Week 10 Homework 1: Project: Machine Learning on Kubernetes

Naznin Nahar

ID: 20118

CS571

Machine-Learning-on-Kubernetes

Step 1: Set up a Functional Kubernetes Cluster

1. Open GKE terminal.
2. Start minikube in the Google Cloud Platform

```
minikube start
```

```
ed routes based clusters, please pass the `--no-enable-ip-alias` flag
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended alternatives. See h
ttps://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways to check usage and for migration inst
ructions.
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster kubia in us-west1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/cloudcomputinginfrastructure/zones/us-west1-b/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1-b/kubia?proj
ect=cloudcomputinginfrastructure
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1-b
MASTER_VERSION: 1.29.6-gke.1038001
MASTER_IP: 34.145.20.215
MACHINE_TYPE: n1-standard-1
NODE_VERSION: 1.29.6-gke.1038001
NUM_NODES: 3
STATUS: RUNNING
```

3. Create requirements.txt file using the following command:

```
nano requirements.txt
```

Enter the following contents in requirements.txt:

```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy==1.19.5
scipy>=0.15.1
scikit-learn==0.24.2
```

```
matplotlib>=1.4.3
pandas>=0.19
flasgger==0.9.4
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ cat requirements.txt
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy==1.19.5 # Adjusted to a version before np.float deprecation
scipy>=0.15.1
scikit-learn==0.24.2 # Ensure compatibility with numpy version
matplotlib>=1.4.3
pandas>=0.19
flasgger==0.9.4
```

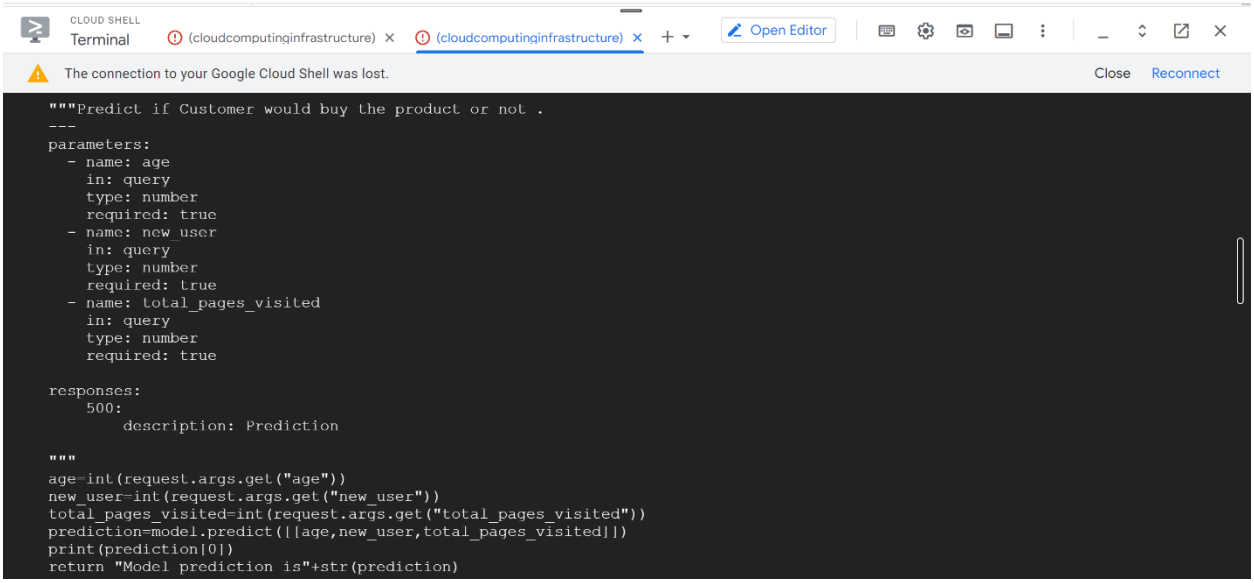
Upload logreg.pkl file which we got from ML.ipynb and using data set

- Click the three dots in the top-right part of the Cloud Shell Terminal
- Choose upload and upload the logreg.pkl file

