1.Install and configure Kserve

**Command:**

kubectl apply -f serving-crds.yaml

kubectl apply -f serving-core.yaml

kubectl apply -f istio.yaml

kubectl apply -f net-istio.yaml

kubectl apply -f cert-manager.yaml

kubectl get pods -A

helm install kserve-crd oci://ghcr.io/kserve/charts/kserve-crd \

--version v0.15.2 \

--namespace kserve \

--create-namespace \

--wait

helm install kserve oci://ghcr.io/kserve/charts/kserve \

--version v0.15.2 \

--namespace kserve \

--create-namespace \

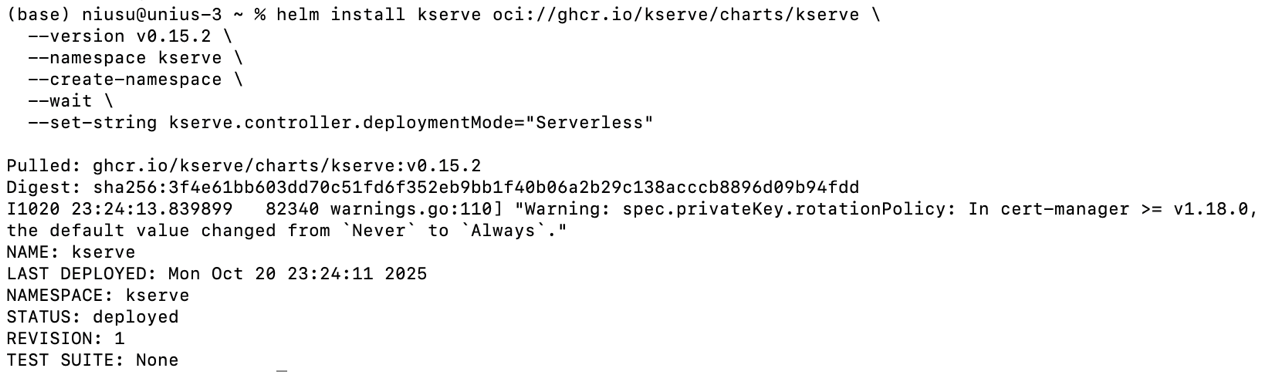
--wait \

--set-string kserve.controller.deploymentMode="Serverless"

