FROM python:3.12 (last pushed 1 day ago)

# specify Current Working Directory
WORKDIR /code

# copy requirements.txt from local to container
COPY ./requirements.txt /code/requirements.txt

# install all requirements in the container
RUN pip install --no-cache-dir --default-timeout=100 -r /code/requirements.txt

# copy dir app from local to container
COPY ./app /code/app

# Expose port 8000 so other app can communicate with container
EXPOSE 8000

# Run following command to start uvicorn server
CMD ["uvicorn", "app.server_app:app", "--host", "0.0.0.0", "--port", "8000"]
```

Create a FastAPI and dockerize it

1) Create fastAPI server

See the code `server_app.py`

```
from fastapi import FastAPI
import joblib
import numpy as np
from sklearn.datasets import load_iris

model = joblib.load('app/iris_model.pkl')

app = FastAPI()

@app.get('/')
def read_root():
    return {'message': 'Welcome to Iris model API'}

@app.post('/predict')
def predict(data: dict):
    features = np.array(data['features']).reshape(1, -1)
    prediction = model.predict(features)
    class_name = load_iris().target_names[prediction[0]]
    return {'predicted_class': class_name}
```

Create a FastAPI and dockerize it

2) Create requirements.txt file

See the file **requirements.txt**

```
fastapi==0.116.1  
uvicorn==0.35.0  
numpy==2.0.2  
scikit-learn==1.6.1
```





3) Create Docker file

See the **Docker** file

step a: Start docker engine

Start the docker engine by simply running the **Docker Desktop**: Start -> Docker Desktop

step b: Build docker image

Run following command in **same directory** which contains Dockerfile:

> **docker build -t image_ml_model .**

This would pull the image from docker hub. This could take **4 - 10 minutes**.

step c: Run container

> **docker run --name container_ml_model -p 8000:8000 image_ml_model**

NOTE: If you make a mistake, then delete the container_ml_model

> **docker rm -f <container_id_name>**

and redo above steps



4) Goto URL <http://127.0.0.1:8000/docs>

Or

<http://127.0.0.1:8000/>

You should see a simple Welcome message

5) Access the API endpoint via python program

Run the file `client_to_access_API_endpoint.py` in a dedicated terminal

Or

Run the jupyter notebook `client_to_access_API_endpoint.ipynb`

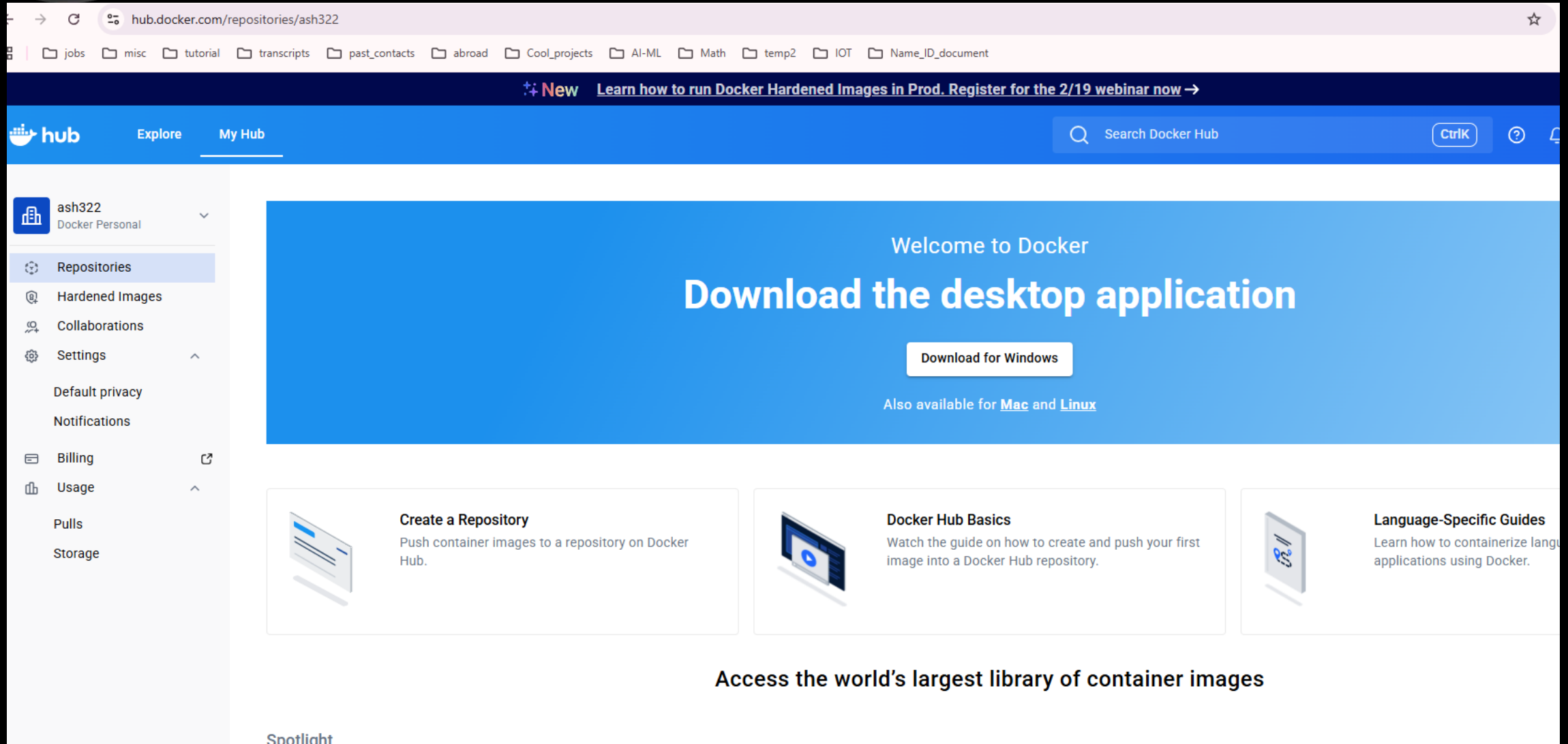
This docker is running locally



Push the Docker Image to Docker Hub

Push the Image to Docker Hub

Goto docker hub and create an account. Also note down user name



