TIC3001 Task 1A

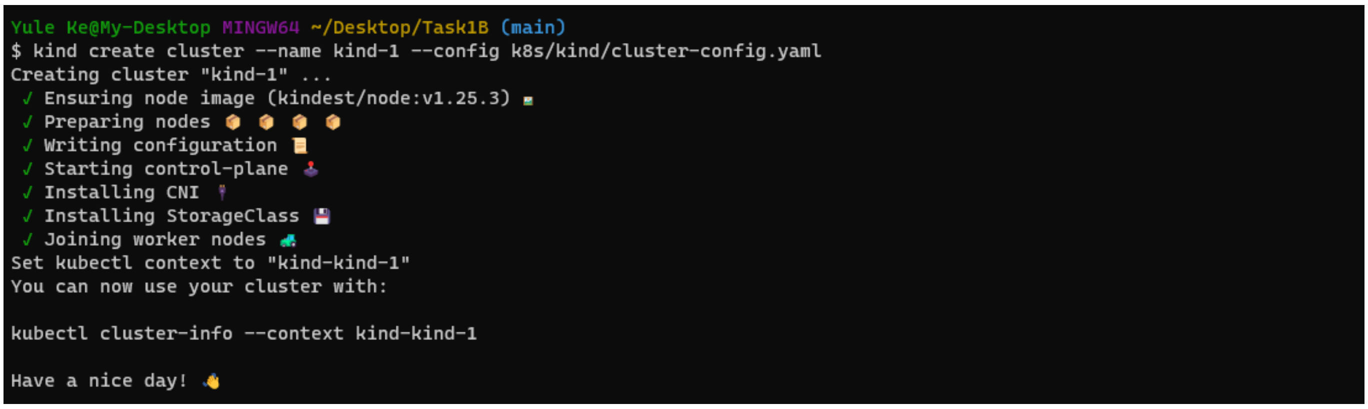
- Name: Ke Yule
- Student Number: A0211495H E0493826
- Github: https://github.com/keyule/3001-Task1B

View the markdown version for better formatting at: https://github.com/keyule/3001-Task1B/blob/main/Report/report.md

Task 1.4 - Deploy a local k8s cluster

1.4.1 Create Cluster

- `kind create cluster --name kind-1 --config k8s/kind/cluster-config.yaml`



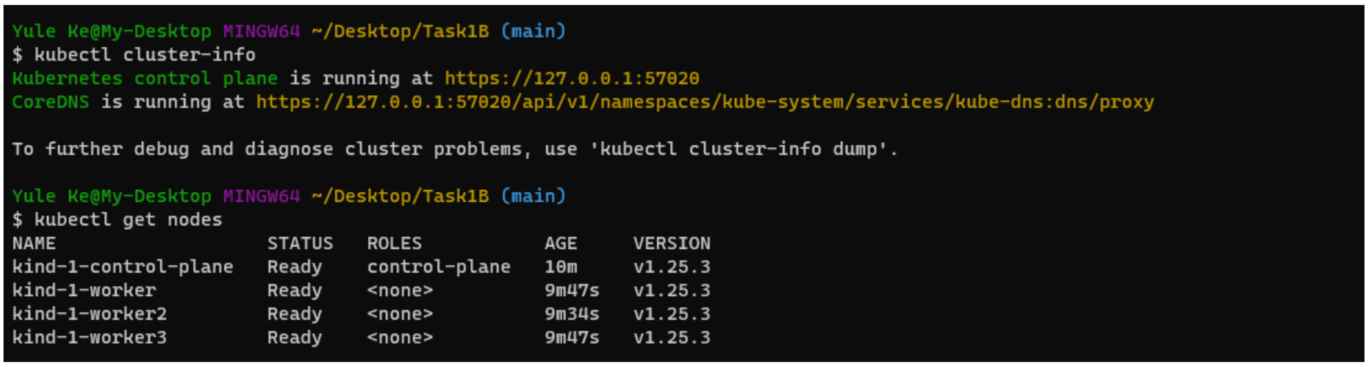
```
Yule Ke@My-Desktop MINGW64 ~/Desktop/Task1B (main)
$ kind create cluster --name kind-1 --config k8s/kind/cluster-config.yaml
Creating cluster "kind-1" ...
 ✓ Ensuring node image (kindest/node:v1.25.3) 🖼
 ✓ Preparing nodes 📦 📦 📦 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing 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! 👋
```

