Lab 7 - Deploying to Kubernetes

Setup Kubernetes Cluster

Setup a 3 Node Kubernetes Cluster following these instructions : Lab K801 - Setting up Kubernetes Cluster - Kubernetes Tutorial with CKA/CKAD Prep

validate your environment using

```
kubectl get nodes
```

[sample output]

```
NAME                 STATUS   ROLES           AGE   VERSION
kind-control-plane   Ready    control-plane   16d   v1.32.2
kind-worker          Ready    <none>          16d   v1.32.2
kind-worker2         Ready    <none>          16d   v1.32.2
```

You should see 3 nodes listed and running.

You could further make sure that the pods are running using

```
kubectl get pods -A
```

Deploy Streamlit App to Kubernetes

```
kubectl create deployment streamlit --image=initcron/streamlit:v1 --port=8501
```

validate

```
kubectl get deploy
kubectl describe deploy
kubectl get pods
```

Try scaling the deployment

```
kubectl scale deploy streamlit --replicas=3
kubectl get all
```

You could also observe it with the visualizer http://localhost:32000/\#scale=2

once done, scale it back to one

```
kubectl scale deploy streamlit --replicas=1
```

Expose the App with a Service

Create a service for streamlit app

```
kubectl create service nodeport streamlit --tcp=8501 --node-port=30000
```

validate

```
kubectl get svc,ep,pods -o wide
kubectl describe svc streamlit
```

You could also access it on http://localhost:30000/

Setup Model Inference on Kubernetes

Similar to the streamlit app you could create deployment and service to setup model inference.

```
kubectl create deployment model --image=initcron/house-price-model:latest --
port=8000

kubectl create service nodeport model --tcp=8000 --node-port=30100
```

validate

```
kubectl get all
```

You could access the fastapi interface using http://localhost:30100/docs

Connecting Streamlit to Model Backend

Since Streamlit is been configured to connect to the model with hostname `model`

e.g.

```
        try:
                # Get API endpoint from environment variable or use default
                api_endpoint = os.getenv("API_URL", "http://model:8000")
                predict_url = f"{api_endpoint.rstrip('/')}/predict"

                st.write(f"Connecting to API at: {predict_url}")
```

Your app should be fully functional at this time.

Congratulations, your Model Inference is deployed to Kubernetes along with the frontend interface.

Auto Generating Kubernetes Manifests

While you created the kubernetes resources with commands, its always recommended to use YAML manifests to manage these resources

```
cd house-price-predictor/deployment/kubernetes
```

```
#replace xxxxxx with your username
kubectl create deployment streamlit --image=xxxxxx/streamlit:latest --
port=8501 -o yaml --dry-run=client > streamlit-deploy.yaml
```

```
kubectl create service nodeport streamlit --tcp=8501 --node-port=30000 -o yaml
--dry-run=client > streamlit-svc.yaml
```

```
#replace xxxxxx with your username
kubectl create deployment model --image=xxxxxx/house-price-model:latest --
port=8000 --dry-run=client -o yaml > model-deploy.yaml
```

```
kubectl create service nodeport model --tcp=8000 --node-port=30100 --dry-
run=client -o yaml > model-svc.yaml
```

File : `kustomization.yaml`

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
```

```
  - model-deploy.yaml
  - model-svc.yaml
  - streamlit-deploy.yaml
  - streamlit-svc.yaml
```

Delete the previous deployments

```
kubectl delete deploy,svc model streamlit
```

and apply with new manifests as

```
cd house-price-predictor
kubectl apply -k deployment/kubernetes
```

#courses/mlops