The screenshot shows the Docker Hub website for user 'ash322'. The browser address bar displays 'hub.docker.com/repositories/ash322'. The page features a blue header with the Docker Hub logo, navigation links for 'Explore' and 'My Hub', and a search bar. A sidebar on the left lists the user's profile 'ash322', navigation options like 'Repositories', 'Hardened Images', 'Collaborations', 'Settings', and account settings like 'Billing' and 'Usage'. The main content area has a large blue banner with the text 'Welcome to Docker' and 'Download the desktop application', with a 'Download for Windows' button and links for 'Mac' and 'Linux'. Below this, three cards promote 'Create a Repository', 'Docker Hub Basics', and 'Language-Specific Guides'. At the bottom, a text link says 'Access the world's largest library of container images'.

hub.docker.com/repositories/ash322

jobs misc tutorial transcripts past_contacts abroad Cool_projects AI-ML Math temp2 IOT Name_ID_document

New Learn how to run Docker Hardened Images in Prod. Register for the 2/19 webinar now →

hub Explore My Hub

Search Docker Hub CtrlK

ash322 Docker Personal

Repositories

Hardened Images

Collaborations

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Welcome to Docker

Download the desktop application

Download for Windows

Also available for [Mac](#) and [Linux](#)

Create a Repository

Push container images to a repository on Docker Hub.

Docker Hub Basics

Watch the guide on how to create and push your first image into a Docker Hub repository.

Language-Specific Guides



Learn how to containerize language applications using Docker.

Access the world's largest library of container images

Spotlight

Push the Image to Docker Hub

Create a repository: **image_ml_model**



hub.docker.com/repository/create?namespace=ash322

jobs misc tutorial transcripts past_contacts abroad Cool_projects AI-ML Math temp2 IOT Name_ID_document

New Learn how to run Docker Hardened Images in Prod. Register for the 2/19 webinar now →

hub Explore My Hub

Search Docker Hub CtrlK

ash322 Docker Personal

Repositories

Hardened Images

Collaborations

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Repositories / Create

Create repository


Repository Name *
iris-model


Short description
This is an image that contains iis model

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

Using 0 of 1 private repositories. [Get more](#)

☒ Public 
Appears in Docker Hub search results

☐ Private 
Only visible to you

Cancel Create

Pushing images

You can push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to replace `tagname` with your desired image repository tag.

Using 0 of 1 private repositories



Push the Image to Docker Hub



1) Now let's look at the docker images on local machine:

> **docker image ls**

2) Now we tag the image:

> **docker tag image_ml_model ash322/image_ml_model**

Here **ash322** is my username

3) Now login:

> **docker login**

(Follow instructions)

3) Now push the image to docker hub (This could take few minutes):

> **docker push ash322/image_ml_model**



NOW CREATE AN AWS EC2 INSTANCE
AND PULL THE DOCKER IMAGE










Here we launch the EC2 instance.

- Name = iris-model-instance
