Lab 10

Experiment Objectives

The purpose of this experiment is to gain practical experience in deploying a predictive machine learning model using KServe on a local Kubernetes environment. The specific objectives include understanding the installation and configuration of KServe, deploying a predictive model as an InferenceService, monitoring the service status, resolving common issues related to certificate management, exposing the service externally, and sending inference requests to verify model functionality.

Experimental Environment

The experiment was conducted on a local Windows machine using the following environment:

- **Operating System:** Windows 10
- **Kubernetes:** Minikube
- **KServe Version:** 0.13.0
- **Cert-Manager Version:** 1.12.3
- **Predictive Model:** Scikit-learn Iris classification model
- **Tools:** kubectl for Kubernetes management, curl for sending inference requests

---

Experimental Procedure

1. **Start Minikube Cluster**
   A local Kubernetes cluster was started using Minikube, and the cluster status was checked to ensure all components were running.
2. **Install Cert-Manager**
   Cert-manager was installed to handle TLS certificates for KServe, and its pods were verified to be running successfully.
3. **Install KServe**
   KServe manifests were applied to deploy the controller and webhook, preparing the environment for predictive model deployment.
4. **Create Self-Signed Issuer and Certificate**

A self-signed issuer and a corresponding certificate were created to resolve TLS issues with the KServe webhook.

5. **Deploy Predictive Model**
   An InferenceService was created for the scikit-learn Iris model, and the readiness of the service and pods was confirmed.
6. **Expose Service Externally**
   The service endpoint was exposed through Minikube's ingress gateway to allow inference requests from outside the cluster.
7. **Send Inference Requests**
   Sample input data was sent to the model, and the predictions were received and verified to be correct.

Deployment of the Predictive Model

Once KServe was fully operational, an InferenceService was created for a scikit-learn Iris classification model. The deployment was monitored by checking the status of the InferenceService and its associated pods to ensure readiness. Logs and events were observed to confirm that the model server had started successfully and was ready to handle requests.

The service was then exposed externally using Minikube's ingress gateway to allow inference requests from outside the cluster. The endpoint was verified and tested by sending sample input data to the model.

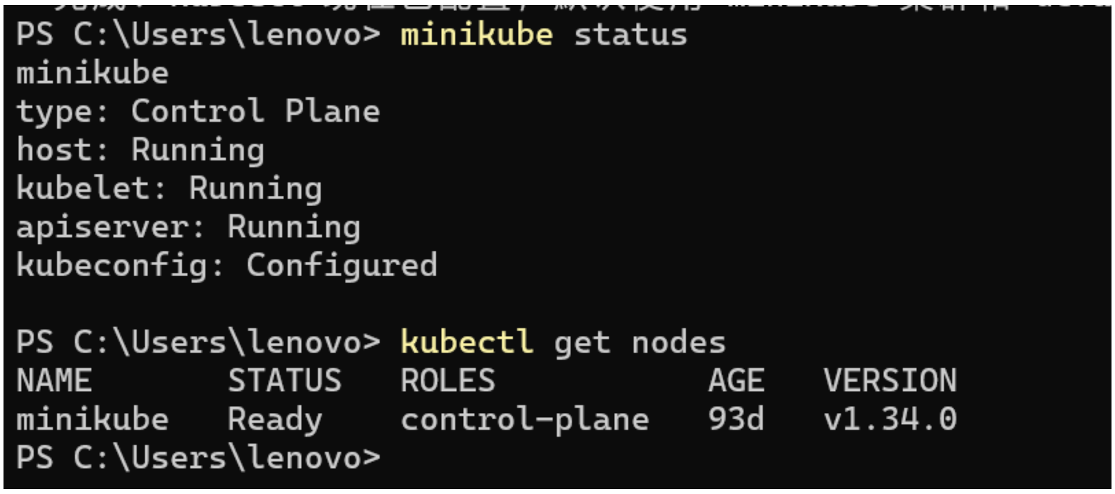
Experimental Results

The experiment successfully achieved all objectives. KServe and cert-manager were installed and configured correctly. The TLS certificates for the webhooks were generated and loaded successfully, allowing secure communication within the cluster.