1.4.2 Verify Cluster

- `kubectl cluster-info`
- `kubectl get nodes`



```
Yule Ke@My-Desktop MINGW64 ~/Desktop/Task1B (main)
$ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:57020
CoreDNS is running at https://127.0.0.1:57020/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

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

Yule Ke@My-Desktop MINGW64 ~/Desktop/Task1B (main)
$ kubectl get nodes
NAME                   STATUS   ROLES           AGE     VERSION
kind-1-control-plane   Ready    control-plane   10m     v1.25.3
kind-1-worker          Ready    <none>          9m47s   v1.25.3
kind-1-worker2         Ready    <none>          9m34s   v1.25.3
kind-1-worker3         Ready    <none>          9m47s   v1.25.3
```

Task 1.5 - Deploy 1A Image

1.5.1 Build & Load Image into Cluster

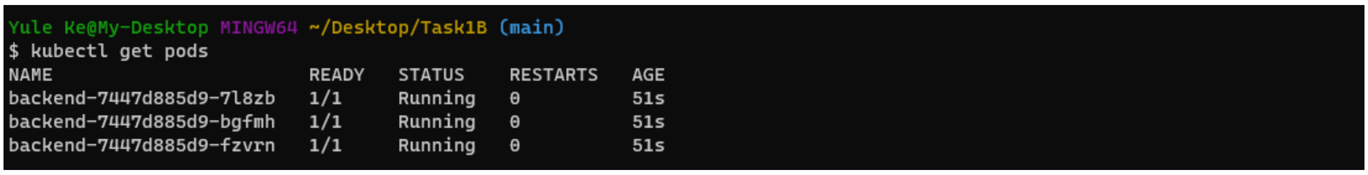
- `docker build -t custom-image:mytag ./app/.`
- `kind load docker-image custom-image:mytag --name kind-1`
- Verify image loaded: `docker exec -it kind-1-worker crictl images`