- OS image=Ubuntu
- Do not need to create key-pair. Proceed without key-pair

The screenshot displays the AWS Management Console for the Asia Pacific (Mumbai) region. The top navigation bar includes a search bar and a user profile 'ash322ash42'. The left-hand menu shows various AWS services. The main content area is divided into several sections:

- Resources:** A table showing the usage of various Amazon EC2 resources in the Asia Pacific (Mumbai) Region. The resources listed are Instances (running), Auto Scaling Groups, Capacity Reservations, Dedicated Hosts, Elastic IPs, Instances, Key pairs, Load balancers, Placement groups, Security groups, Snapshots, and Volumes.
- Launch instance:** A section with a 'Launch instance' button and a 'Migrate a server' button. It includes a note: 'Note: Your instances will launch in the Asia Pacific (Mumbai) Region'.
- Service health:** A section showing the status of the AWS service in the Asia Pacific (Mumbai) region. The status is 'This service is operating normally'.
- Instance alarms:** A section showing the status of instance alarms. It indicates '0 in alarm' and '0 OK'.
- Zones:** A table listing the available zones in the region. The zones listed are ap-south-1a, ap-south-1b, and ap-south-1c, each with a corresponding Zone ID.
- EC2 cost:** A section showing the total cost of EC2 resources. The total cost is \$0.00.




Now we connect to our EC2 instance: Connect using a Public IP

[Alt+S]Asia Pacific (Mumbai) ▼ash322ash422

[EC2](#) > [Instances](#) > [i-02c06c5d64146ea03](#) > [Connect to instance](#)

Connect Info

Connect to an instance using the browser-based client.


Instance ID  i-02c06c5d64146ea03 (image-temp)	VPC ID  vpc-05d5014b44e8bf33	Security groups  sg-0daa70c0ec529a816 (launch-wizard-2)	IAM role -
--	---	--	----------------------

EC2 Instance Connect

SSM Session Manager

SSH client


EC2 serial console

Instance ID
 i-02c06c5d64146ea03 (image-temp)

Connection type

☒ **Connect using a Public IP**
Connect using a public IPv4 or IPv6 address


☐ **Connect using a Private IP**
Connect using a private IP address and a VPC endpoint


☒ **Public IPv4 address**
 65.2.172.161

☐ **IPv6 address**
-

Username

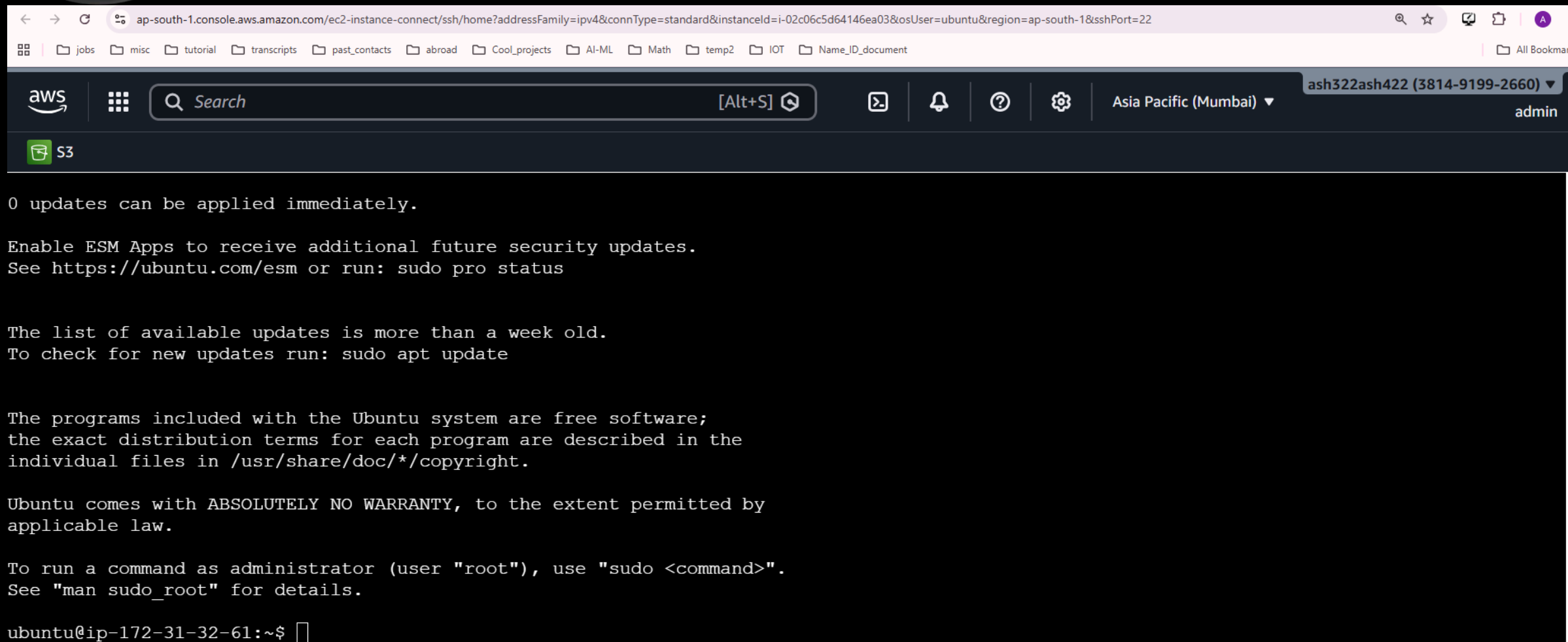
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.



 **Note:** In most cases, the default username, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

[Cancel](#) [Connect](#)

You should see following EC2 console



The screenshot shows the AWS Management Console interface for an EC2 instance. The browser address bar displays the URL: `ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-02c06c5d64146ea03&osUser=ubuntu®ion=ap-south-1&sshPort=22`. The console header includes the AWS logo, a search bar, and navigation links. The main content area shows a terminal window with the following text:

```
0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-32-61:~$
```

i-02c06c5d64146ea03 (image-temp)

PublicIPs: 65.2.172.161 PrivateIPs: 172.31.32.61



Run following commands on **EC2 console**:

Install Docker on EC2

```
> sudo apt update  
> sudo apt install docker.io -y  
> sudo systemctl start docker
```

Check if docker is running:

```
> ps -aux | grep dock
```

Login to docker:

```
> docker login
```

Go to url specified and paste the code

(Now we give permission)

```
> sudo usermod -aG docker $USER  
> newgrp docker
```

Pull Image on Ubuntu OS:

```
> docker pull ash322/image_ml_model
```

Run Container

```
> docker run -d -p 8000:8000 ash322/image_ml_model
```



Open Security Group:
Make sure you have following
Custom TCP → Port 8000 → Anywhere

EC2 > Security Groups > sg-0465e4d71bbe6c894 - launch-wizard-1

✓ Inbound security group rules successfully modified on security group (sg-0465e4d71bbe6c894 | launch-wizard-1)
▶ Details

sg-0465e4d71bbe6c894 - launch-wizard-1

Actions

Details

Security group name launch-wizard-1	Security group ID sg-0465e4d71bbe6c894	Description launch-wizard-1 created 2024-04-05T15:13:10.348Z	VPC ID vpc-05d5014b44e8bfb33
Owner 381491992660	Inbound rules count 4 Permission entries	Outbound rules count 1 Permission entry	

Inbound rules | Outbound rules | Sharing | VPC associations | Related resources - new | Tags

Inbound rules (4)

Manage tags Edit inbound rules


< 1 > ⚙

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-0a52ad55fdeef2ad6	IPv4	SSH	TCP	22	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-0bd1fb168f234f8fa	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	-
<input type="checkbox"/>	-	sgr-057eccf271c0357b8	IPv4	HTTP	TCP	80	0.0.0.0/0	allow http request
<input type="checkbox"/>	-	sgr-0e1cc319080eb849a	IPv4	HTTPS	TCP	443	0.0.0.0/0	allow https request



Access Model:

<http://EC2-PUBLIC-IP:8000/docs>



→ ↻ ⚠ Not secure 13.201.86.113:8000/docs ☆

📁 jobs 📁 misc 📁 tutorial 📁 transcripts 📁 past_contacts 📁 abroad 📁 Cool_projects 📁 AI-ML 📁 Math 📁 temp2 📁 IOT 📁 Name_ID_document

FastAPI 0.1.0 OAS 3.1

[/openapi.json](#)

default ^

GET / Read Root ▾

POST /predict Predict ▾

Schemas ^

HTTPValidationError > Expand all object

ValidationError > Expand all object



Now run following:

```
import json
import requests
```

```
data = [
    [5.4, 3.4, 1.7, 0.2],
    [5.1, 3.7, 1.5, 0.4],
    [5, 2.3, 3.3, 1],
    [6.7, 3.3, 5.7, 2.1],
]

# url = 'http://127.0.0.1:8000/predict/'
url = 'http://13.201.86.113:8000/predict/' # from AWS EC2

predictions = []
for record in data:
    payload = {'features': record}
    payload = json.dumps(payload)
    response = requests.post(url, data=payload)
    predictions.append(response.json()['predicted_class'])

print("The prediction received from server:\n", predictions)
```

```
The prediction received from server:
['setosa', 'setosa', 'versicolor', 'virginica']
```





