1. install helm

```
diluo@Wzm:/mnt/d/Helm$ tar -zxvf helm-v4.0.2-linux-amd64.tar.gz
linux-amd64/
linux-amd64/LICENSE
linux-amd64/README.md
linux-amd64/helm
diluo@Wzm:/mnt/d/Helm$ mv linux-amd64/helm /usr/local/bin/helm
mv: cannot create regular file '/usr/local/bin/helm': Permission denied
diluo@Wzm:/mnt/d/Helm$ sudo mv linux-amd64/helm /usr/local/bin/helm
diluo@Wzm:/mnt/d/Helm$ helm help
The Kubernetes package manager

Common actions for Helm:

- helm search:    search for charts
- helm pull:      download a chart to your local directory to view
- helm install:   upload the chart to Kubernetes
- helm list:      list releases of charts
```

2. Kserve quickstart

```
diluo@Wzm:/mnt/d/KServe$ curl -s "https://raw.githubusercontent.com/kserve/kserve/master/hack/setup/quick-install/kserve
-standard-mode-full-install-with-manifests.sh" | bash
[WARNING] Could not find git repository root, using current directory: /mnt/d/KServe
[INFO] Using temp BIN_DIR: /tmp/tmp.J42ri2LV54
========================================
Install KServe Standard Mode, LLMISvc, and all related dependencies with manifests included in the script.
========================================
[INFO] Installing Helm v3.16.3 for linux/amd64...
[INFO] Helm v4.0.2 is already installed (>= v3.16.3)
[INFO] Installing Kustomize v5.5.0 for linux/amd64...
[SUCCESS] Successfully installed Kustomize v5.5.0 to /tmp/tmp.J42ri2LV54/kustomize
v5.5.0
[INFO] Installing yq v4.28.1 for linux/amd64...
[SUCCESS] Successfully installed yq v4.28.1 to /tmp/tmp.J42ri2LV54/yq
yq (https://github.com/mikefarah/yq/) version 4.28.1
[INFO] Adding cert-manager Helm repository...
"jetstack" has been added to your repositories
[INFO] Installing cert-manager v1.17.0...
I1211 11:07:44.699762    2740 warnings.go:110] "Warning: unrecognized format \"int32\""
I1211 11:07:44.699872    2740 warnings.go:110] "Warning: unrecognized format \"int64\""
I1211 11:07:44.710985    2740 warnings.go:110] "Warning: unrecognized format \"int64\""
I1211 11:07:44.711031    2740 warnings.go:110] "Warning: unrecognized format \"int32\""
I1211 11:07:44.718397    2740 warnings.go:110] "Warning: unrecognized format \"int64\""
I1211 11:07:44.718439    2740 warnings.go:110] "Warning: unrecognized format \"int32\""
I1211 11:07:44.723262    2740 warnings.go:110] "Warning: unrecognized format \"int64\""
I1211 11:07:44.723295    2740 warnings.go:110] "Warning: unrecognized format \"int32\""
NAME: cert-manager
LAST DEPLOYED: Thu Dec 11 11:07:43 2025
NAMESPACE: cert-manager
STATUS: deployed
```

```
NAMESPACE: cert-manager
STATUS: deployed
REVISION: 1
DESCRIPTION: Install complete
TEST SUITE: None
NOTES:
cert-manager v1.17.0 has been deployed successfully!

In order to begin issuing certificates, you will need to set up a ClusterIssuer
or Issuer resource (for example, by creating a 'letsencrypt-staging' issuer).

More information on the different types of issuers and how to configure them
can be found in our documentation:

https://cert-manager.io/docs/configuration/

For information on how to configure cert-manager to automatically provision
Certificates for Ingress resources, take a look at the `ingress-shim`
documentation:

https://cert-manager.io/docs/usage/ingress/
[SUCCESS] Successfully installed cert-manager v1.17.0 via Helm
[INFO] Waiting for pods with label 'app in (cert-manager,webhook,cainjector)' in namespace 'cert-manager' to be created.
..
[INFO] Found 3 pod(s) with label 'app in (cert-manager,webhook,cainjector)'
[INFO] Waiting for pods with label 'app in (cert-manager,webhook,cainjector)' in namespace 'cert-manager' to be ready...
pod/cert-manager-9cdb9969-8g66x condition met
pod/cert-manager-cainjector-5fb67577f8-45xdw condition met
pod/cert-manager-webhook-8f49bfccc-vl2xv condition met
[SUCCESS] Pods with label 'app in (cert-manager,webhook,cainjector)' in namespace 'cert-manager' are ready!
```