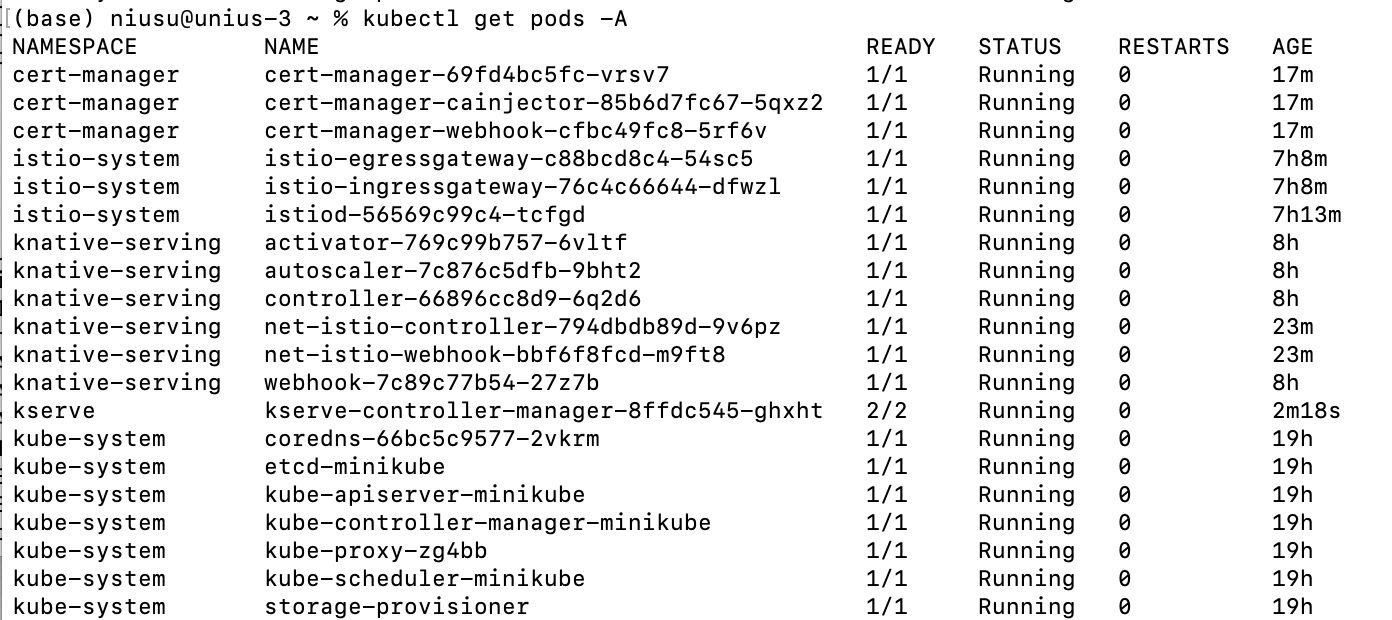
kubectl get crd | grep kserve

**Note:**

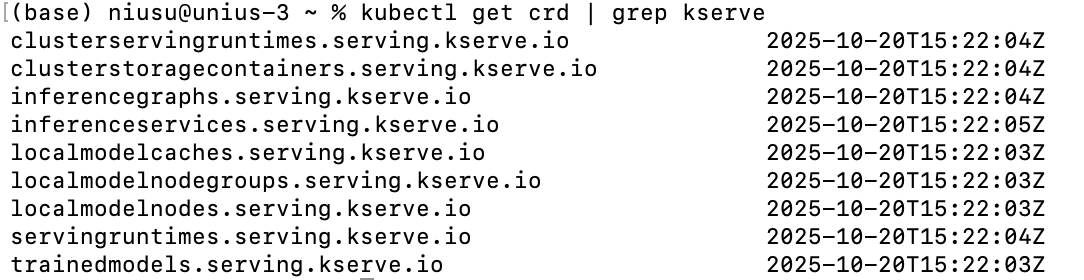
Install and configure KServe along with Knative, Istio, and Cert-Manager to set up the environment required for deploying and managing machine learning models.

******

***Figure1: Installing kserve***

******

***Figure2: Status of all pods***



***Figure3: Kserve crd registered***

2. KServe InferenceService

**Command:**

kubectl get services -n kserve

kubectl get inferenceservice sklearn-iris

cat <<EOF | kubectl apply -f -

apiVersion: serving.kserve.io/v1beta1

kind: InferenceService

metadata:

name: sklearn-iris

namespace: default

spec:

predictor:

model:

modelFormat:

name: sklearn

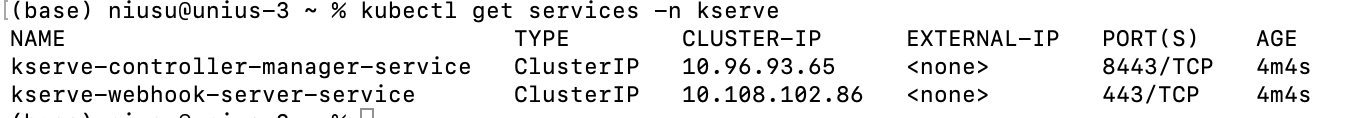
storageUri: gs://kfserving-examples/models/sklearn/1.0/model

EOF

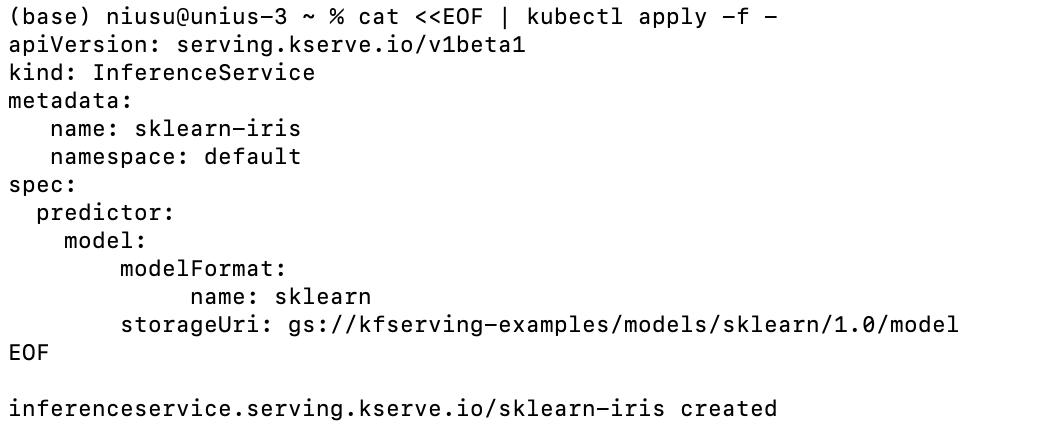
kubectl get inferenceservice sklearn-iris

**Note:**

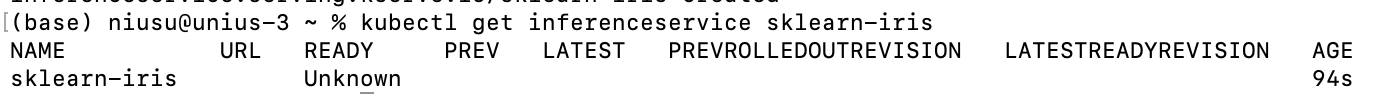
Create an InferenceService resource that defines the model type, format, and storage path, and deploy it on KServe for serving predictions.



