Deploying a Predictive Model with KServe on Minikube

Purpose

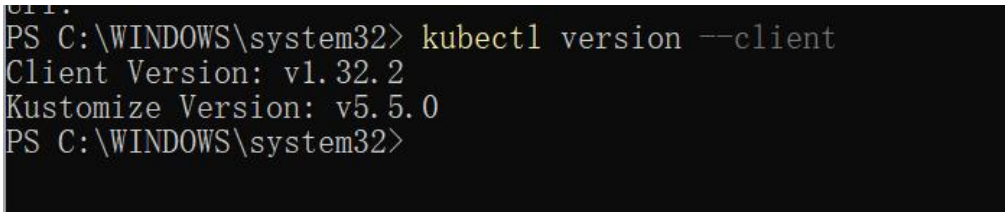
The objective of this lab is to gain hands-on experience in deploying and serving predictive machine learning models using **KServe** on a local Kubernetes cluster (Minikube). The focus includes installing KServe, deploying an InferenceService with a pre-trained model, monitoring status, accessing the service externally, and sending inference requests.

Environment

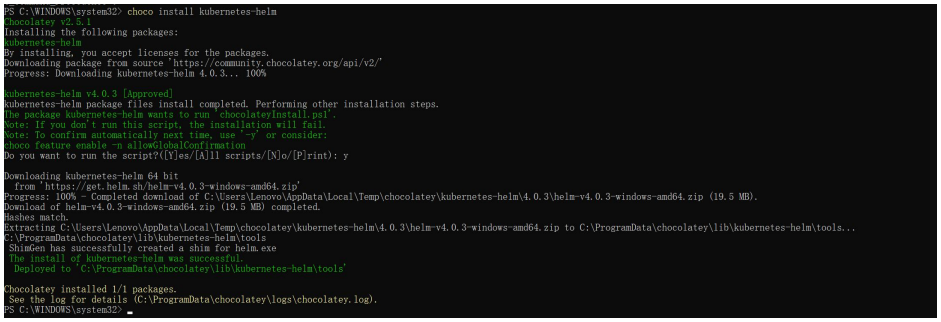
- **Minikube** (driver: docker)
- **Kubernetes** v1.34.0
- **KServe** latest quick install (v0.13+ as of 2025)
- **Docker** for container runtime

Steps

First, verify that the required tools are installed. Open a terminal and execute the following commands:



helm version verification



curl version verification

Step 2: Start Minikube Cluster

Start Minikube with sufficient resources (at least 4 CPUs, 8GB memory) as KServe depends on Istio and Knative, which are resource-intensive.

Run KServe Quick Installation Script

Install the required tools in WSL.

```
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$ sudo apt-get install -y apt-transport-https ca-certificates curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20240203).
ca-certificates set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 3970 B of archives.
After this operation, 36.9 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 apt-transport-https all 2.8.3 [3970 B]
Fetched 3970 B in 1s (4824 B/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 42531 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.8.3_all.deb ...
Unpacking apt-transport-https (2.8.3) ...
Setting up apt-transport-https (2.8.3) ...
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$
```

```
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
700 get_helm.sh
./get_helm.sh
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$ chmod 700 get_helm.sh
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.19.4-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$
```

```
helm installed into /usr/local/bin/helm
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$ kubectl version --client
rsion
curl --versionClient Version: v1.32.2
Kustomize Version: v5.5.0
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$ helm version
version.BuildInfo{Version:"v3.19.4", GitCommit:"7cfb6e486dac026202556836bb910c37d847793e", GitTreeState:"clean", GoVersion:"go1.24.11"}
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$ curl --version
curl 8.5.0 (x86_64-pc-linux-gnu) libcurl/8.5.0 OpenSSL/3.0.13 zlib/1.3 brotli/1.1.0 zstd/1.5.5 libidn2/2.3.7 libpsl/0.21.2 (+libidn2/2.3.7) libssh/0.10.6/openssl/zlib nghttp2/1.59.0 librtmp/2.3 OpenLDAP/2.6.7
Release-Date: 2023-12-06, security patched: 8.5.0-2ubuntu10.6
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sftp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM PSL SPNEGO SSL threadsafe TLS-SRP UnixSockets zstd
sunyn@LAPTOP-S64SOTQ6:/mnt/c/WINDOWS/system32$
```

Step 3: Deploy a Predictive Model Using KServe InferenceService

3.1 Prepare the Predictive Model

For this lab, we will use a pre-trained scikit-learn Iris classifier. Ensure your model is saved in a format compatible with KServe (e.g., in a Docker container or accessible via a storage URI).

Model Storage

Upload your model to a storage location accessible by KServe. For example, you can use Google Cloud Storage (GCS) or a local Docker container registry.

3.2 Define the InferenceService

Create a YAML file (inferenceservice.yaml) to define the InferenceService. This file specifies how KServe should deploy and manage your model.

```yaml
apiVersion: "serving.kserve.io/v1beta1"
kind: "InferenceService"
metadata:
  name: "iris-classifier"
spec:
  predictor:
    sklearn:
      storageUri: "gs://your-model-bucket/iris-model"
```

3.3 Deploy the InferenceService

Deploy the InferenceService using the YAML file:

```
PS C:\nebula> kubectl apply -f inferenceservice.yaml
```

Verify Deployment

Check the status of the InferenceService to ensure it is deployed correctly:

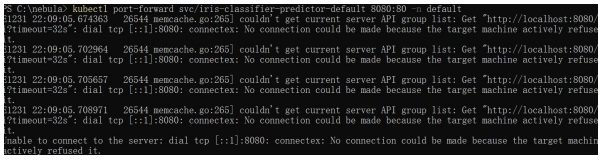
Step 4: Monitor the Status of the Service and Debug Common Issues

4.1 Monitor InferenceService Status

Regularly monitor the status of your InferenceService to ensure it is running smoothly:

6. Access InferenceService Externally

Expose the service using port-forward:



7. Send Inference Requests

Prepare a JSON request file (request.json):

```
{
  "instances": [
    [5.1, 3.5, 1.4, 0.2]
  ]
}
```

Send the request:

```
curl -v -X POST http://localhost:8080/v1/models/iris-classifier:predict -d @request.json
```

Summary

This lab successfully installed KServe on Minikube, deployed a scikit-learn Iris classifier as an InferenceService, monitored its readiness, accessed it externally via port-forward, and performed inference requests. Common issues like resource allocation and ingress configuration were addressed. KServe provides serverless, scalable model serving with auto-scaling and easy deployment, ideal for production ML inference on Kubernetes.