1.5.2 Create deployment

- Deployment Script: *test_deployment.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: backend
          image: custom-image:mytag
          ports:
            - name: http
              containerPort: 3000
          resources:
            limits:
              cpu: 40m
              memory: 100Mi
```

- `kubectl apply -f test_deployment.yaml`
- Verify with: `kubectl get pods`
- or `kubectl get deployment/backend --watch` *I prefer to just get pods*



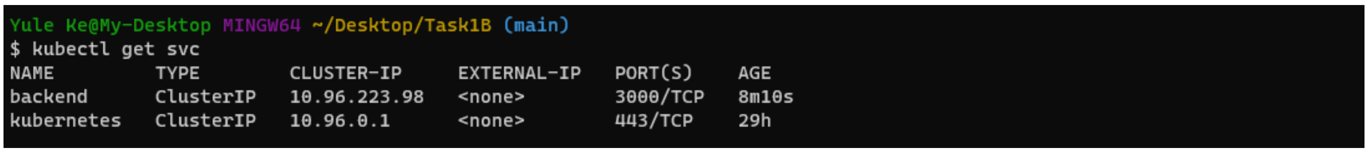
```
Yule Ke@My-Desktop MINGW64 ~/Desktop/Task1B (main)
$ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
backend-7447d885d9-7l8zb   1/1     Running   0          51s
backend-7447d885d9-bgfmh   1/1     Running   0          51s
backend-7447d885d9-fzvrn   1/1     Running   0          51s
```

1.5.3 Create Service

- Service Script: *test_service.yaml*

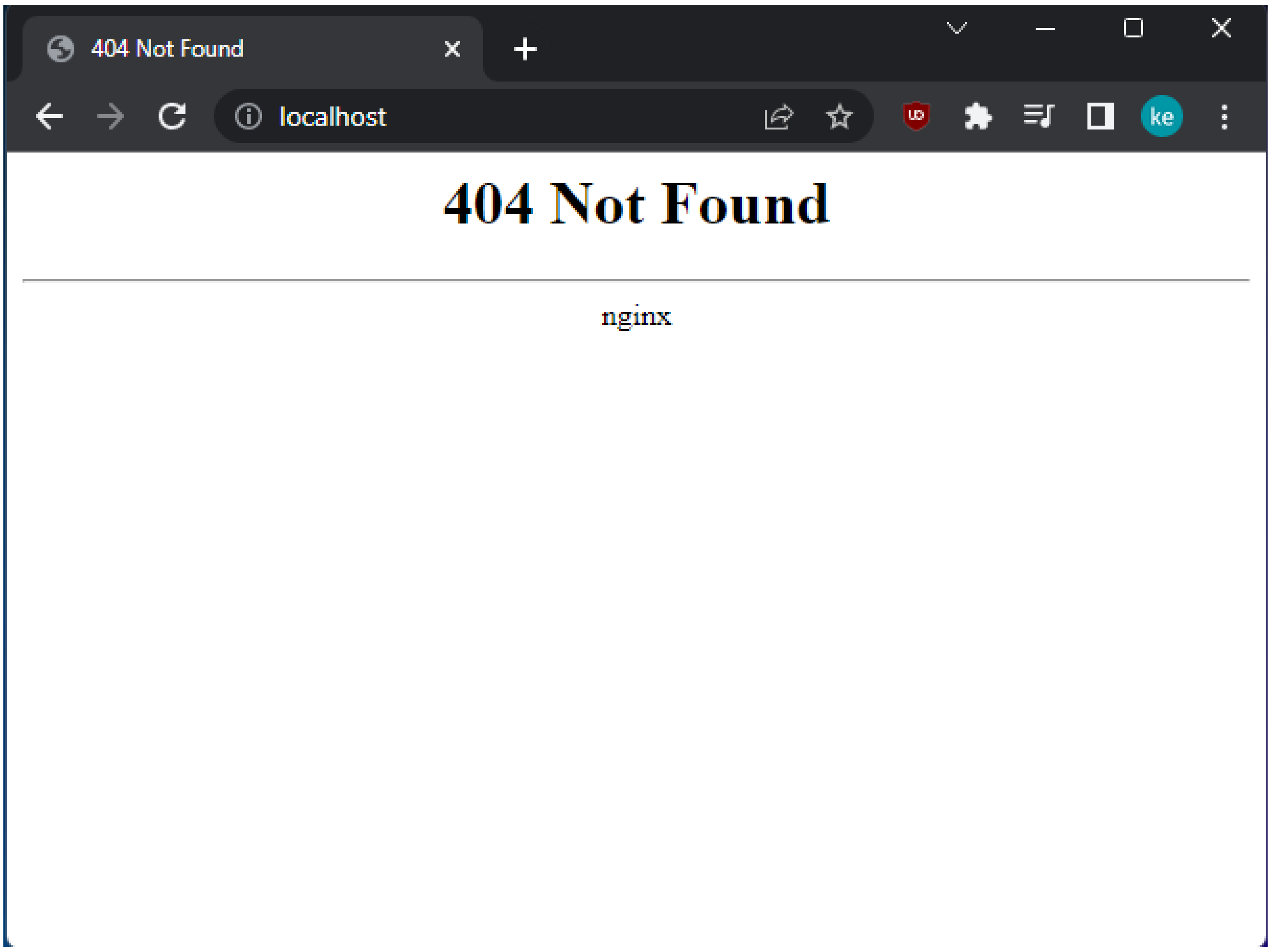
```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: backend
  name: backend
spec:
  selector:
    app: backend
  type: ClusterIP
  ports:
    - name: http
      port: 3000
      protocol: TCP
      targetPort: 3000
```

- `kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml`
- wait for it to be ready with `kubectl -n ingress-nginx get deploy -w`
- `kubectl apply -f test_service.yaml`
- Verify with: `kubectl get svc`



