You can access this homework on my github repository: https://github.com/yzzlqyxc/kbforML

Get access to GKE

First, we need to configure access to GKE cluster and enable Docker authentication, run the following commands:

```
gcloud container clusters get-credentials my-cluster \
  --zone us-central1-a \
  --project steam-idiom-450021-s3
gcloud auth configure-docker
```

These commands will set up your kubectl context for interacting with the cluster and configure Docker to authenticate with Google Container Registry.

prepare docker container

Second, use the following commands to build your Docker image and push it to Google Container Registry:

```
docker build -t gcr.io/steam-idiom-450021-s3/frontend:latest
docker push gcr.io/steam-idiom-450021-s3/frontend:latest
```

Once pushed, the image will be available for use in your GKE deployments.

Apply the graphic driver to the container

Third, we need to enable GPU support on GKE nodes, apply the NVIDIA driver installer DaemonSet:

```
kubectl apply -f
https://raw.githubusercontent.com/GoogleCloudPlatform/container-engine-accelerators/master/nvidia-driver-installer/ubuntu/daemonset-preloaded-R525.yaml
```

This will install the required GPU drivers on all eligible nodes in the cluster, allowing your containers to utilize GPU resources.

Create PVC to allow file access from different pods

1. Request a Cloud Filestore instance through the GCP Console. Once provisioned, obtain its IP address.
2. Create a pv.yaml and pvc.yaml to define the NFS-backed storage using the IP address obtained above. Apply them using:

```
kubectl apply -f pv.yaml
kubectl apply -f pvc.yaml
```

3. Modify train.yaml and service.yaml to mount the shared volume via the PVC in inference and training part. This enables multiple pods to read from and write to the same storage path.

apply the yaml to GKE

Apply your Kubernetes service and deployment configurations to your GKE cluster:

```
kubectl apply -f *service.yaml
kubectl apply -f *deploy.yaml
```

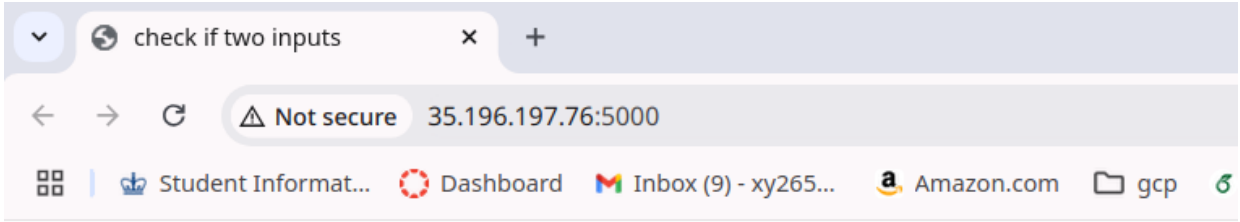
These commands will automatically create the corresponding Pods on available nodes in your cluster, based on the specifications defined in your deployment files.

command used for debugging

`kubectl exec -it PODNAME -- /bin/bash` to access the pod directly and inspect its environment
`kubectl logs PODNAME CONATAINER` to view the logs of a specific container within a pod
These commands help you troubleshoot and analyze the state of your pods and containers in real-time.

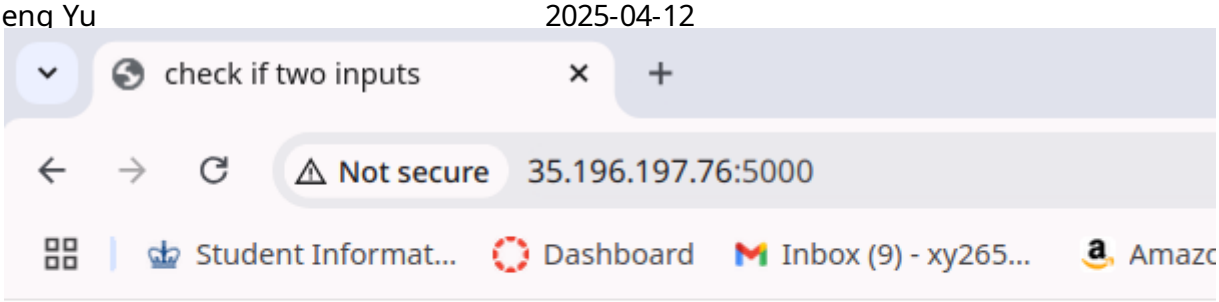
Filter | Is system object : False | Filter workloads

Name	Status	Type	Pods	Namespace	Cluster
flask-app	OK	Deployment	1/1	default	cluster-2
inference-serv	OK	Deployment	1/1	default	cluster-2
train-serv	OK	Deployment	1/1	default	cluster-2



Please input two sentences

submit



Please input two sentences

He said the food was fantastic | He hate eating anything.dddd | submit

[[0.9828916788101196, 0.01710834726691246]], 0