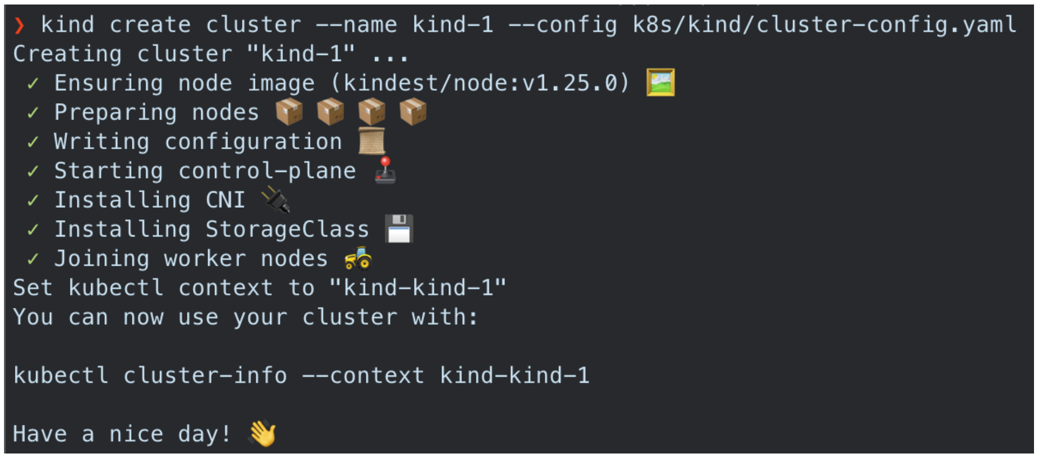
Name: Chen Hsiao Ting
Matriculation Number: A0222182R
Link to GitHub repository: https://github.com/hsiaotingluv/CS3219-OTOT-TaskA2-A3
Link to Demo video:
https://drive.google.com/file/d/1q6xOcmwVU5KD8ske_ENULFgm_kKD_omM/view?usp=sharing

Instructions on how to run commands to create the clusters and k8s objects

Task A2.1: deploy a local k8s cluster

https://github.com/CS3219-AY2223S1/OTOT-A2-A3/tree/main/demo/a2

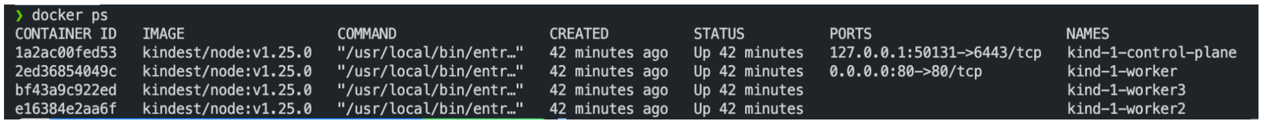
1. create cluster



```
❯ kind create cluster --name kind-1 --config k8s/kind/cluster-config.yaml
Creating cluster "kind-1" ...
 ✓ Ensuring node image (kindest/node:v1.25.0) 🖼
 ✓ Preparing nodes 📦 📦 📦 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ 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

Have a nice day! 👋
```

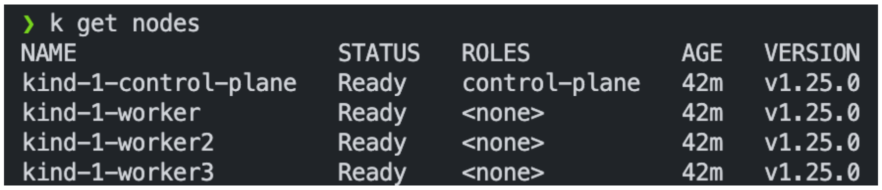
- run `kind create cluster --name kind-1 --config k8s/kind/cluster-config.yaml` in root of repo

