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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kind create cluster --name kind-1 --config ../kind/cluster-con
fig.yaml
Creating cluster "kind-1" ...
 • Ensuring node image (kindest/node:v1.25.3) 🖼 ...
 ✓ Ensuring node image (kindest/node:v1.25.3) 🖼
 • Preparing nodes 📦 📦 📦 📦   ...
 ✓ Preparing nodes 📦 📦 📦 📦
 • Writing configuration 📜  ...
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 • Installing CNI 🔌  ...
 ✓ Installing CNI 🔌
 • Installing StorageClass 💾  ...
 ✓ Installing StorageClass 💾
 • Joining worker nodes 🚜  ...
Set kubectl context to "kind-kind-1"
You can now use your cluster with:
kubectl cluster-info --context kind-kind-1
```

Check if Local Cluster is Running

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

```
TOT-A2-A3\k8s\manifests> kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:52381
CoreDNS is running at https://127.0.0.1:52381/api/v1/namespaces/kube-system/services/ku
```

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`

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl get deployment/backend
```

Check if Deployment is Running & Ready

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl get deployment/backend
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
backend   3/3     3            3           8m54s
```

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
backend-757f6c7c8-jqqws    1/1     Running   0          9m8s
backend-757f6c7c8-qvwjg    1/1     Running   0          9m8s
backend-757f6c7c8-s57ld    1/1     Running   0          9m8s
```

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl apply -f https://raw.githubusercontent.com/kubernetes/
ingress-nginx/main/deploy/static/provider/kind/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
ated
```

Check if Ingress Controller is Running & Ready

Step 6: Run the command `kubectl -n ingress-nginx get deploy` to check if the deployment of the ingress controller is ready.

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl -n ingress-nginx get deploy -w
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
```

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`

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl apply -f backend-service.yaml
service/backend created
```

Check if Service is Created

Step 9: Run the command `kubectl get svc` to check if the service is created

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl get svc
NAME         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
backend      ClusterIP   10.96.166.147   <none>        8080/TCP   9s
kubernetes   ClusterIP   10.96.0.1       <none>        443/TCP    11m
```

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`

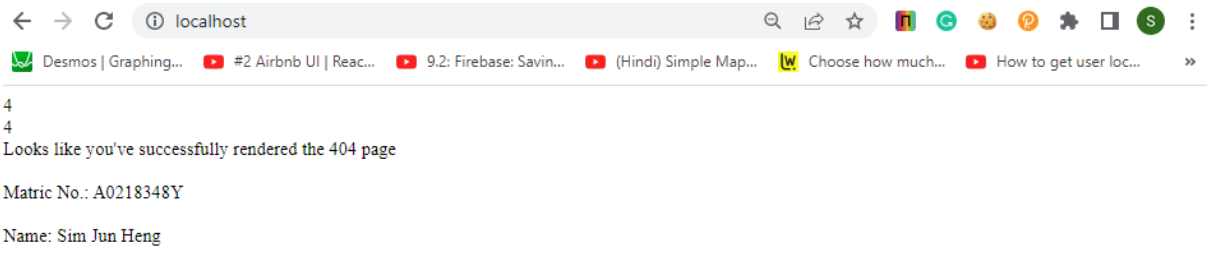
```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl apply -f backend-ingress.yaml
ingress.networking.k8s.io/backend created
```

Check if Ingress is Created Successfully

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kubectl get ingress
NAME      CLASS    HOSTS   ADDRESS   PORTS   AGE
backend   <none>   *                 80      6s
```

Check if App Can Be Accessed



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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\O
TOT-A2-A3\k8s\manifests> kind create cluster --name kind-1 --config ../kind/cluster-con
fig.yaml
Creating cluster "kind-1" ...
 • Ensuring node image (kindest/node:v1.25.3) 🖼 ...
 ✓ Ensuring node image (kindest/node:v1.25.3) 🖼
 • Preparing nodes 📦 📦 📦 📦   ...
 ✓ Preparing nodes 📦 📦 📦 📦
 • Writing configuration 📜  ...
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 • Installing CNI 🔌  ...
 ✓ Installing CNI 🔌
 • Installing StorageClass 💾  ...
 ✓ Installing StorageClass 💾
 • Joining worker nodes 🚜  ...
Set kubectl context to "kind-kind-1"
You can now use your cluster with:
kubectl cluster-info --context kind-kind-1
```

