Lab10

*I use wsl2 to study this lab.

Quickstart

1. Install minikube and Kubernetes on wsl:

```
undefined@Undefined:~$ curl -LO https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  133M  100  133M    0     0  3222k      0  0:00:42  0:00:42 --:--:-- 10.0M
undefined@Undefined:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
[sudo] password for undefined:
undefined@Undefined:~$ ls
```

   check:

```
undefined@Undefined:~$ kubectl version --client
Client Version: v1.32.2
Kustomize Version: v5.5.0
```

```
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass

❗  /usr/local/bin/kubectl is version 1.32.2, which may have incompatibilities with Kubernetes 1.34.0.
    ▪ Want kubectl v1.34.0? Try 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

```
undefined@Undefined:~$ kubectl get pods -A
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   coredns-66bc5c9577-gjbj5           1/1     Running   0          102s
kube-system   etcd-minikube                      1/1     Running   0          108s
kube-system   kube-apiserver-minikube            1/1     Running   0          110s
kube-system   kube-controller-manager-minikube   1/1     Running   0          108s
kube-system   kube-proxy-5lp9f                   1/1     Running   0          103s
kube-system   kube-scheduler-minikube            1/1     Running   0          108s
kube-system   storage-provisioner                1/1     Running   0          106s
```

2. Install helm:

```
undefined@Undefined:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-4
undefined@Undefined:~$ ls
get_helm.sh  snap
undefined@Undefined:~$ chmod 700 get_helm.sh
undefined@Undefined:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v4.0.1-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
[sudo] password for undefined:
helm installed into /usr/local/bin/helm
```

3. Install KServe Quickstart Environment

   curl -s

   "https://raw.githubusercontent.com/kserve/kserve/master/hack/quick_install.sh" |

   bash -s -- -r

```
undefined@Undefined:~$ curl -s "https://raw.githubusercontent.com/kserve/kserve/master/hack/quick_
stall.sh" | bash -s -- -r
WARNING: version difference between client (1.32) and server (1.34) exceeds the supported minor ve
ion skew of +/-1
Installing Gateway API CRDs ...
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/gatewayclasses.gateway.networking.k8s.io created
Warning: unrecognized format "int32"
customresourcedefinition.apiextensions.k8s.io/gateways.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/grpcroutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/httproutes.gateway.networking.k8s.io created
customresourcedefinition.apiextensions.k8s.io/referencegrants.gateway.networking.k8s.io created
"istio" has been added to your repositories
I1208 20:52:16.910210    3752 warnings.go:110] "Warning: unrecognized format \"int32\""
I1208 20:52:16.925145    3752 warnings.go:110] "Warning: unrecognized format \"int32\""
I1208 20:52:16.946900    3752 warnings.go:110] "Warning: unrecognized format \"int32\""
I1208 20:52:16.953090    3752 warnings.go:110] "Warning: unrecognized format \"int32\""
I1208 20:52:16.953242    3752 warnings.go:110] "Warning: unrecognized format \"double\""
I1208 20:52:16.983132    3752 warnings.go:110] "Warning: unrecognized format \"int32\""
I1208 20:52:17.009512    3752 warnings.go:110] "Warning: unrecognized format \"int32\""
I1208 20:52:17.009582    3752 warnings.go:110] "Warning: unrecognized format \"double\""
I1208 20:52:17.009619    3752 warnings.go:110] "Warning: unrecognized format \"binary\""
I1208 20:52:17.095287    3752 warnings.go:110] "Warning: unrecognized format \"int32\""
I1208 20:52:17.095363    3752 warnings.go:110] "Warning: unrecognized format \"double\""
NAME: istio-base
LAST DEPLOYED: Mon Dec  8 20:52:15 2025
NAMESPACE: istio-system
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
TEST SUITE: None
NOTES:
Istio base successfully installed!

To learn more about the release, try:
  $ helm status istio-base -n istio-system
  $ helm get all istio-base -n istio-system
```

4. ! Webhook problem: solved by helm upgrade kserve:

```
Internal error occurred: failed calling webhook "clusterservingruntime.kserve-webhook-server.validat
or": failed to call webhook: Post "https://kserve-webhook-server-service.kserve.svc:443/validate-ser
ving-kserve-io-v1alpha1-clusterservingruntime?timeout=10s": EOF
Internal error occurred: failed calling webhook "clusterservingruntime.kserve-webhook-server.validat
or": failed to call webhook: Post "https://kserve-webhook-server-service.kserve.svc:443/validate-ser
ving-kserve-io-v1alpha1-clusterservingruntime?timeout=10s": EOF
Internal error occurred: failed calling webhook "clusterservingruntime.kserve-webhook-server.validat
or": failed to call webhook: Post "https://kserve-webhook-server-service.kserve.svc:443/validate-ser
ving-kserve-io-v1alpha1-clusterservingruntime?timeout=10s": EOF
Internal error occurred: failed calling webhook "clusterservingruntime.kserve-webhook-server.validat
or": failed to call webhook: Post "https://kserve-webhook-server-service.kserve.svc:443/validate-ser
ving-kserve-io-v1alpha1-clusterservingruntime?timeout=10s": EOF
Internal error occurred: failed calling webhook "clusterservingruntime.kserve-webhook-server.validat
or": failed to call webhook: Post "https://kserve-webhook-server-service.kserve.svc:443/validate-ser
ving-kserve-io-v1alpha1-clusterservingruntime?timeout=10s": EOF
undefined@Undefined:~$ kubectl get pods -n kserve
NAME                                          READY   STATUS    RESTARTS   AGE
kserve-controller-manager-7bcd48cd6d-tcp96    1/2     Running   0          41s
undefined@Undefined:~$ kubectl get svc -n kserve
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kserve-controller-manager-service   ClusterIP   10.97.191.206   <none>        8443/TCP   56s
kserve-webhook-server-service       ClusterIP   10.106.63.40    <none>        443/TCP    56s
undefined@Undefined:~$ kubectl create namespace kserve || true
Error from server (AlreadyExists): namespaces "kserve" already exists
undefined@Undefined:~$ helm upgrade kserve oci://ghcr.io/kserve/charts/kserve --version v0.16.0 \
  --namespace kserve \
  --install \
  --set-string kserve.controller.deploymentMode="RawDeployment" \
  --wait
Pulled: ghcr.io/kserve/charts/kserve:v0.16.0
Digest: sha256:46d78a58fff65902213a7254ec16520286baa81d33340c4a03f5263d846e2124
Release "kserve" has been upgraded. Happy Helming!
NAME: kserve
LAST DEPLOYED: Mon Dec  8 20:55:53 2025
NAMESPACE: kserve
STATUS: deployed
REVISION: 2
DESCRIPTION: Upgrade complete
TEST SUITE: None
undefined@Undefined:~$ kubectl get pods -n kserve
NAME                                          READY   STATUS    RESTARTS   AGE
kserve-controller-manager-7bcd48cd6d-tcp96    2/2     Running   0          3m1s
```

5. Installed successfully