```
Yule Ke@My-Desktop MINGW64 ~/Desktop/Task1B (main)
$ kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
backend      ClusterIP   10.96.223.98   <none>        3000/TCP   8m10s
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP    29h
```

- Localhost should return an nginx 404 as well



404 Not Found

nginx

Task 1.6 - Create Ingress Controller

1.6.1 Create Controller

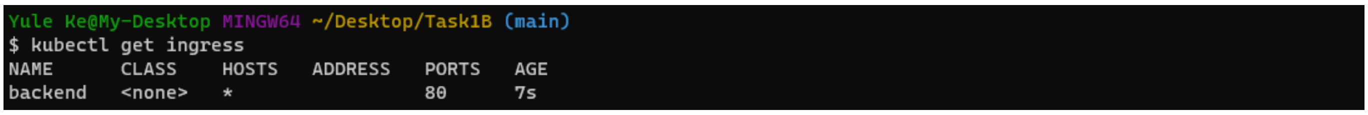
- label nodes as ingress ready
- `kubectl label node <Node Name> ingress-ready=true`
- Ingress Script: *test_ingressobject.yaml*

```
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
```

- kubectl apply -f test_ingressobject.yaml

1.6.2 Verify Ingress

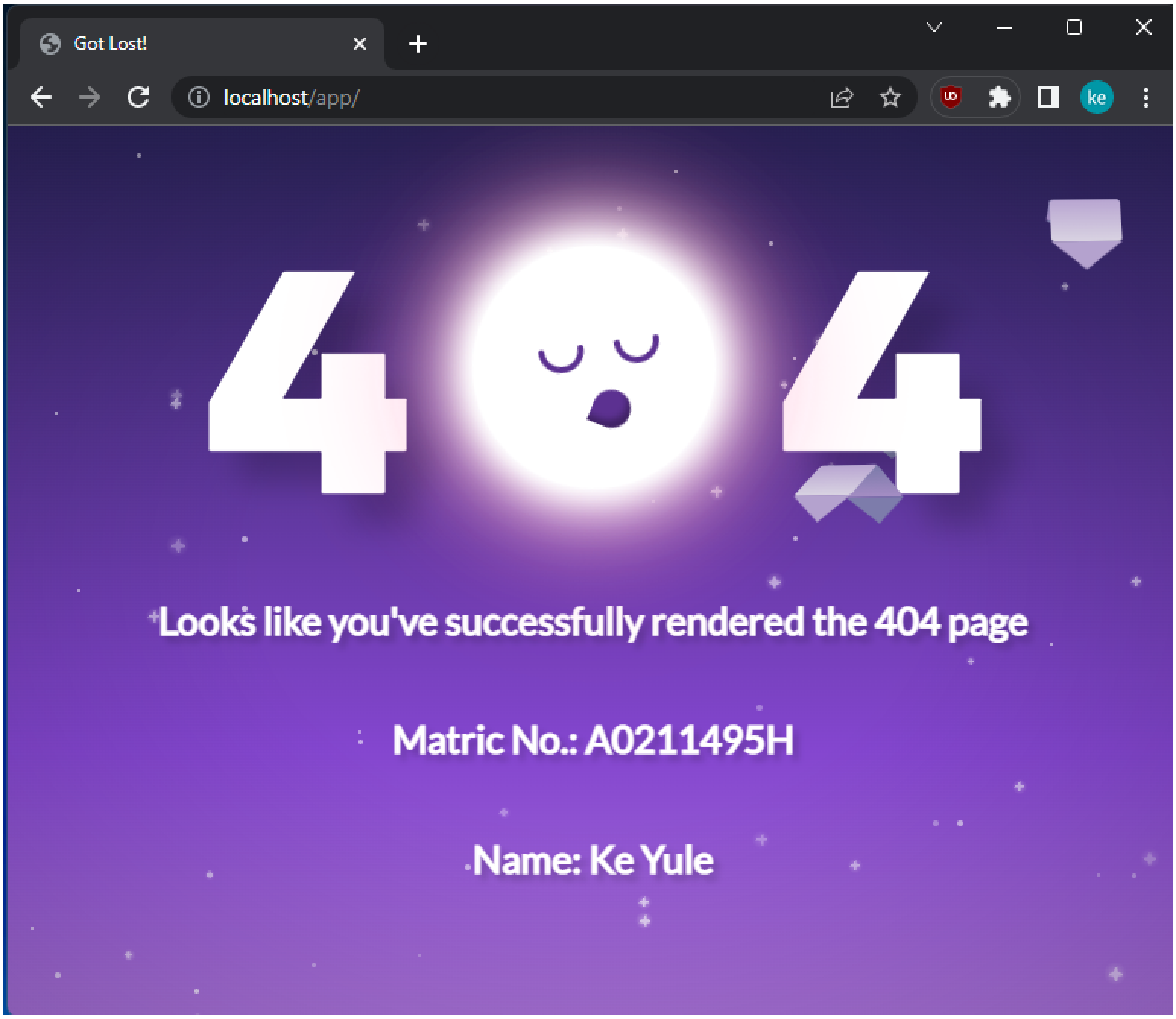
- `kubectl get ingress`