4. Create flask_api.py file using the command:

```
nano flask_api.py
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ cat flask_api.py
# -*- coding: utf-8 -*-
"""
Created on Mon May 25 12:50:04 2020

@author: pramod.singh
"""

from flask import Flask, request
import numpy as np
import pickle
import pandas as pd
import flasgger
from flasgger import Swagger

app=Flask(__name__)
Swagger(app)

pickle_in = open("logreg.pkl","rb")
model=pickle.load(pickle_in)


@app.route('/predict',methods=["Get"])
def predict_class():
```




```
@app.route('/predict_file',methods=["POST"])
def prediction_test_file():
    """Prediction on multiple input test file .
    ---
    parameters:
      - name: file
        in: formData
        type: file
        required: true

    responses:
        500:
            description: Test file Prediction

    """
    df_test=pd.read_csv(request.files.get("file"))
    prediction=model.predict(df_test)

    return str(list(prediction))

if __name__=='__main__':
    app.run(debug=True,host='0.0.0.0',port=5000)
```

or, you can also upload it from flask_api.py file

5. Create Dockerfile using the command:

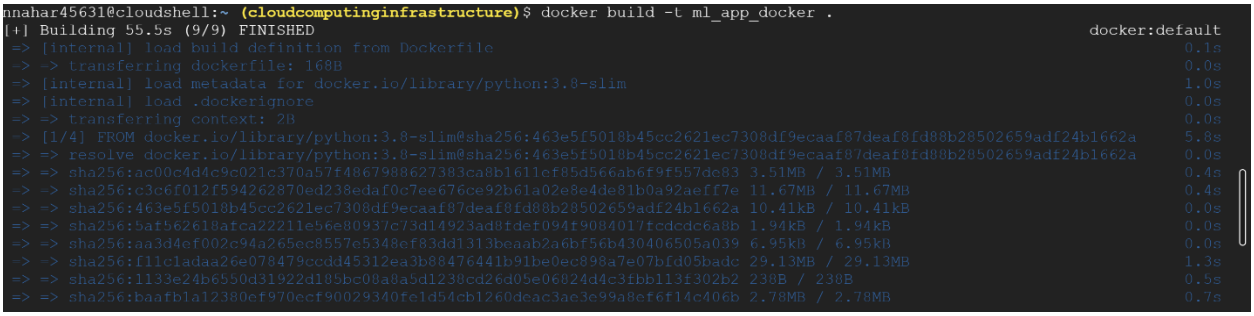
```
nano Dockerfile
```

Enter the following contents in Dockerfile:

```
FROM python:3.8-slim
WORKDIR /app
COPY . /app
EXPOSE 5000
RUN pip install -r requirements.txt
CMD ["python", "flask_api.py"]
```

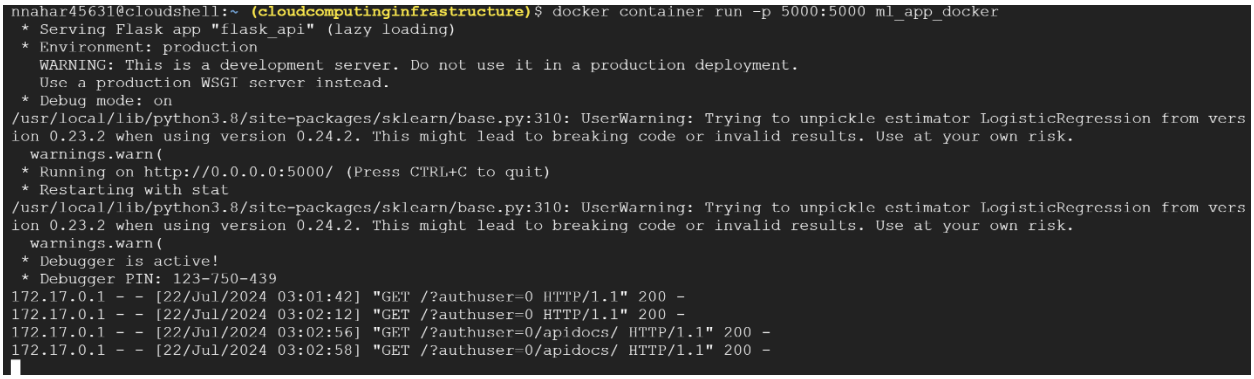
6. Build the Docker image using the command:

```
sudo docker build -t ml_app_docker .
```