```
[SUCCESS] KServe manifests installed successfully!
==========================================
✅ Installation completed successfully!
==========================================
```

6. Check:

```
undefined@Undefined:~/kserve$ kubectl get pods -n kserve
NAME                                         READY   STATUS    RESTARTS   AGE
kserve-controller-manager-7c77df7f5d-cw79k   2/2     Running   0          5m21s
kserve-localmodel-controller-manager-5d5b4c74b8-7sd9q   1/1     Running   0          5m21s
llmisvc-controller-manager-798dbc7dd4-flmgs  1/1     Running   0          5m21s
```

Deploy Your First Predictive Inference Service

1. Create a namespace

   kubectl create namespace kserve-test

```
undefined@Undefined:~/kserve$ kubectl create namespace kserve-test
namespace/kserve-test created
```

2. Create an InferenceService to deploy the Iris model. This model will be served using KServe's Scikit-learn runtime for optimized performance.

```
undefined@Undefined:~/kserve$ kubectl apply -n kserve-test -f - <<EOF
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: "sklearn-iris"
  namespace: kserve-test
spec:
  predictor:
    model:
      modelFormat:
        name: sklearn
      storageUri: "gs://kfserving-examples/models/sklearn/1.0/model"
      resources:
        requests:
          cpu: "100m"
          memory: "512Mi"
        limits:
          cpu: "1"
          memory: "1Gi"
EOF
inferenceservice.serving.kserve.io/sklearn-iris created
```

3. Check InferenceService status

   kubectl get inferenceservices sklearn-iris -n kserve-test

```
undefined@Undefined:~$ kubectl get inferenceservices sklearn-iris -n kserve-test
NAME          URL                                        READY   PREV   LATEST   PREVROLLEDOUTREVISION   LATESTREADY
REVISION          AGE
sklearn-iris   http://sklearn-iris.kserve-test.example.com   True           100                             sklearn-iri
s-predictor-00001   6m59s
```

4. Determine the ingress IP and ports

   kubectl get svc istio-ingressgateway -n istio-system

```
undefined@Undefined:~$ kubectl get svc istio-ingressgateway -n istio-system
NAME                   TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)                                      AGE
istio-ingressgateway   LoadBalancer   10.98.55.91   <pending>     15021:32615/TCP,80:32074/TCP,443:32733/TCP   30m
```

5. Port forward

```
undefined@Undefined:~$ INGRESS_GATEWAY_SERVICE=$(kubectl get svc -n istio-system --selector="app=istio-ingressgateway" -
o jsonpath='{.items[0].metadata.name}')
kubectl port-forward -n istio-system svc/${INGRESS_GATEWAY_SERVICE} 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```

6. Open another terminal, and enter the following to perform inference:
   export INGRESS_HOST=localhost
   export INGRESS_PORT=8080

   SERVICE_HOSTNAME=$(kubectl get inferenceservice sklearn-iris -n kserve-test -o jsonpath='{.status.url}' | cut -d "/" -f 3)

7. prepare your inference input request inside a file:

```
undefined@Undefined:~$ cat <<EOF > "./iris-input.json"
{
  "instances": [
    [6.8,  2.8,  4.8,  1.4],
    [6.0,  3.4,  4.5,  1.6]
  ]
}
EOF
```

8. Depending on your setup, use one of the following commands to curl the InferenceService:
   SERVICE_HOSTNAME=$(kubectl get inferenceservice sklearn-iris -n kserve-test -o jsonpath='{.status.url}' | cut -d "/" -f 3)
   curl -sS \
   -H "Host: ${SERVICE_HOSTNAME}" \
   -H "Content-Type: application/json" \
   "http://localhost:8080/v1/models/sklearn-iris:predict" \
   -d @iris-input.json ; echo

```
{"predictions":[1,1]}uncurl -sS \defined:~$ curl -sS \
  -H "Host: ${SERVICE_HOSTNAME}" \
  -H "Content-Type: application/json" \
  "http://localhost:8080/v1/models/sklearn-iris:predict" \
  -d @iris-input.json ; echo
{"predictions":[1,1]}
```

Success!