***Figure4: Running service***



***Figure5: Create a machine learning model reasoning service***



***Figure6: Inferenceservice running status***

3. Monitor the status of the service

**Command:**

kubectl describe pod sklearn-iris-predictor-00001-deployment-67966646b-f7tbx

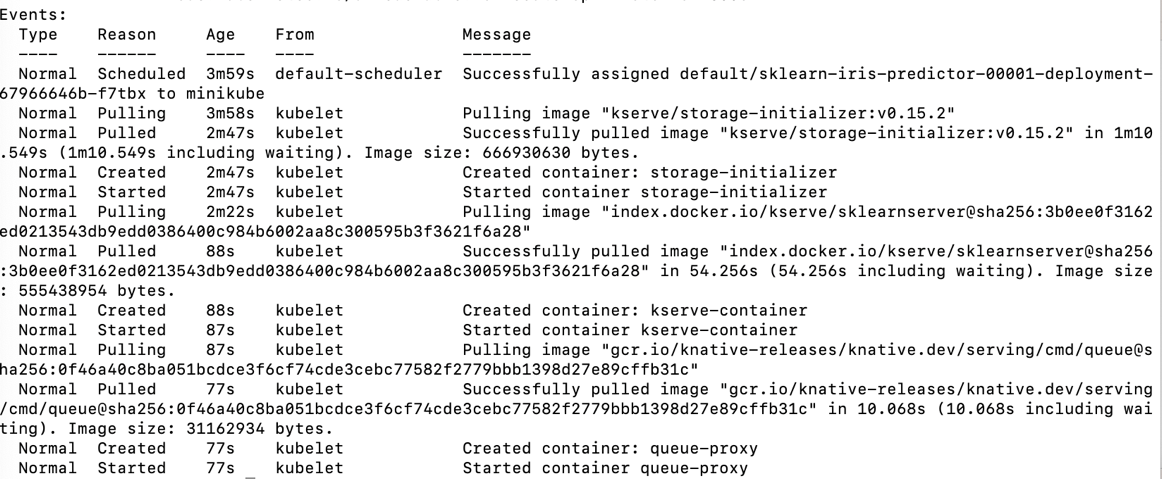
kubectl get inferenceservice sklearn-iris

**Note:**

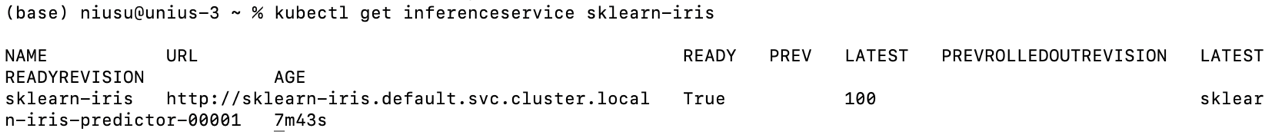
Check the running status of the model Pod and InferenceService to verify that the model has been loaded correctly and the service is ready to handle inference requests.



***Figure7: Check the model reasoning pod name***



***Figure8: Check the pod***



***Figure9: Check the status of reasoning model***

4. Reach the InferenceService from outside the cluster

**Command:**

kubectl get svc istio-ingressgateway -n istio-system

kubectl get svc --namespace istio-system \

--selector="app=istio-ingressgateway" \

--output jsonpath='{.items[0].metadata.name}'

INGRESS\_GATEWAY\_SERVICE=$(kubectl get svc --namespace istio-system \

--selector="app=istio-ingressgateway" \

--output jsonpath='{.items[0].metadata.name}')

echo $INGRESS\_GATEWAY\_SERVICE

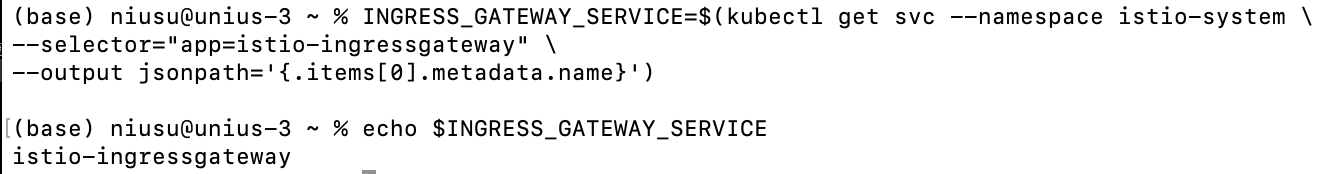
kubectl port-forward --namespace istio-system svc/${INGRESS\_GATEWAY\_SERVICE} 8080:80