2. verify cluster



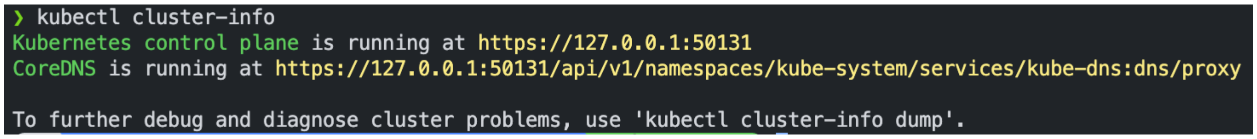
```
❯ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                       NAMES
1a2ac00fed53   kindest/node:v1.25.0   "/usr/local/bin/entr…"   42 minutes ago   Up 42 minutes   127.0.0.1:50131->6443/tcp   kind-1-control-plane
2ed36854049c   kindest/node:v1.25.0   "/usr/local/bin/entr…"   42 minutes ago   Up 42 minutes   0.0.0.0:80->80/tcp          kind-1-worker
bf43a9c922ed   kindest/node:v1.25.0   "/usr/local/bin/entr…"   42 minutes ago   Up 42 minutes                               kind-1-worker3
e16384e2aa6f   kindest/node:v1.25.0   "/usr/local/bin/entr…"   42 minutes ago   Up 42 minutes                               kind-1-worker2
```

- inspect the node containers using `docker ps` after creating the cluster



```
❯ k get nodes
NAME                   STATUS   ROLES           AGE   VERSION
kind-1-control-plane   Ready    control-plane   42m   v1.25.0
kind-1-worker          Ready    <none>          42m   v1.25.0
kind-1-worker2         Ready    <none>          42m   v1.25.0
kind-1-worker3         Ready    <none>          42m   v1.25.0
```

- inspect nodes in k8s using `kubectl get nodes`



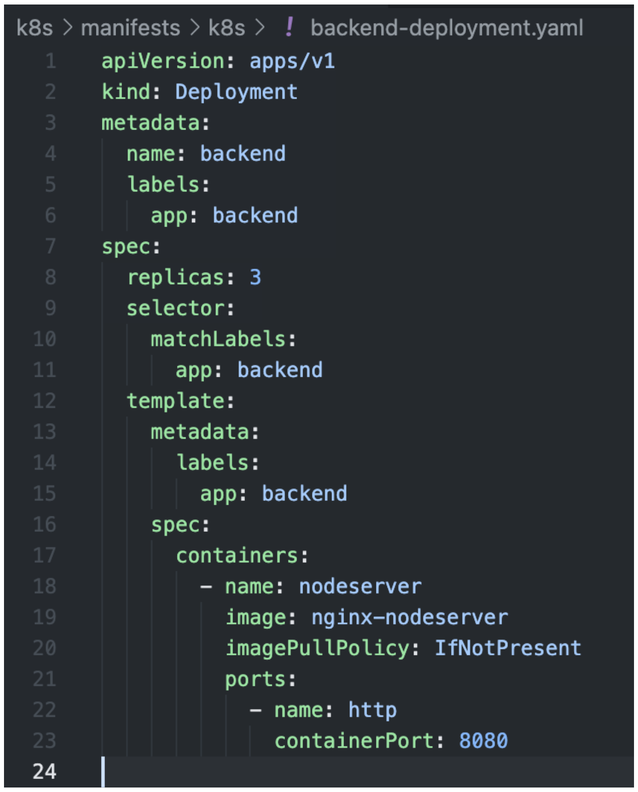
```
❯ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:50131
CoreDNS is running at https://127.0.0.1:50131/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

- run `kubectl cluster-info` to verify cluster

Task A2.2: deploy your A1 Docker image to Kubernetes as a Deployment with 3 replicas exposed by a Service object

https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/

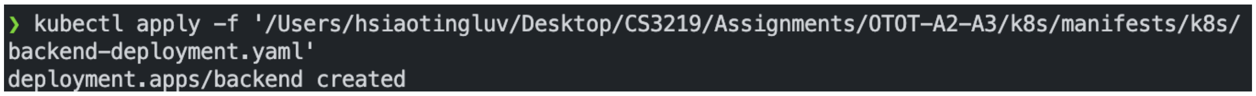
1. Add the relevant Deployment manifest



k8s > manifests > k8s > ! backend-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  labels:
    app: backend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: nodeserver
          image: nginx-nodeserver
          imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 8080