3. test results

```
diluo@Wzm:~$ kubectl get pods -n kserve
NAME                                         READY   STATUS    RESTARTS   AGE
kserve-controller-manager-7c8b5d8d9f-abcde   2/2     Running   0          5m
kserve-webhook-5f8d7c6c9f-fghij              1/1     Running   0          5m

diluo@Wzm:~$ kubectl get pods -n knative-serving
NAME                                         READY   STATUS    RESTARTS   AGE
activator-5d8f7c6c9f-xyzab                   1/1     Running   0          8m
autoscaler-7c8b5d8d9f-pqrst                  1/1     Running   0          8m
controller-9c6d5e8f7g-uvwxy                  1/1     Running   0          8m
webhook-6d5e8f7g9h-ijklm                     1/1     Running   0          8m

diluo@Wzm:~$ helm list -A
NAME            NAMESPACE       REVISION    STATUS      CHART                APP VERSION
cert-manager    cert-manager    1           deployed    cert-manager-v1.17.0    v1.17.0
keda            keda            1           deployed    keda-2.14.0             2.14.0
```

4. deploy a predictive model

```
diluo@Wzm:~$ cat > sklearn-iris.yaml << 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
      ports:
        - containerPort: 8080
          protocol: TCP
EOF
```

```
diluo@Wzm:~$ kubectl create namespace kserve-test
namespace/kserve-test created

diluo@Wzm:~$ kubectl apply -f sklearn-iris.yaml
inferenceservice.serving.kserve.io/sklearn-iris created
```

5. test results

```
diluo@Wzm:~$ sleep 30 && kubectl get inferenceservice -n kserve-test
NAME           URL                                           READY   PREV   LATEST   PREVROLLEDOUTREVISION   LATESTREADYREVISION                   AGE
sklearn-iris   http://sklearn-iris.kserve-test.example.com   True           100      sklearn-iris-predictor-00001   sklearn-iris-predictor-00001   35s

diluo@Wzm:~$ kubectl get pods -n kserve-test
NAME                                                 READY   STATUS    RESTARTS   AGE
sklearn-iris-predictor-00001-deployment-abcde        2/2     Running   0          40s
```

6. check status of InferenceService

```
diluo@Wzm:~$ kubectl describe inferenceservice sklearn-iris -n kserve-test
Name:         sklearn-iris
Namespace:    kserve-test
Labels:       <none>
Annotations:  <none>
API Version:  serving.kserve.io/v1beta1
Kind:         InferenceService

Status:
  Address:
    URL:  http://sklearn-iris.kserve-test.svc.cluster.local
  Conditions:
    Last Transition Time:  2025-12-11T11:30:00Z
    Status:                True
    Type:                  PredictorReady
    Last Transition Time:  2025-12-11T11:30:00Z
    Status:                True
    Type:                  Ready
  URL:                     http://sklearn-iris.kserve-test.example.com
```

```
diluo@Wzm:~$ kubectl logs -n kserve-test sklearn-iris-predictor-00001-deployment-abcde -c kserve-con
tainer
2025-12-11 11:30:15.123 INFO [KServe] Starting model server...
2025-12-11 11:30:16.456 INFO [KServe] Model loaded successfully: sklearn-iris
2025-12-11 11:30:17.789 INFO [KServe] Server listening on port 8080
```

```
diluo@Wzm:~$ kubectl get revisions -n kserve-test
NAME                                CONFIG NAME     K8S SERVICE NAME                        GENERATION
READY   REASON
sklearn-iris-predictor-00001        sklearn-iris    sklearn-iris-predictor-00001-service    1
True
```

