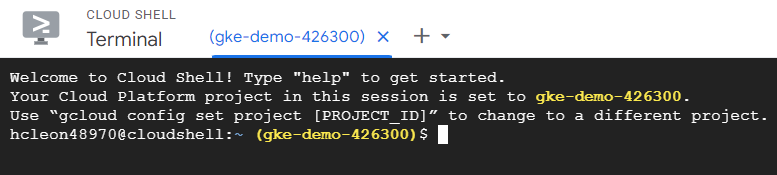
**Week 10 Homework 1: Machine Learning on Kubernetes**

Student ID: 20145

Student Name: Hartina Vonyee Cleon

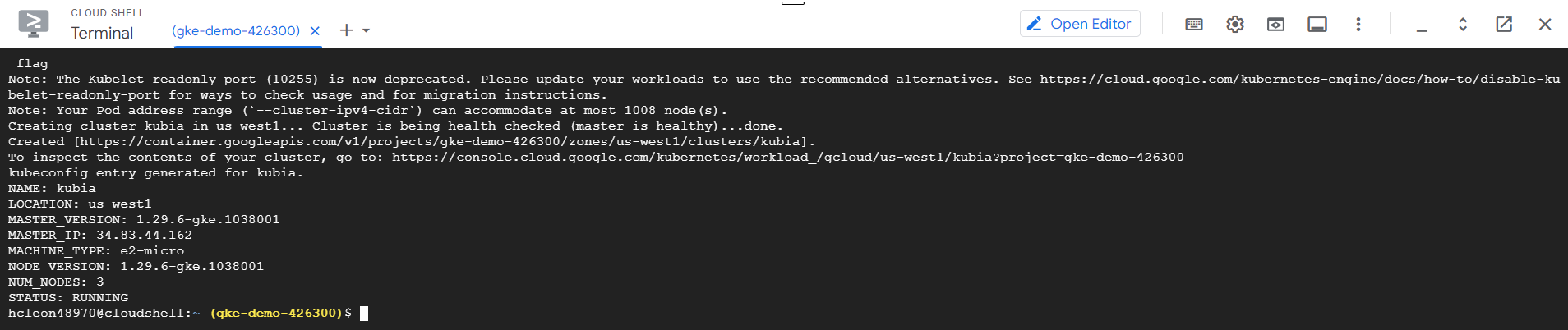
**Step 1: Set up a functional Kubernetes cluster**

1. **Open GKE terminal**



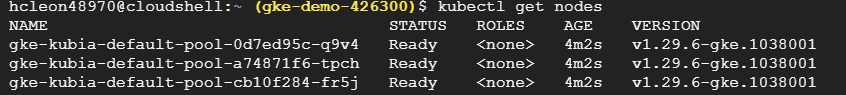
1. Now, lets create a kubenetes cluster with three nodes

gcloud container clusters create kubia --num-nodes=1 --machine-type=e2- micro --region=us-west1