```

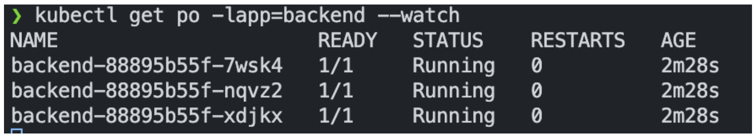
2. Load the image locally for the Node App
    - run `kind load docker-image nginx-nodeserver --name kind-1`
3. Apply the Deployment manifest



```
❯ kubectl apply -f '/Users/hsiaotingluv/Desktop/CS3219/Assignments/OTOT-A2-A3/k8s/manifests/k8s/
backend-deployment.yaml'
deployment.apps/backend created
```

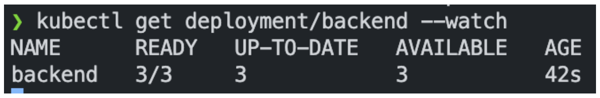
- run `kubectl apply -f backend-deployment.yaml`

4. Verify that the Pods/containers are running



```
❯ kubectl get po -lapp=backend --watch
NAME                      READY   STATUS    RESTARTS   AGE
backend-88895b55f-7wsk4   1/1     Running   0          2m28s
backend-88895b55f-nqvz2   1/1     Running   0          2m28s
backend-88895b55f-xdjkx   1/1     Running   0          2m28s
```

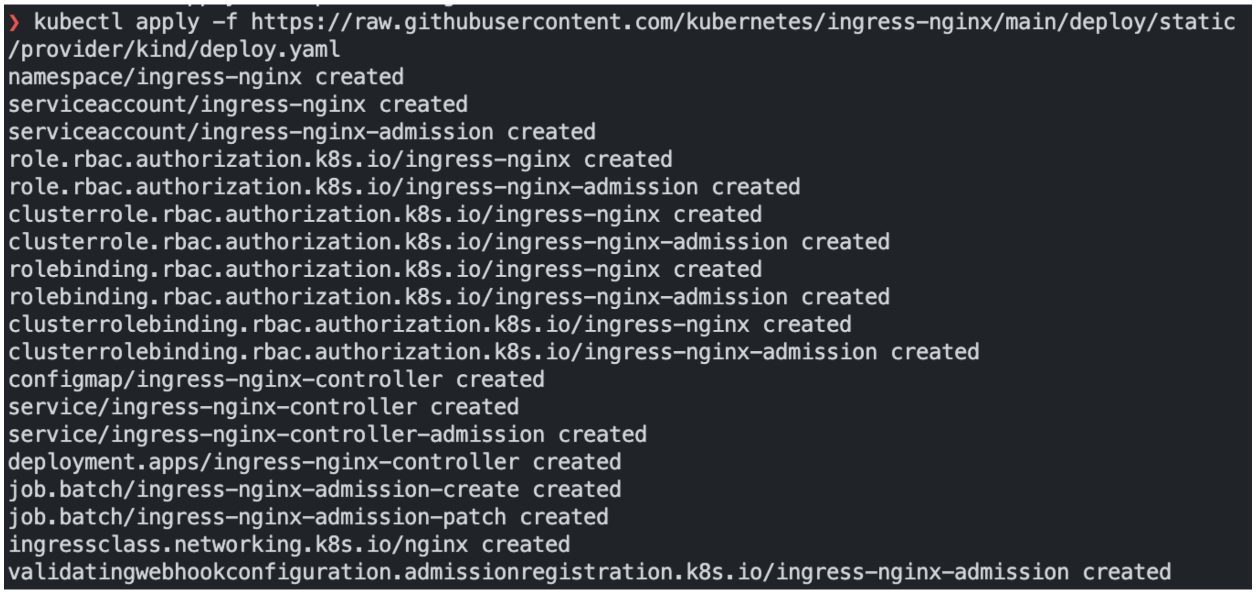
- run `kubectl get po -lapp=backend --watch` to check if each of the individual containers are ready



```
❯ kubectl get deployment/backend --watch
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
backend   3/3     3            3           42s
```

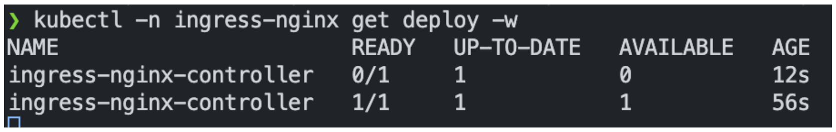
- run `kubectl get deployment/backend --watch` to check if all the containers are ready