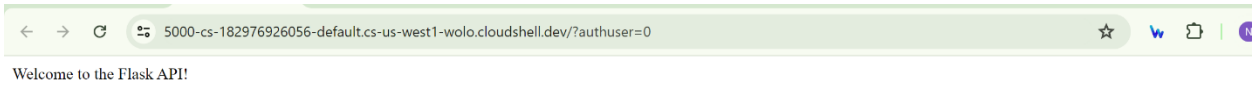
7. Run the Docker container using the command:

```
docker container run -p 5000:5000 ml_app_docker
```

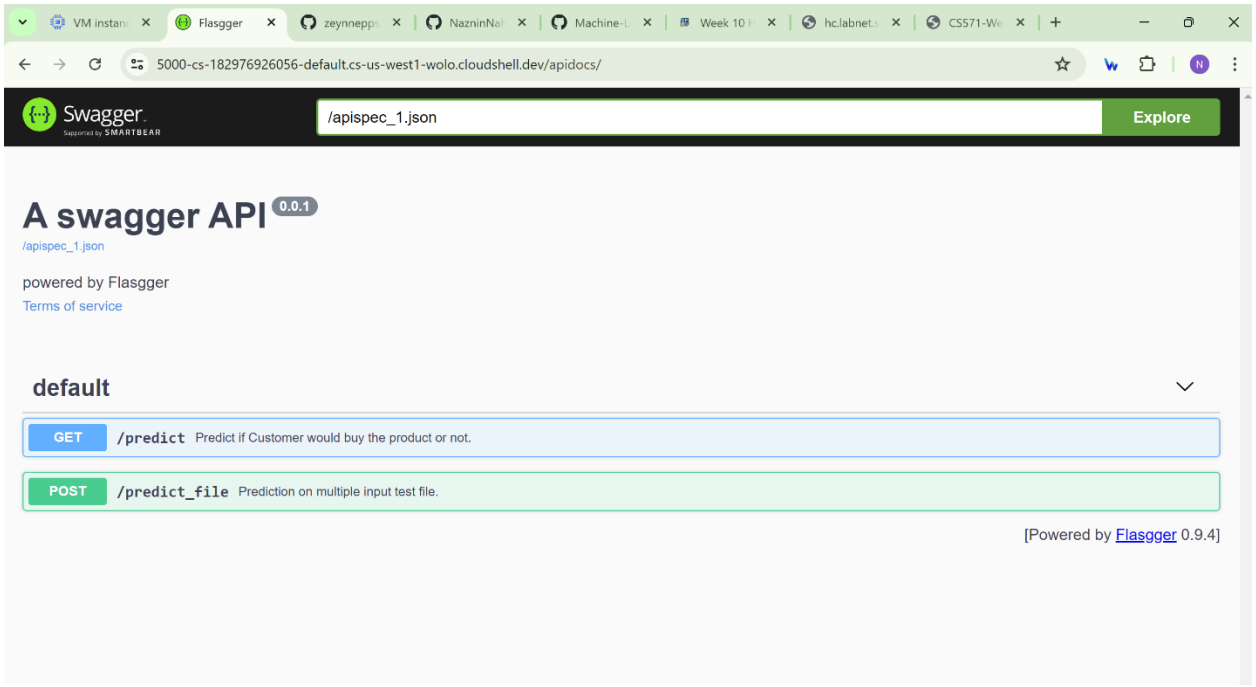


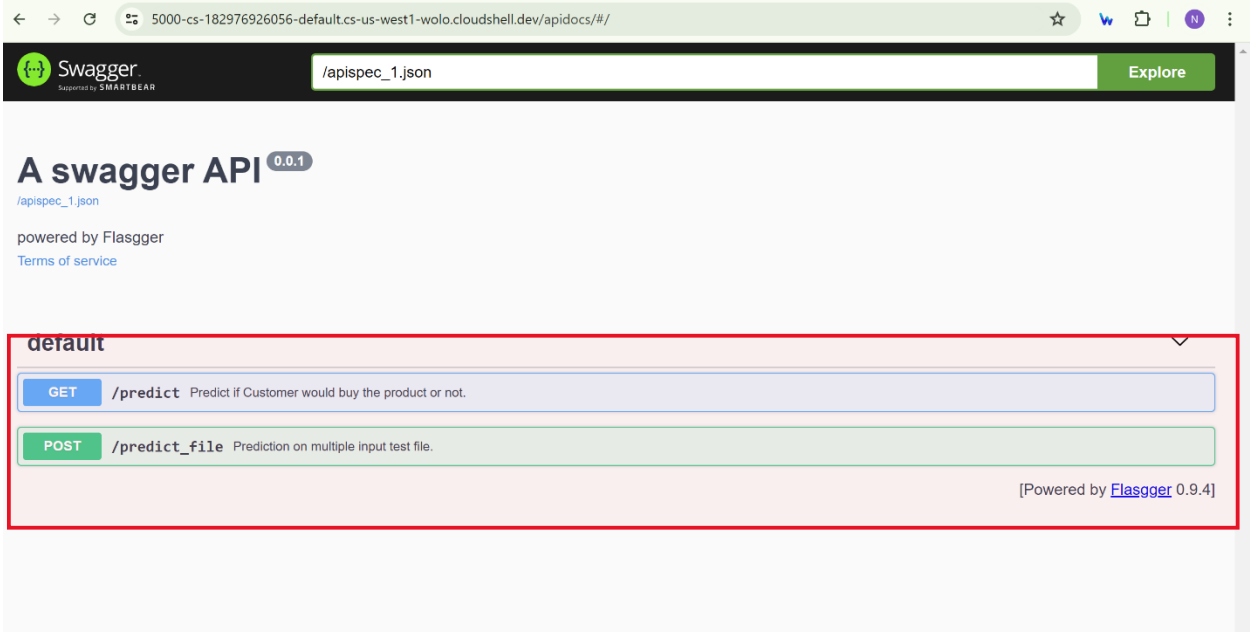
8. Preview the application:

- In the upper-right side of the terminal, click the eye-shaped button and then click "Preview on port 5000".



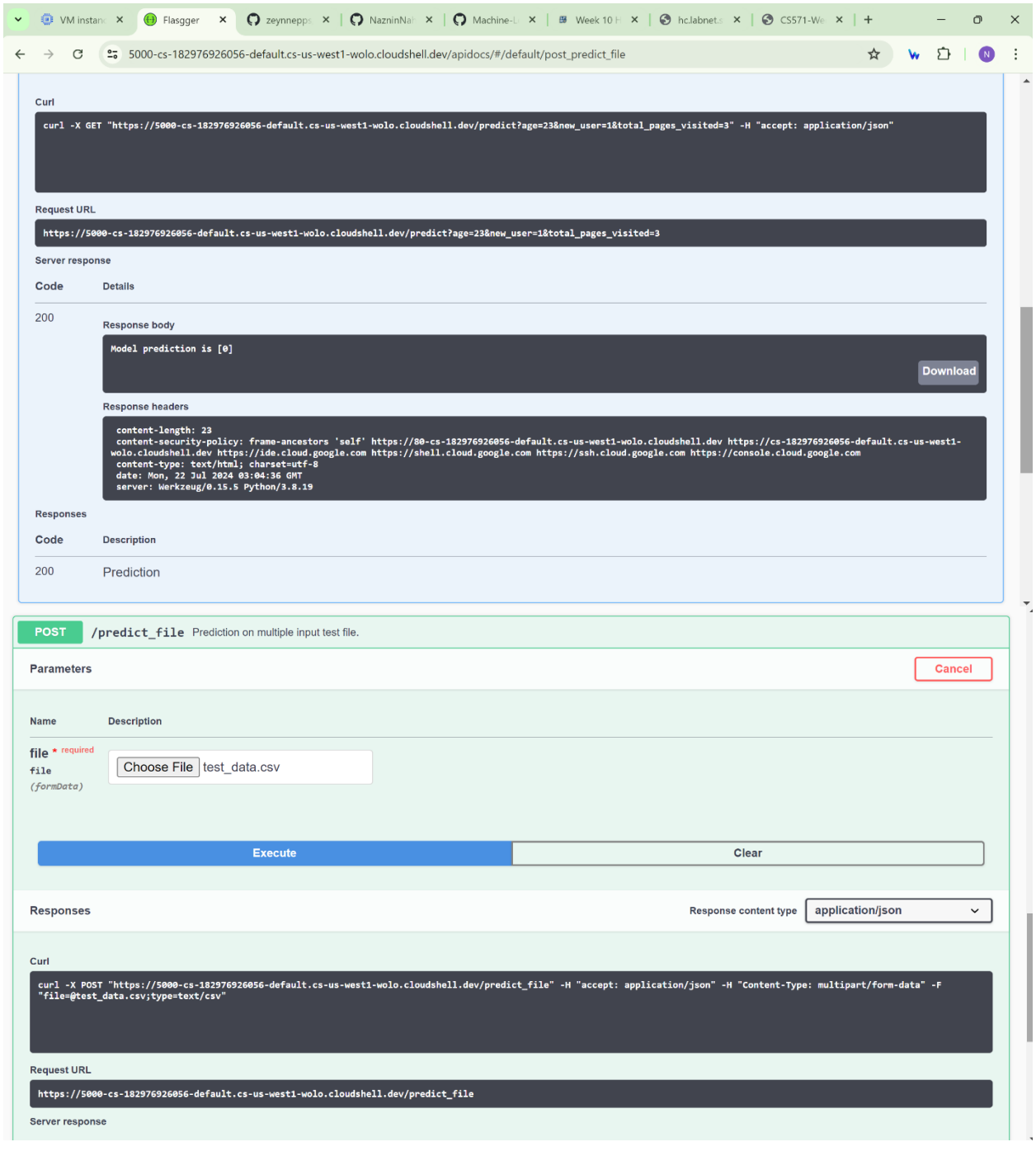
- Add **/apidocs/** at the end of the link to access the running ml-app.