1. Double check if nodes are correctly created

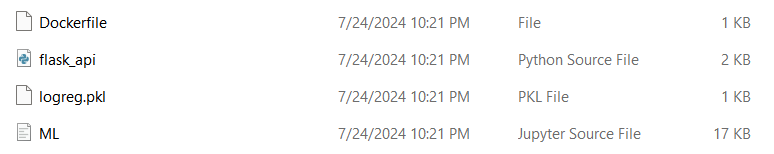
kubectl get nodes



**Step 2: Machine Learning Deployment using Docker Implement the following steps in your local machine**

**My computer: Windows hp**

1. Create a local directory
2. Put the following files into your directory

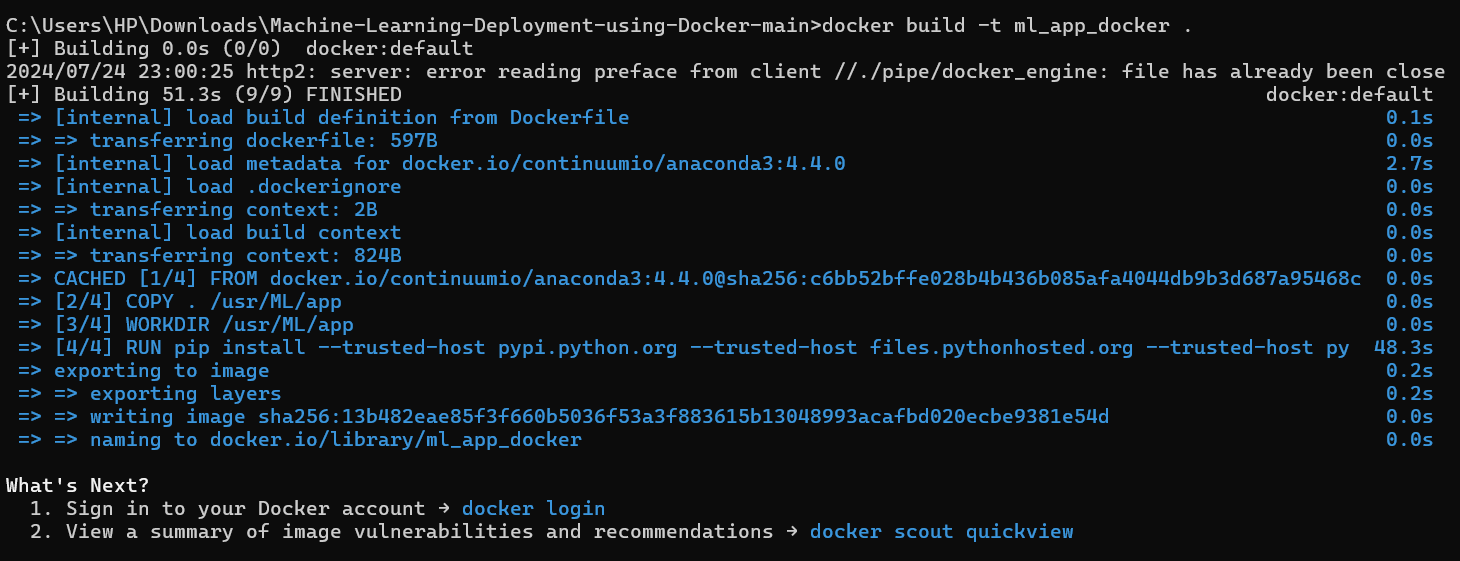




Source: <https://github.com/HasnaeTalibi/Machine-Learning-Deploymentusing-Docker/tree/main>