5. Create Ingress controller (nginx-ingress-controller)



```
❯ 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
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
```

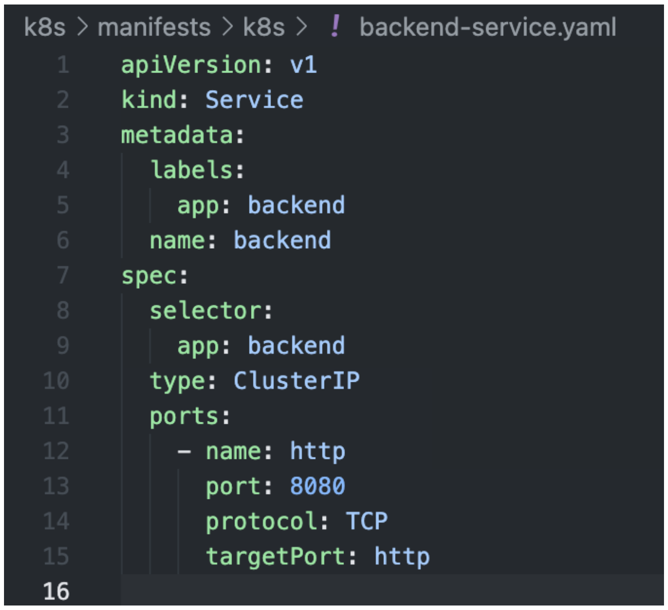
- run `kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml`



```
❯ kubectl -n ingress-nginx get deploy -w
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
ingress-nginx-controller   0/1     1            0           12s
ingress-nginx-controller   1/1     1            1           56s
```

- run `kubectl -n ingress-nginx get deploy -w`

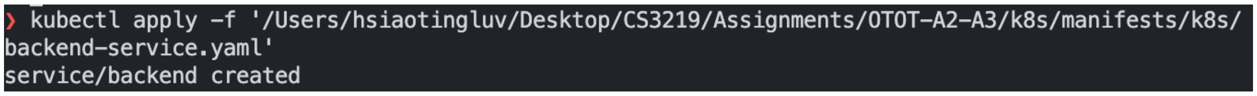
6. Add the relevant Service manifest



k8s > manifests > k8s > ! backend-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: backend
  name: backend
spec:
  selector:
    app: backend
  type: ClusterIP
  ports:
    - name: http
      port: 8080
      protocol: TCP
      targetPort: http

```

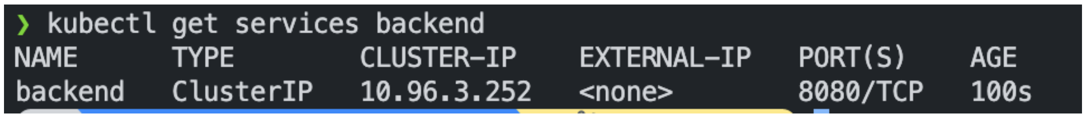
7. Apply the Service manifest



```
❯ kubectl apply -f '/Users/hsiaotingluv/Desktop/CS3219/Assignments/OTOT-A2-A3/k8s/manifests/k8s/
backend-service.yaml'
service/backend created
```

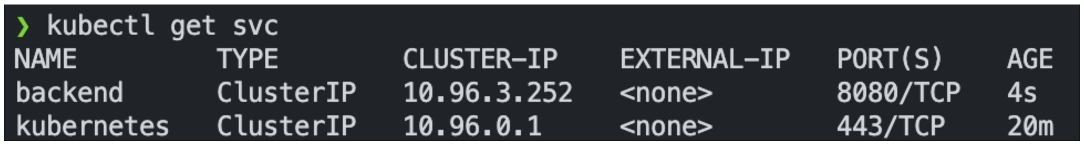
- run `kubectl apply -f backend-service.yaml`

8. Verify Service