10. Make predictions for a group of customers (test data) via a POST request:

- Upload the test data file containing the same parameters in a similar order.
- Execute to display the results.


Flasgger
5000-cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev/apidocs/#/default/post_predict_file
Curl
curl -X GET "https://5000-cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev/predict?age=23&new_user=1&total_pages_visited=3" -H "accept: application/json"
Request URL
https://5000-cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev/predict?age=23&new_user=1&total_pages_visited=3
Server response
Code
Details
200
Response body
Model prediction is [0]
Download
Response headers
content-length: 23
content-security-policy: frame-ancestors 'self' https://80-cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev https://cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev https://ide.cloud.google.com https://shell.cloud.google.com https://ssh.cloud.google.com https://console.cloud.google.com
content-type: text/html; charset=utf-8
date: Mon, 22 Jul 2024 03:04:36 GMT
server: Werkzeug/0.15.5 Python/3.8.19
Responses
Code
Description
200
Prediction
POST
/predict_file
Prediction on multiple input test file.
Parameters
Cancel
Name
Description
file * required
file
(formData)
Choose File
test_data.csv
Execute
Clear
Responses
Response content type
application/json
Curl
curl -X POST "https://5000-cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev/predict_file" -H "accept: application/json" -H "Content-Type: multipart/form-data" -F "file=@test_data.csv;type=text/csv"
Request URL
https://5000-cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev/predict_file
Server response

Step 1.1. Stop/kill the running container:

- List running Docker containers using the command:

```
docker ps
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS          PORTS                                                                                                                                                                                  NAMES
753831e1a9c0   gcr.io/k8s-minikube/kicbase:v0.0.44   "/usr/local/bin/entr…"   24 minutes ago   Up 24 minutes   127.0.0.1:32768->22/tcp, 127.0.0.1:32769->2376/tcp, 127.0.0.1:32770->5000/tcp, 127.0.0.1:32771->8443/tcp, 127.0.0.1:32772->32443/tcp   minikube
```

- Use the command to kill the running container

```
docker kill <CONTAINER ID>
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ docker kill 753831e1a9c0
753831e1a9c0
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$
```