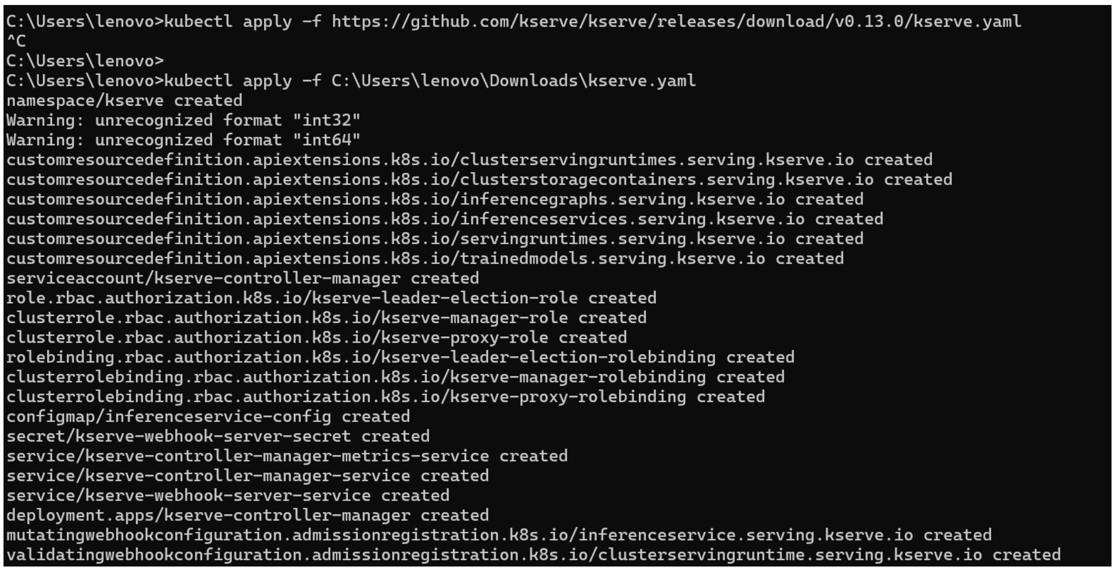
The predictive model was deployed as an InferenceService and reached a ready state. Inference requests sent to the service returned correct predictions for the test input, demonstrating that the model server was functioning as expected. Logs confirmed that no errors occurred during service startup, and external access through the Minikube ingress gateway worked properly.

Conclusion

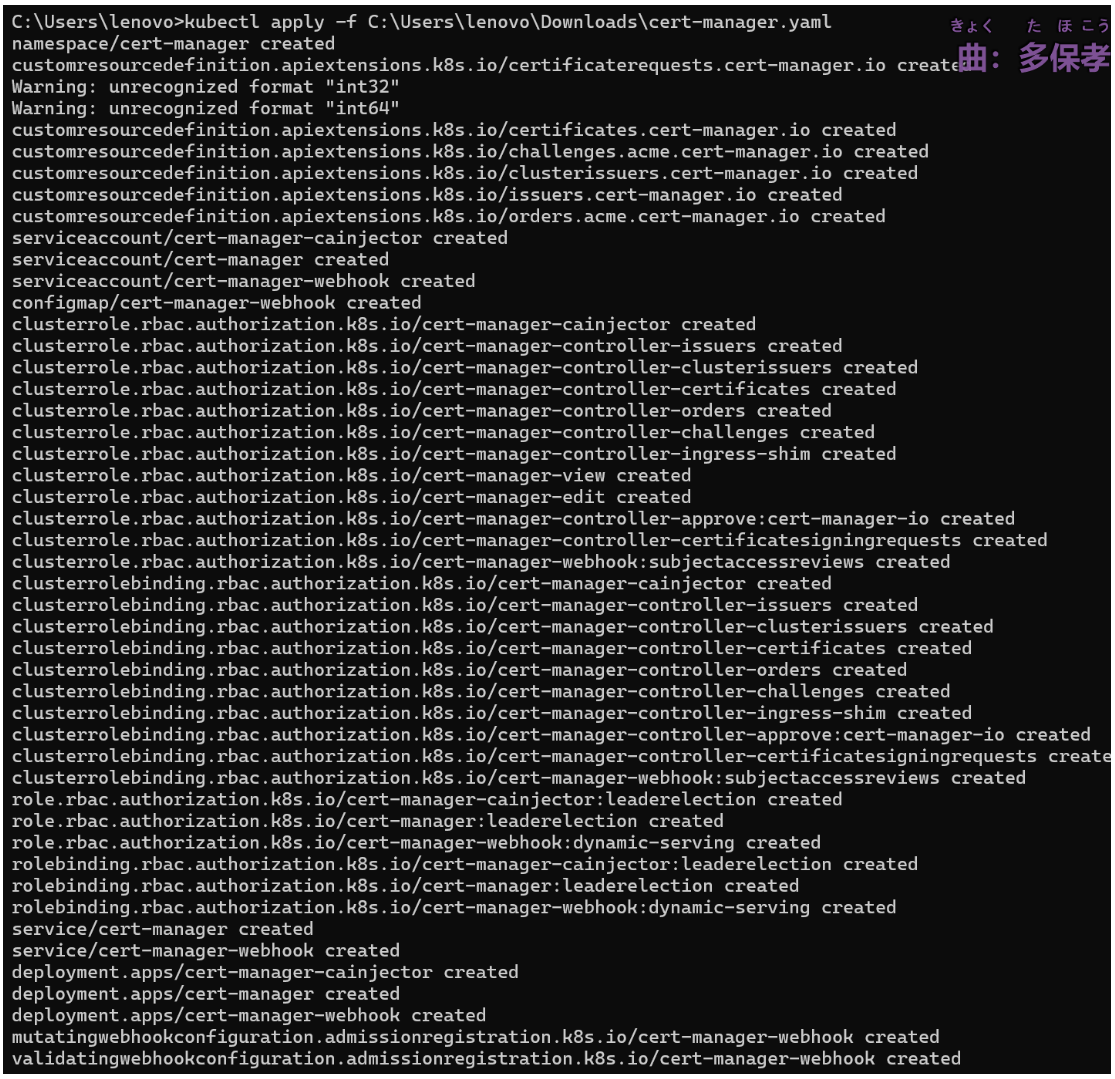
This experiment provided hands-on experience in deploying a predictive model using KServe in a local Kubernetes environment. Key takeaways include the importance of proper certificate management for webhook services, monitoring InferenceService status for debugging, and exposing services externally to validate functionality. The workflow demonstrated in this experiment serves as a practical guide for deploying and testing machine learning models in a production-like Kubernetes setup.