Check if Local Cluster is Running

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:60787
CoreDNS is running at https://127.0.0.1:60787/api/v1/namespaces/kube-system/services/kube-dns:d
ns/proxy
```

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/la
test/download/components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

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[].

```
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        k8s-app: metrics-server
    spec:
      containers:
      - args:
        - --cert-dir=/tmp
        - --secure-port=4443
        - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
        - --kubelet-use-node-status-port
        - --metric-resolution=15s
        - --kubelet-insecure-tls
        image: k8s.gcr.io/metrics-server/metrics-server:v0.6.1
        imagePullPolicy: IfNotPresent
        livenessProbe:
          failureThreshold: 3
          httpGet:
```

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl -nkube-system edit deploy/metrics-server
deployment.apps/metrics-server edited
```

Step 6: Run the command `kubectl -nkube-system rollout restart deploy/metrics-server` to restart the deployment.

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl -nkube-system rollout restart deploy/metrics-server
deployment.apps/metrics-server restarted
```

Check if Metrics Server is Working

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl get pods --all-namespaces | findstr "metrics-server"
kube-system          metrics-server-bf9946b6d-746b8               1/1     Running   0
```

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`

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl apply -f backend-hpa.yaml
horizontalpodautoscaler.autoscaling/backend-zone-aware created
```

Check if Horizontal Pod Auto Scaler is Working

Step 10: Run the command `kubectl get hpa` to check if the hpa working.

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl get hpa
NAME                REFERENCE                      TARGETS   MINPODS   MAXPODS   REPLICAS   A
GE
backend-zone-aware   Deployment/backend-zone-aware   0%/50%    1         10        10         6
6s
```

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`

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl apply -f backend-deployment-zone-aware.yaml
deployment.apps/backend-zone-aware created
```

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.

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl get po -lapp=backend-zone-aware -owide --sort-by='.spec.nodeName'
NAME                                  READY   STATUS    RESTARTS   AGE     IP           NODE
          NOMINATED NODE   READINESS GATES
backend-zone-aware-566cc6bc6d-5jtzv   1/1     Running   0          7m48s   10.244.1.5   kind-1-
worker    <none>           <none>
backend-zone-aware-566cc6bc6d-7gn9g   1/1     Running   0          7m48s   10.244.1.3   kind-1-
worker    <none>           <none>
backend-zone-aware-566cc6bc6d-dp8l2   1/1     Running   0          7m48s   10.244.1.4   kind-1-
worker    <none>           <none>
backend-zone-aware-566cc6bc6d-rpxkd   1/1     Running   0          7m48s   10.244.1.2   kind-1-
worker    <none>           <none>
backend-zone-aware-566cc6bc6d-84v8l   1/1     Running   0          7m48s   10.244.3.4   kind-1-
worker2   <none>           <none>
backend-zone-aware-566cc6bc6d-8b7vm   1/1     Running   0          7m48s   10.244.2.6   kind-1-
worker3   <none>           <none>
backend-zone-aware-566cc6bc6d-l55nt   1/1     Running   0          7m48s   10.244.2.3   kind-1-
worker3   <none>           <none>
backend-zone-aware-566cc6bc6d-nwkjh   1/1     Running   0          7m48s   10.244.2.4   kind-1-
worker3   <none>           <none>
backend-zone-aware-566cc6bc6d-rrsp5   1/1     Running   0          7m48s   10.244.2.7   kind-1-
worker3   <none>           <none>
backend-zone-aware-566cc6bc6d-tbfxg   1/1     Running   0          7m48s   10.244.2.5   kind-1-
worker3   <none>           <none>
```