```
❯ kubectl get services backend
NAME      TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
backend   ClusterIP   10.96.3.252   <none>        8080/TCP   100s
```

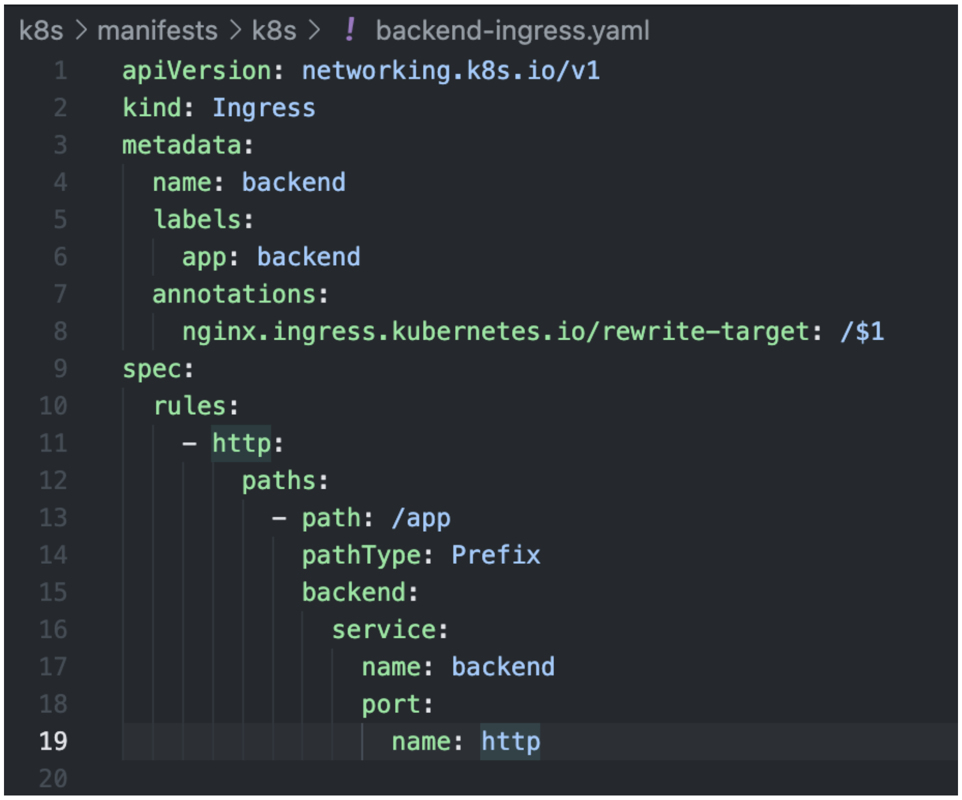
- run `kubectl get services backend`



```
❯ kubectl get svc
NAME         TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)    AGE
backend      ClusterIP   10.96.3.252   <none>        8080/TCP   4s
kubernetes   ClusterIP   10.96.0.1     <none>        443/TCP    20m
```

- run `kubectl get services`

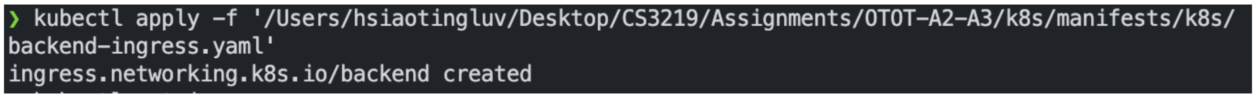
9. Add the relevant Ingress manifest



k8s > manifests > k8s > ! backend-ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: backend
  labels:
    app: backend
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - http:
        paths:
          - path: /app
            pathType: Prefix
            backend:
              service:
                name: backend
                port:
                  name: http

```

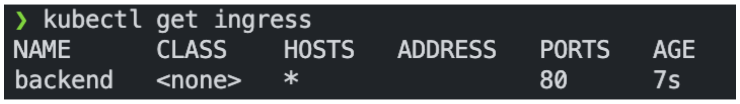
10. Apply the Ingress manifest



```
❯ kubectl apply -f '/Users/hsiaotingluv/Desktop/CS3219/Assignments/OTOT-A2-A3/k8s/manifests/k8s/
backend-ingress.yaml'
ingress.networking.k8s.io/backend created
```

