**Student Name:** Sim Jun Heng

**Matriculation Number:** A0218348Y

**Github Link:** <https://github.com/simjunheng/OTOT-A2-A3>

**A2 Video Link:** <https://drive.google.com/file/d/1pYifqFVKvFLchJ_txQVEzUQpS-Dbvb5s/view?usp=sharing>

**A3 Video Link:** <https://drive.google.com/file/d/1jFxgDTWCcztP8DDJ2UG9rmeJVsj2QleJ/view?usp=sharing>

**PREREQUISITES TO TASK A2 & A3**

**Pushing A1 App Image to Docker Hub**

Step 1: Push your image from A1 into docker hub by running the command **docker login** and **docker push simjunheng1/myfirstapp**

**INSTRUCTIONS FOR TASK A2.1**

**Creating a Local Cluster**

Step 1: Switch to the manifest directory using the command cd k8s/manifests

Step 2: Run the command kind create cluster --name kind-1 --config ../kind/cluster-config.yaml

Text

Description automatically generated

**Check if Local Cluster is Running**

Step 3: Check if the cluster is running using the command kubectl cluster-info

Text

Description automatically generated

**INSTRUCTIONS FOR TASK A2.2**

**Creating a Deployment Manifest**

Step 1: Create a backend-deployment.yaml file (located in the repo).

Step 2: Run the command kubectl apply -f backend-deployment.yaml



**Check if Deployment is Running & Ready**

Step 3: Run the command kubectl get deployment/backend to check if the deployments are running fine.

Text

Description automatically generated

Step 4: Run the command kubectl get pods to check if the pods are running

Text

Description automatically generated with medium confidence

**Creating an Ingress Controller**

Step 5: Run the command kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml>

A screenshot of a computer

Description automatically generated with medium confidence

**Check if Ingress Controller is Running & Ready**

Step 6:Run the command kubectl -n ingress-nginx get deployto checkif the deployment of the ingress controller is ready.

A screenshot of a computer

Description automatically generated with medium confidence

**Creating a Service Manifest**

Step 7: Create a backend-service.yaml file (located in the repo)..

Step 8: Run the command kubectl apply -f backend-service.yaml

Graphical user interface, text

Description automatically generated

**Check if Service is Created**

Step 9: Run the command kubectl get svcto check if the service is created

Text

Description automatically generated

**INSTRUCTIONS FOR TASK A2.3**

**Creating an Ingress Manifest**

Step 1: Create a backend-ingress.yaml file (located in the repo).

Step 2: Run the command kubectl apply -f backend-ingress.yaml

Text

Description automatically generated

**Check if Ingress is Created Successfully**

Step 3: Run the command kubectl get ingress check if the ingress is set up successfully.

Text

Description automatically generated with medium confidence

**Check if App Can Be Accessed**

Graphical user interface, text, application, email

Description automatically generated

**INSTRUCTIONS FOR TASK A3.1**

**Creating a Local Cluster**

Step 1: Switch to the manifest directory using the command cd k8s/manifests

Step 2: Run the command kind create cluster --name kind-1 --config ../kind/cluster-config.yaml

Text

Description automatically generated

**Check if Local Cluster is Running**

Step 3: Check if the cluster is running using the command kubectl cluster-info

Text

Description automatically generated

**Creating a Metrics Server**

Step 4: Install manifest from the provided link in A3 guide using the command kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>

Text

Description automatically generated

Step 5: Run the command kubectl -nkube-system edit deploy/metrics-server**.** Then manually edit the deployment manifest to add a flag --kubelet-insecure-tls to deployment.spec.containers[].args[].

Graphical user interface, text, application, email

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Step 6: Run the command kubectl -nkube-system rollout restart deploy/metrics-serverto restart the deployment**.**

Text

Description automatically generated

**Check if Metrics Server is Working**

Step 7:Run the command kubectl get pods --all-namespaces | findstr "metrics-server" to checkif the metric server is running on my cluster.

Text

Description automatically generated

**Creating a Horizontal Pod Auto Scaler**

Step 8: Create a backend-hpa file (located in the repo).

Step 9: Run the command kubectl apply -f backend-hpa.yaml

A screenshot of a computer

Description automatically generated with medium confidence

**Check if Horizontal Pod Auto Scaler is Working**

Step 10:Run the command kubectl get hpa to checkif the hpa working.

Text

Description automatically generated

**INSTRUCTIONS FOR TASK A3.2**

**Creating a Deployment Manifest**

Step 1: Create a backend-deployment-zone-aware.yaml file (located in the repo).

Step 2: Run the command kubectl apply -f backend-deployment-zone-aware.yaml

Text

Description automatically generated

**Check if Deployment is Running & Ready**

Step 3: Run the command kubectl get po -lapp=backend-zone-aware -owide --sort-by='.spec.nodeName' to check if the deployments are running fine.

Text

Description automatically generated

Step 4: Run the command kubectl get pods to check if the pods are running

Graphical user interface, text

Description automatically generated with medium confidence

**Creating an Ingress Controller**

Step 5: Run the command kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml>

Text

Description automatically generated

**Check if Ingress Controller is Running & Ready**

Step 6:Run the command kubectl -n ingress-nginx get deployto checkif the deployment of the ingress controller is ready.

Graphical user interface, text

Description automatically generated

**Creating a Service Manifest**

Step 7: Create a backend-service-zone-aware.yaml file (located in the repo)..

Step 8: Run the command kubectl apply -f backend-service-zone-aware.yaml

A screenshot of a computer

Description automatically generated with medium confidence

**Check if Service is Created**

Step 9: Run the command kubectl get svcto check if the service is created

Text

Description automatically generated

**Creating an Ingress Manifest**

Step 10: Create a backend-ingress-zone-aware.yaml file (located in the repo)..

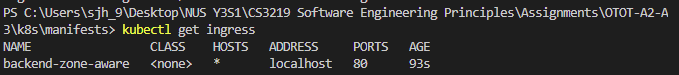
Step 11: Run the command kubectl apply -f backend-ingress-zone-aware.yaml

Text

Description automatically generated

**Check if Ingress is Created Successfully**

Step 12: Run the command kubectl get ingress check if the ingress is set up successfully.



**Check if Pods are Running On Different Zones**

Step 13: Run the command kubectl get nodes -L topology.kubernetes.io/zone

Text

Description automatically generated with medium confidence

**Check if HPA is Scaling Properly**

Step 14: Run the command kubectl get hpa -w check if the hpa is scaling properly.

**BEFORE (1 Replica) AND AFTER SCALE (4 Replica)**

**Explanation: HPA will always seek to achieve an average of 50% CPU utilization rate among the pods. If it exceeds this range, HPA will increase the number of replicas/pods as seen below.**

Text

Description automatically generated

**Check if App Can Be Accessed**

Graphical user interface, text, application, email

Description automatically generated