```
Yule Ke@My-Desktop MINGW64 ~/Desktop/Task1B (main)
$ kubectl get ingress
NAME      CLASS    HOSTS   ADDRESS   PORTS   AGE
backend   <none>   *                 80      7s
```

Done!

- Task 1A should be rendered on `http://localhost/app/`



Looks like you've successfully rendered the 404 page

Matric No.: A0211495H

Name: Ke Yule

> *Note: Only works with the extra `/` at the end. I have no idea why. Its most probably how I set up my rules in ingress object. If I remove the `/app` and set my rule as just `/?(.*)` it would work if i visit localhost.*

Appendix

One-Click.sh File

```
# this first line just prints a banner
base64 -d
<<<"H4sIAAAAAAAAA9NThgI9IIBSejARmKienh6cAVfPy4UsCaSh0pg8iCYEV1kZRS9cCsUiOEdPD7de
JBMQblZG0orHzcpwURQfkGgvdv+S7GYUCbAgkpt5uXTJBwCpuiaj4QEAAA==" | gunzip

echo -e "\nBuilding docker Image ... " ;

docker build -t custom-image:mytag ./app/. ;

echo -e "\nSetting up kind Cluster ... " ;

kind create cluster --name kind-1 --config k8s/kind/cluster-config.yaml ;
```

```
echo -e "\nLoading image into cluster ... " ;

kind load docker-image custom-image:mytag --name kind-1;

echo -e "\nApplying deployment manifest ... " ;

kubectl apply -f test_deployment.yaml ;

echo -e "\nApplying nginx-ingress-controller ... " ;

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


while [[ "$(kubectl -n ingress-nginx get deploy ingress-nginx-controller | tail -n
1 | awk '{print $2}')" != "1/1" ]]; do
  echo "Waiting for deployment to become available... (current status: $(kubectl -
n ingress-nginx get deploy ingress-nginx-controller | tail -n 1 | awk '{print
$2}'))"
  sleep 5
done

echo "Deployment is ready!"


echo -e "\nLabeling workers as ingress ready ... " ;

kubectl label node kind-1-worker2 ingress-ready=true;
kubectl label node kind-1-worker3 ingress-ready=true;

echo -e "\nApplying service ... " ;

kubectl apply -f test_service.yaml ;

echo -e "\nApplying ingress Object ... " ;

kubectl apply -f test_ingressobject.yaml ;

echo -e "\nShould be done ... " ;

while ! curl -s -I localhost/app/ | grep "HTTP/1.1 200 OK" >/dev/null; do
  echo "Waiting for the webpage to become available..."
  sleep 5
done
echo "Webpage is up!";
echo -e "Opening localhost on default browser ... " ;

start "http://localhost/app/" ;
```