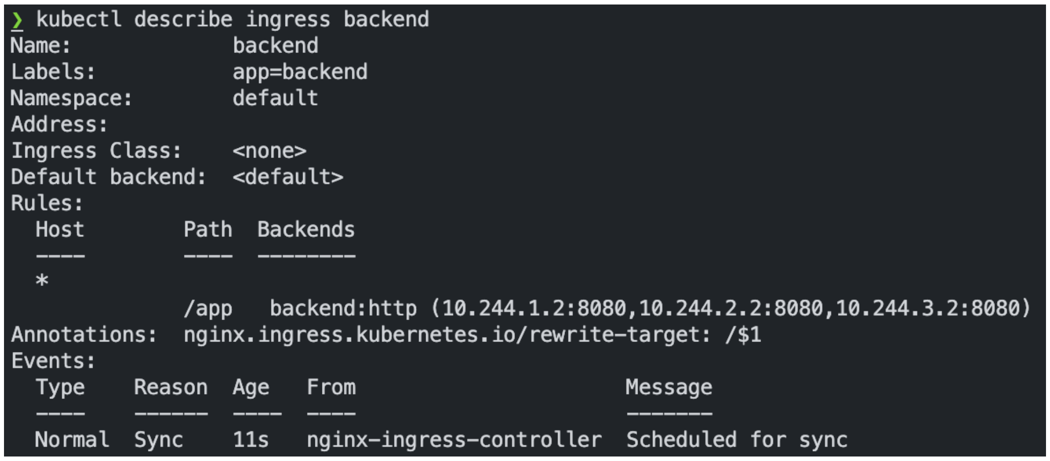
- run `kubectl apply -f 'kubectl apply -f backend-ingress.yaml`

11. Verify Ingress



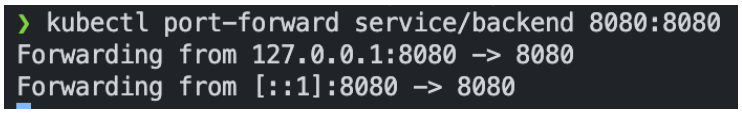
```
❯ kubectl get ingress
NAME      CLASS    HOSTS   ADDRESS   PORTS   AGE
backend   <none>   *                 80      7s
```

- run `kubectl get ingress` to check if ingress is running



```
❯ kubectl describe ingress backend
Name:             backend
Labels:           app=backend
Namespace:        default
Address:
Ingress Class:    <none>
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  *
              /app   backend:http (10.244.1.2:8080,10.244.2.2:8080,10.244.3.2:8080)
Annotations:  nginx.ingress.kubernetes.io/rewrite-target: /$1
Events:
  Type    Reason  Age   From                      Message
  ----    ------  ----  ----                      -------
  Normal  Sync    11s   nginx-ingress-controller  Scheduled for sync
```

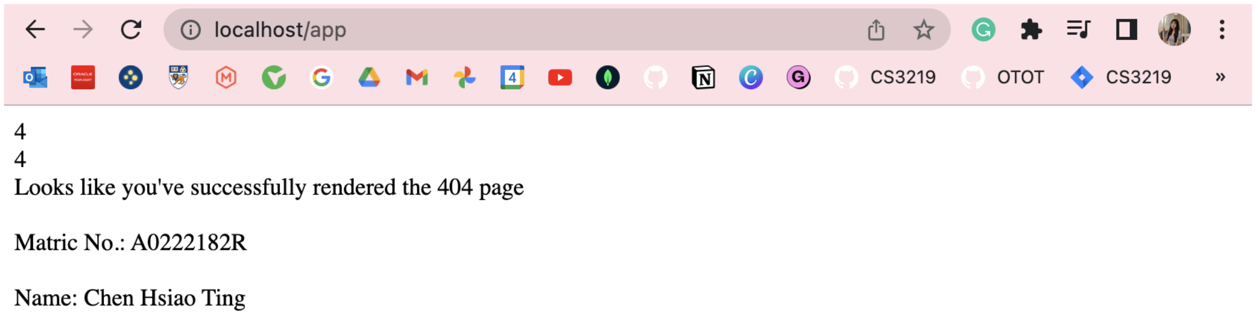
- run `kubectl describe ingress backend` to check the details



```
❯ kubectl port-forward service/backend 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

- run `kubectl port-forward service/backend 8080:8080`

12. Verify if app is running on http://localhost/app



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

Matric No.: A0222182R

Name: Chen Hsiao Ting