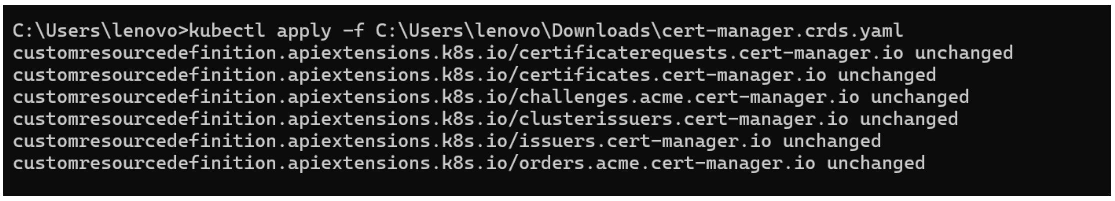
```
PS C:\Users\lenovo> minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Users\lenovo> kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   93d   v1.34.0
PS C:\Users\lenovo>
```



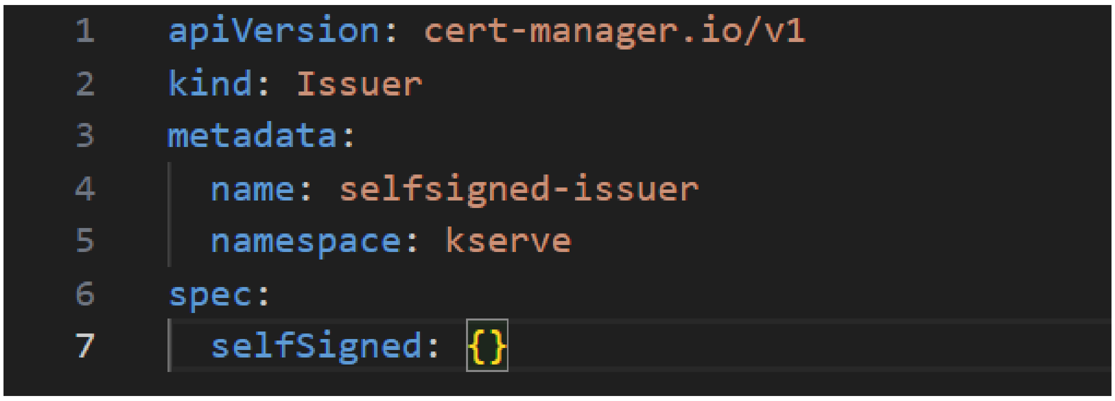
```
C:\Users\lenovo>kubectl apply -f https://github.com/kserve/kserve/releases/download/v0.13.0/kserve.yaml
^C
C:\Users\lenovo>
C:\Users\lenovo>kubectl apply -f C:\Users\lenovo\Downloads\kserve.yaml
namespace/kserve created
Warning: unrecognized format "int32"
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/clusterservingruntimes.serving.kserve.io created
customresourcedefinition.apiextensions.k8s.io/clusterstoragecontainers.serving.kserve.io created
customresourcedefinition.apiextensions.k8s.io/inferencegraphs.serving.kserve.io created
customresourcedefinition.apiextensions.k8s.io/inferenceservices.serving.kserve.io created
customresourcedefinition.apiextensions.k8s.io/servingruntimes.serving.kserve.io created
customresourcedefinition.apiextensions.k8s.io/trainedmodels.serving.kserve.io created
serviceaccount/kserve-controller-manager created
role.rbac.authorization.k8s.io/kserve-leader-election-role created
clusterrole.rbac.authorization.k8s.io/kserve-manager-role created
clusterrole.rbac.authorization.k8s.io/kserve-proxy-role created
rolebinding.rbac.authorization.k8s.io/kserve-leader-election-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/kserve-manager-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/kserve-proxy-rolebinding created
configmap/inferenceservice-config created
secret/kserve-webhook-server-secret created
service/kserve-controller-manager-metrics-service created
service/kserve-controller-manager-service created
service/kserve-webhook-server-service created
deployment.apps/kserve-controller-manager created
mutatingwebhookconfiguration.admissionregistration.k8s.io/inferenceservice.serving.kserve.io created
validatingwebhookconfiguration.admissionregistration.k8s.io/clusterservingruntime.serving.kserve.io created
```