**Note:**

Configure Istio Ingress Gateway and use port forwarding to allow external clients to access the deployed InferenceService from outside the Kubernetes cluster.

******

***Figure10: Get the name of the entrance service***



***Figure11: Establish port forwarding***

5. Send inference requests and interpret results

**Command:**

export INGRESS\_HOST=localhost

export INGRESS\_PORT=8080

kubectl get inferenceservice sklearn-iris -o jsonpath='{.status.url}'

kubectl get inferenceservice sklearn-iris -o jsonpath='{.status.url}' | cut -d "/" -f 3

SERVICE\_HOSTNAME=$(kubectl get inferenceservice sklearn-iris -o jsonpath='{.status.url}' | cut -d "/" -f 3)

curl -v -H "Host: ${SERVICE\_HOSTNAME}" -H "Content-Type: application/json" "http://${INGRESS\_HOST}:${INGRESS\_PORT}/v1/models/sklearn-iris:predict" -d @./iris-input.json

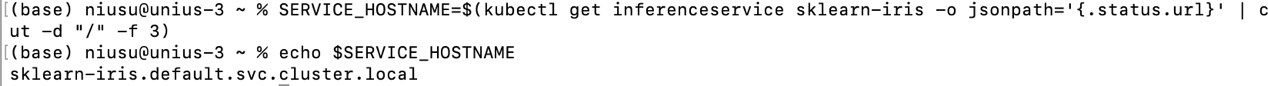
kubectl port-forward -n istio-system svc/knative-local-gateway 8080:80

curl -v -H "Host: sklearn-iris.default.svc.cluster.local" -H "Content-Type: application/json" -d '{"instances": [[5.1, 3.5, 1.4, 0.2]]}' http://localhost:8080/v1/models/sklearn-iris:predict

**Note:**

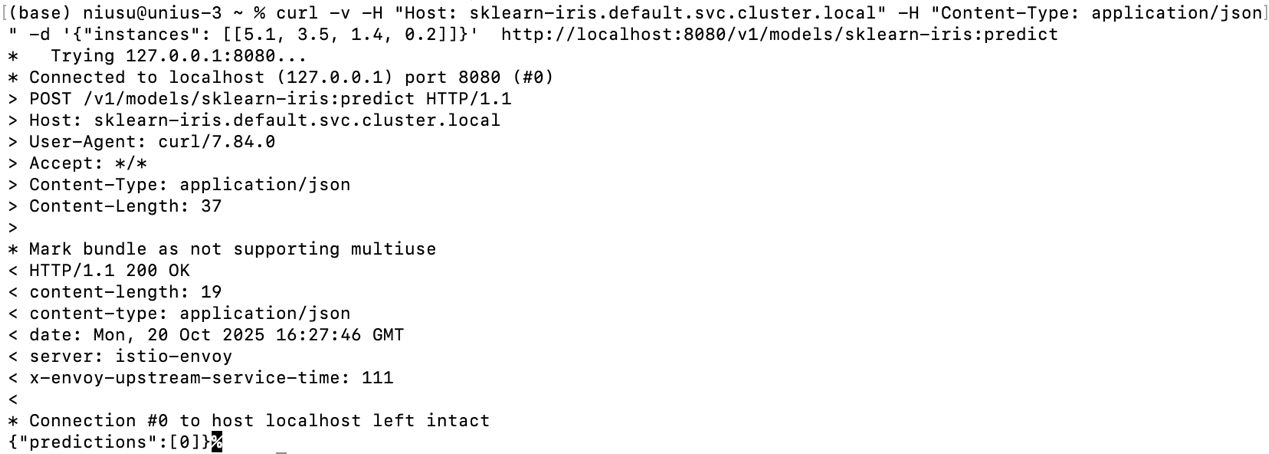
Send test input data to the InferenceService using HTTP requests (via curl) and interpret the returned JSON results to validate the model’s predictions.

 ***Figure12: Get InferenceService URL***



***Figure13: Get the hostname***

Predicting Result: 0



***Figure14: Output of the model***