1. Build a docker image
2. Turn on your docker app in your local computer
3. Use the following command to build an image

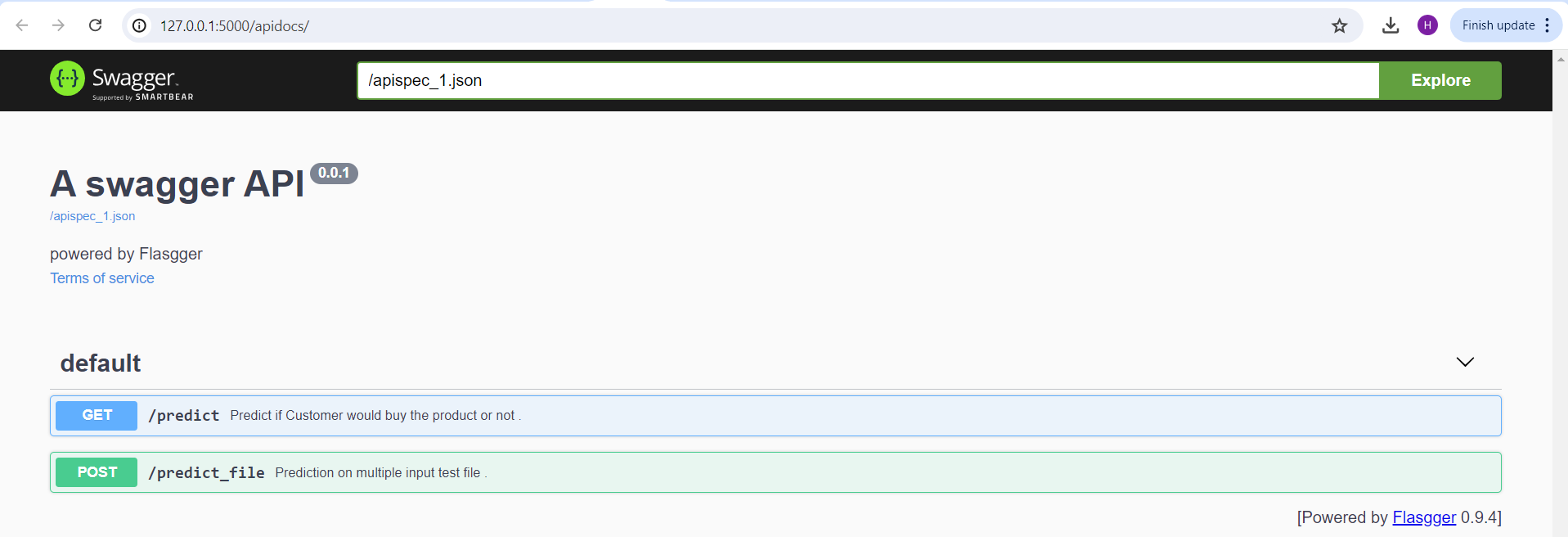
docker build -t ml\_app\_docker .



1. Initiate the container to run our ML app

docker container run -p 5000:5000 ml\_app\_docker

To access to the app, we simply have to go to http://127.0.0.1:5000/apidocs to load the Swagger UI page



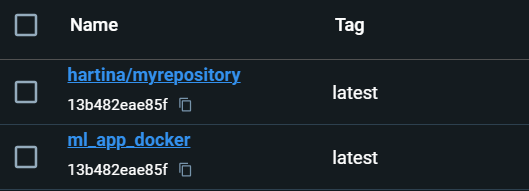
1. Push the image to your docker hub
2. Login your docker hub

docker login

1. Tag the image ”ml\_app\_docker” as latest

docker tag ml\_app\_docker yourdockerhubusername/yourrepository

1. docker push yourdockerhubusername/yourrepositoryname:latest



1. Stop the container

The last step left after running the application is to stop the running container. This can be done using the docker stop or kill command on the running container. We can see the list of running containers using the docker ps command and can select the running container ID to stop it.

docker ps

docker kill<Container\_ID>

**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.

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

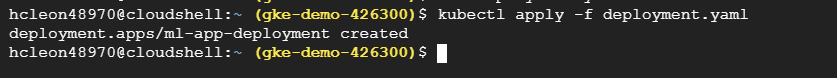
image: hartina/myrepository

ports:

- containerPort: 5000

1. Create deployment with the above file

kubectl apply -f deployment.yaml



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

kubectl get pods



1. Create a service.yaml

apiVersion: v1

kind: Service

metadata:

name: ml-app-service

spec:

selector:

app: ml-app

ports:

- protocol: TCP

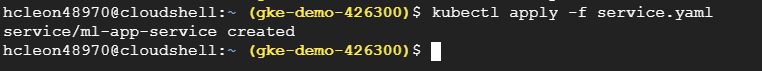
port: 80

targetPort: 5000

type: LoadBalancer

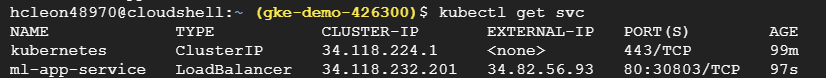
1. Create service with the above file

kubectl apply -f service.yaml



1. Get service external ip

kubectl get svc



1. Access using browser:

external-ip/apidocs