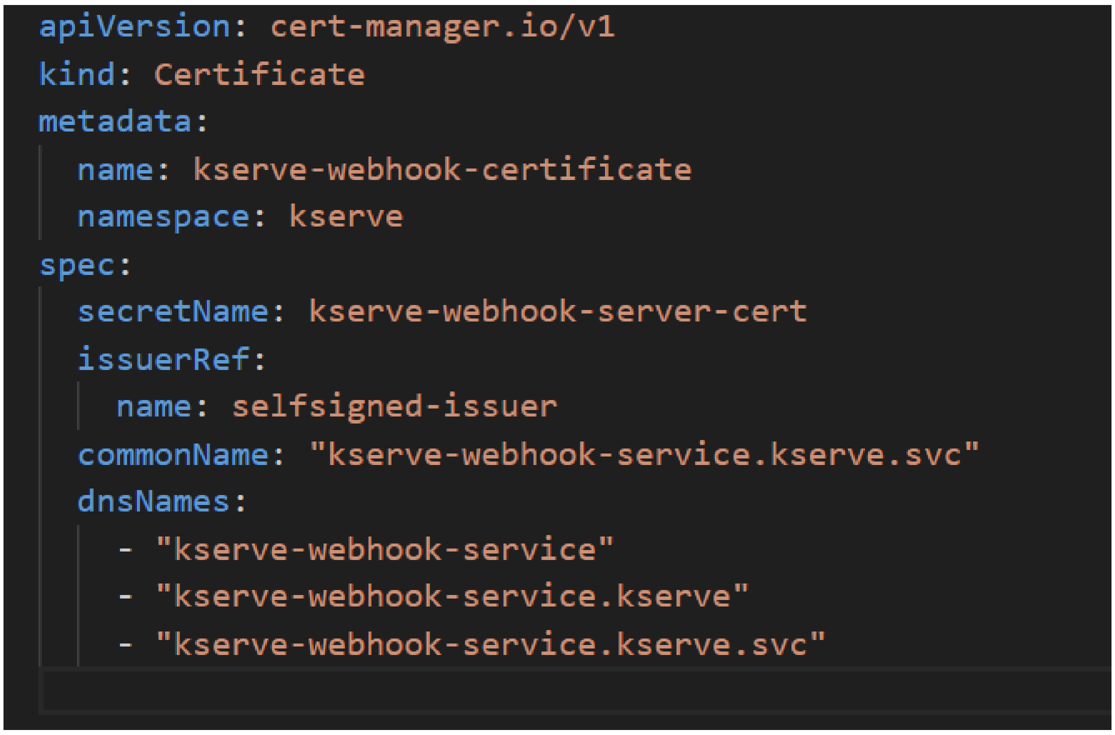
```
C:\Users\lenovo>kubectl apply -f C:\Users\lenovo\Downloads\cert-manager.yaml
namespace/cert-manager created
customresourcedefinition.apiextensions.k8s.io/certificaterequests.cert-manager.io create
Warning: unrecognized format "int32"
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/certificates.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/challenges.acme.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/clusterissuers.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/issuers.cert-manager.io created
customresourcedefinition.apiextensions.k8s.io/orders.acme.cert-manager.io created
serviceaccount/cert-manager-cainjector created
serviceaccount/cert-manager created
serviceaccount/cert-manager-webhook created
configmap/cert-manager-webhook created
clusterrole.rbac.authorization.k8s.io/cert-manager-cainjector created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-issuers created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-clusterissuers created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-certificates created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-orders created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-challenges created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-ingress-shim created
clusterrole.rbac.authorization.k8s.io/cert-manager-view created
clusterrole.rbac.authorization.k8s.io/cert-manager-edit created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-approve:cert-manager-io created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-certificatesigningrequests created
clusterrole.rbac.authorization.k8s.io/cert-manager-webhook:subjectaccessreviews created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-cainjector created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-issuers created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-clusterissuers created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-certificates created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-orders created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-challenges created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-ingress-shim created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-approve:cert-manager-io created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-certificatesigningrequests create
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-webhook:subjectaccessreviews created
role.rbac.authorization.k8s.io/cert-manager-cainjector:leaderelection created
role.rbac.authorization.k8s.io/cert-manager:leaderelection created
role.rbac.authorization.k8s.io/cert-manager-webhook:dynamic-serving created
rolebinding.rbac.authorization.k8s.io/cert-manager-cainjector:leaderelection created
rolebinding.rbac.authorization.k8s.io/cert-manager:leaderelection created
rolebinding.rbac.authorization.k8s.io/cert-manager-webhook:dynamic-serving created
service/cert-manager created
service/cert-manager-webhook created
deployment.apps/cert-manager-cainjector created
deployment.apps/cert-manager created
deployment.apps/cert-manager-webhook created
mutatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created
validatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created
```