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
backend-zone-aware-566cc6bc6d-5jtzv   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-7gn9g   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-84v8l   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-8b7vm   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-dp8l2   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-l55nt   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-nwkjh   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-rpxkd   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-rrsp5   1/1     Running   0          8m11s
backend-zone-aware-566cc6bc6d-tbfxg   1/1     Running   0          8m11s
```

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-
3\k8s\manifests> kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/m
in/deploy/static/provider/kind/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
```

Check if Ingress Controller is Running & Ready

Step 6: Run the command `kubectl -n ingress-nginx get deploy` to check if the deployment of the ingress controller is ready.

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl -n ingress-nginx get deploy
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
ingress-nginx-controller   1/1     1            1           82s
```

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`

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl apply -f backend-service-zone-aware.yaml
service/backend-zone-aware created
```

Check if Service is Created

Step 9: Run the command `kubectl get svc` to check if the service is created

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl get svc
NAME                 TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
backend              ClusterIP   10.96.40.251    <none>        8080/TCP   110s
backend-zone-aware   ClusterIP   10.96.31.200    <none>        8080/TCP   3s
kubernetes           ClusterIP   10.96.0.1       <none>        443/TCP    17m
```

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`

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl apply -f backend-ingress-zone-aware.yaml
ingress.networking.k8s.io/backend-zone-aware created
```

Check if Ingress is Created Successfully

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl get ingress
NAME                 CLASS    HOSTS   ADDRESS     PORTS   AGE
backend-zone-aware   <none>   *       localhost   80      93s
```

Check if Pods are Running On Different Zones

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

```
PS C:\Users\sjh_9\Desktop\NUS Y3S1\CS3219 Software Engineering Principles\Assignments\OTOT-A2-A
3\k8s\manifests> kubectl get nodes -L topology.kubernetes.io/zone
NAME                  STATUS   ROLES           AGE   VERSION   ZONE
kind-1-control-plane  Ready    control-plane   20m   v1.25.3
kind-1-worker         Ready    <none>          20m   v1.25.3   a
kind-1-worker2        Ready    <none>          20m   v1.25.3   a
kind-1-worker3        Ready    <none>          20m   v1.25.3   b
```

Check if HPA is Scaling Properly

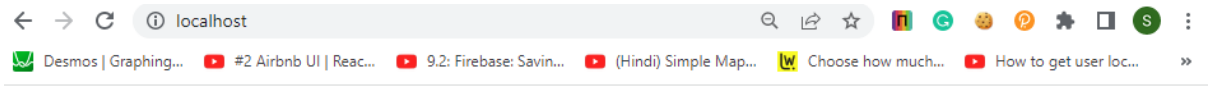
Step 14: Run the command `kubectl get hpa -w` check if the hpa is scaling properly.

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.

```
backend-zone-aware   Deployment/backend-zone-aware   0%/50%     1   10   1   22m
backend-zone-aware   Deployment/backend-zone-aware   10%/50%    1   10   1   23m
backend-zone-aware   Deployment/backend-zone-aware   100%/50%   1   10   1   23m
backend-zone-aware   Deployment/backend-zone-aware   97%/50%    1   10   2   23m
backend-zone-aware   Deployment/backend-zone-aware   100%/50%   1   10   2   24m
backend-zone-aware   Deployment/backend-zone-aware   88%/50%    1   10   2   24m
backend-zone-aware   Deployment/backend-zone-aware   57%/50%    1   10   4   24m
backend-zone-aware   Deployment/backend-zone-aware   55%/50%    1   10   4   24m
```

Check if App Can Be Accessed



4
4
Looks like you've successfully rendered the 404 page

Matric No.: A0218348Y

Name: Sim Jun Heng