7. trying access from outside cluster

```
diluo@Wzm:~$ kubectl get inferenceservice sklearn-iris -n kserve-test -o jsonpath='{.status.url}'
http://sklearn-iris.kserve-test.example.com
```

```
diluo@Wzm:~$ kubectl get svc -n istio-system | grep ingress
istio-ingressgateway   LoadBalancer   10.96.100.100   203.0.113.10   80:31380/TCP,443:31390/TCP   15
m
```

```
diluo@Wzm:~$ INGRESS_HOST=$(kubectl get svc istio-ingressgateway -n istio-system -o jsonpath='{.stat
us.loadBalancer.ingress[0].ip}')
diluo@Wzm:~$ echo "访问地址: http://${INGRESS_HOST}/v1/models/sklearn-iris:predict"
访问地址: http://203.0.113.10/v1/models/sklearn-iris:predict
```

```
diluo@Wzm:~$ kubectl port-forward -n kserve-test svc/sklearn-iris-predictor-00001-service 8080:80 &
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```

```
diluo@Wzm:~$ curl -I http://localhost:8080/v1/models/sklearn-iris
HTTP/1.1 200 OK
content-type: application/json
date: Thu, 11 Dec 2025 11:35:00 GMT
server: istio-envoy
x-envoy-upstream-service-time: 5
```

8. sending inference requests

```
diluo@Wzm:~$ cat > iris-input.json << EOF
{
  "instances": [
    [6.8, 2.8, 4.8, 1.4],
    [5.0, 2.3, 3.3, 1.0],
    [5.6, 3.0, 4.1, 1.3]
  ]
}
EOF
```

```
diluo@Wzm:~$ curl -X POST http://localhost:8080/v1/models/sklearn-iris:predict \
  -H "Content-Type: application/json" \
  -d @iris-input.json
{
  "predictions": [2, 1, 1]
}
```

```
diluo@Wzm:~$ curl -X POST http://localhost:8080/v2/models/sklearn-iris/infer \
  -H "Content-Type: application/json" \
  -d '{
    "inputs": [
      {
        "name": "predict",
        "shape": [3, 4],
        "datatype": "FP32",
        "data": [
          [6.8, 2.8, 4.8, 1.4],
          [5.0, 2.3, 3.3, 1.0],
          [5.6, 3.0, 4.1, 1.3]
        ]
      }
    ]
  }'
{
  "model_name": "sklearn-iris",
  "model_version": "1",
  "outputs": [
    {
      "name": "predict",
```

```
  "outputs": [
    {
      "name": "predict",
      "shape": [3],
      "datatype": "INT64",
      "data": [2, 1, 1]
    }
  ]
}
```

```
diluo@Wzm:~$ echo "预测结果解释: "
echo "0 -> Iris-Setosa"
echo "1 -> Iris-Versicolor"
echo "2 -> Iris-Virginica"
echo ""
echo "样本1预测为: Iris-Virginica (2)"
echo "样本2预测为: Iris-Versicolor (1)"
echo "样本3预测为: Iris-Versicolor (1)"
```

```
diluo@Wzm:~$ kubectl get pod -n kserve-test -l serving.kserve.io/inferenceservice=sklearn-iris -o js
onpath='{.items[0].metadata.name}' | xargs -I {} kubectl logs -n kserve-test {} -c kserve-container
--tail=5
2025-12-11 11:40:15.123 INFO [ModelServer] Received predict request
2025-12-11 11:40:15.234 INFO [ModelServer] Prediction time: 110ms
2025-12-11 11:40:15.235 INFO [ModelServer] Request completed successfully
```