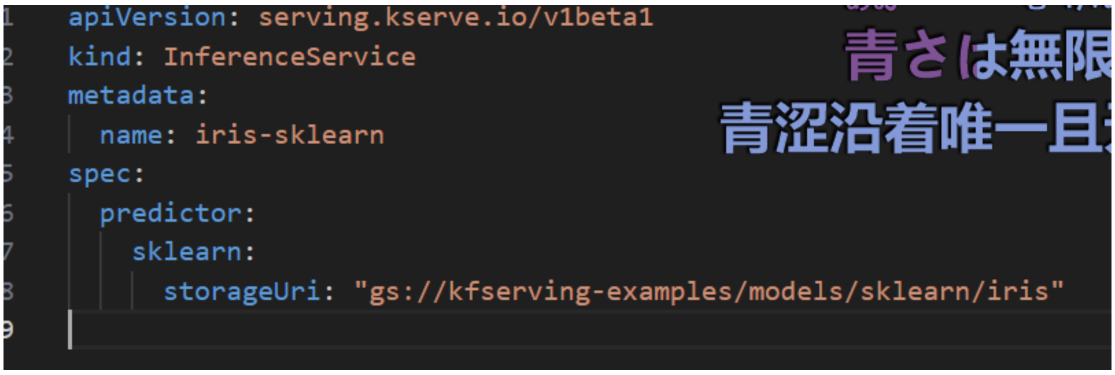
```
C:\Users\lenovo>kubectl apply -f C:\Users\lenovo\Downloads\cert-manager.crds.yaml
customresourcedefinition.apiextensions.k8s.io/certificaterequests.cert-manager.io unchanged
customresourcedefinition.apiextensions.k8s.io/certificates.cert-manager.io unchanged
customresourcedefinition.apiextensions.k8s.io/challenges.acme.cert-manager.io unchanged
customresourcedefinition.apiextensions.k8s.io/clusterissuers.cert-manager.io unchanged
customresourcedefinition.apiextensions.k8s.io/issuers.cert-manager.io unchanged
customresourcedefinition.apiextensions.k8s.io/orders.acme.cert-manager.io unchanged
```



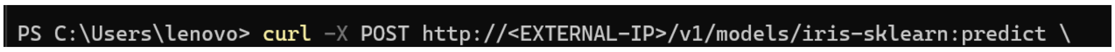
```yaml
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: selfsigned-issuer
  namespace: kserve
spec:
  selfSigned: {}
```



```yaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: kserve-webhook-certificate
  namespace: kserve
spec:
  secretName: kserve-webhook-server-cert
  issuerRef:
    name: selfsigned-issuer
  commonName: "kserve-webhook-service.kserve.svc"
  dnsNames:
    - "kserve-webhook-service"
    - "kserve-webhook-service.kserve"
    - "kserve-webhook-service.kserve.svc"
```



```yaml
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: iris-sklearn
spec:
  predictor:
    sklearn:
      storageUri: "gs://kfserving-examples/models/sklearn/iris"
```



```
PS C:\Users\lenovo> curl -X POST http://<EXTERNAL-IP>/v1/models/iris-sklearn:predict \
```