Step 2: Push the image to your docker hub

1. Log in to Docker Hub:

```
docker login
```

2. Tag the Image:

```
docker tag ml_app_docker nazninnahar054/ml_docker:latest
```

3. Push the Image:

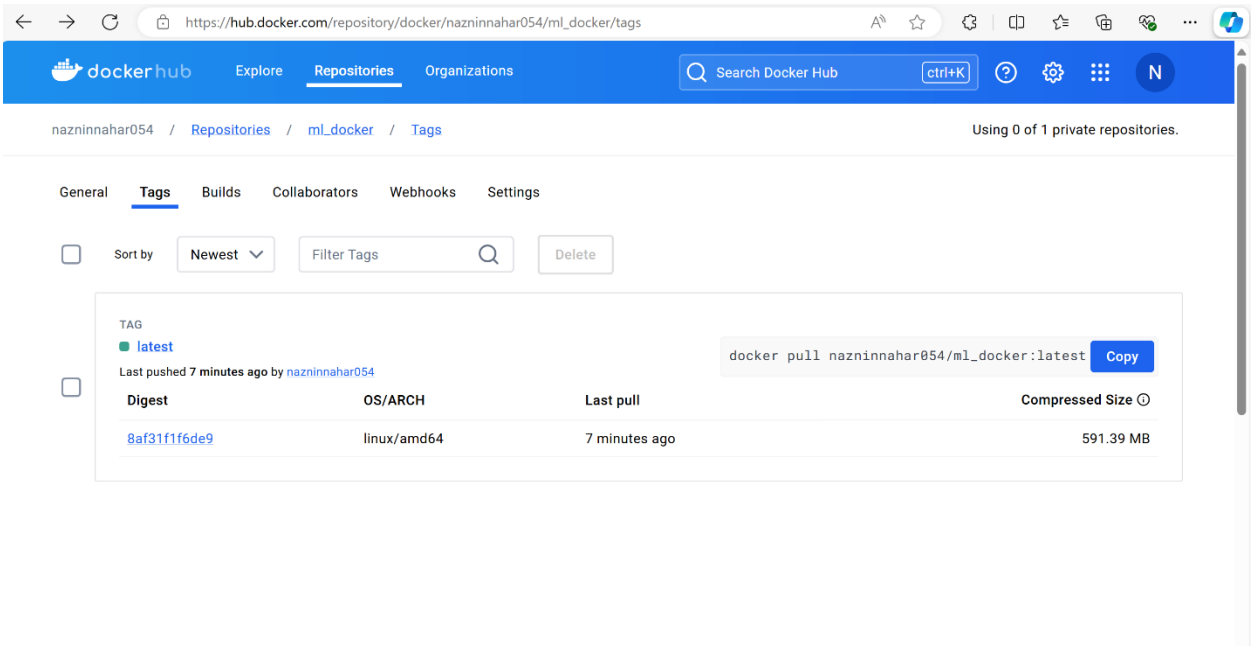
```
docker push nazninnahar054/ml_docker:latest
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ docker tag ml_app_docker nazninnahar054/ml_docker:latest
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/nnahar45631/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ docker push nazninnahar054/ml_docker:latest
The push refers to repository [docker.io/nazninnahar054/ml_docker]
f011e0d62c30: Pushed
9156ec0babee: Pushed
861e06e00097: Pushed
8b81fa4c988f: Mounted from library/python
38d395ab1f6c: Mounted from library/python
fda4aa6f33af: Mounted from library/python
a696d13c7344: Mounted from library/python
32148f9f6c5a: Mounted from library/python
latest: digest: sha256:8af31f1f6de9e069edbcc41143f46df4f61a66afcf2d312e7fa1d6ff87c09719 size: 2002
```

4. Verify the image exists locally by listing your Docker images:

```
docker images
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
nazninnahar054/ml_docker      latest    8de0993c50fd   2 minutes ago    1.28GB
ml_app_docker                 latest    8de0993c50fd   2 minutes ago    1.28GB
<none>                        <none>    70343b54dbda   18 minutes ago   1.28GB
gcr.io/k8s-minikube/kicbase   v0.0.44   5a6e59a9bdc0   2 months ago     1.26GB
```



P.S. Before login to docker build the image if pushing isn't working

Step 3: Deploy your ML app to GKE

Use the GKE we have created in Step 1

1. Create a deployment.yaml with the following contents.

   ```
   nano deployment.yaml
   ```

   then write this content

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ nano deployment.yaml
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ cat deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-app-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ml-app
  template:
    metadata:
      labels:
        app: ml-app
    spec:
      containers:
      - name: ml-app-container
        image: ruichencn/w10hw1
        ports:
        - containerPort: 5000
```

2. Create deployment with the above file.

```
kubectl apply -f ml-app-deployment.yaml
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ kubectl apply -f deployment.yaml
deployment.apps/ml-app-deployment created
```

3. Wait for couple minutes and list all the pods created

```
kubectl get deployments
kubectl get pods
```

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ kubectl get pods
NAME                                READY   STATUS              RESTARTS   AGE
ml-app-deployment-644bfb6d76-tj5r5  0/1     ContainerCreating   0          17s
```

4. Create a service.yaml

```
nano service.yaml
```

then write this content

```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ cat service.yaml
apiVersion: v1
kind: Service
metadata:
  name: ml-app-service
spec:
  selector:
    app: ml-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
      nodePort: 30001
  type: NodePort
```

6. Create service with the above file

```
kubectl apply -f ml-app-service.yaml
```

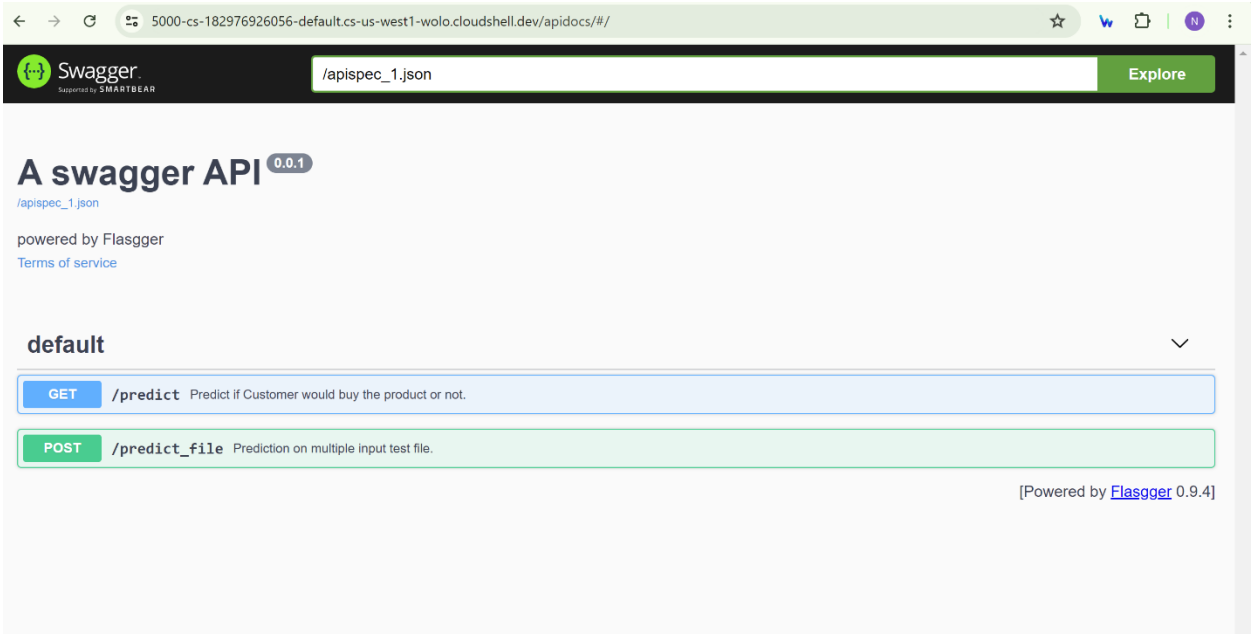
```
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ nano service.yaml
nnahar45631@cloudshell:~ (cloudcomputinginfrastructure)$ kubectl apply -f service.yaml
service/ml-app-service created
```

7. Get service external ip

```
kubectl get services
```

8. Access using browser:

```
external-ip/apidocs
```


5000-cs-182976926056-default.cs-us-west1-wolo.cloudshell.dev/apidocs/#/
Swagger
Supported by SMARTBEAR
/apispec_1.json
Explore
A swagger API
0.0.1
/apispec_1.json
powered by Flasgger
Terms of service
default
GET
/predict
Predict if Customer would buy the product or not.
POST
/predict_file
Prediction on multiple input test file.
[Powered by Flasgger 0.9.4]