

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:

```
PS C:\WINDOWS\system32> kubectl version --client
Client Version: v1.32.2
Kustomize Version: v5.5.0
PS C:\WINDOWS\system32>
```

helm version verification

```

PS C:\WINDOWS\system32> choco install kubernetes-helm
Chocolatey v2.5.1
Installing the following packages:
  helm
By installing, you accept licenses for the packages.
Download package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading kubernetes-helm 4.0.3... 100%
kubernetes-helm v4.0.3 [Approved]
kubernetes-helm package files install completed. Performing other installation steps.
The package 'kubernetes-helm' wants to run 'chocoatey\helm'.
This will run 'chocoatey\helm'. Installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?(Y|y/[A]ll scripts/[N]o/[P]rint): y
Downloading kubernetes-helm 64 bit
From 'https://get.helm.sh/helm-v4.0.3-windows-amd64.zip'
Progress: Downloading helm-v4.0.3-windows-amd64.zip of C:\Users\lenovo\AppData\Local\Temp\chocolatey\kubernetes-helm\4.0.3\helm-v4.0.3-windows-amd64.zip (19.5 MB).
Download of helm-v4.0.3-windows-amd64.zip (19.5 MB) completed.
Hashes match.
Extracting C:\Users\lenovo\AppData\Local\Temp\chocolatey\kubernetes-helm\4.0.3\helm-v4.0.3-windows-amd64.zip to C:\ProgramData\chocolatey\lib\kubernetes-helm\tools...
chocoatey\helm\sh\helm-v4.0.3-windows-amd64.exe has been successfully created.
ShimEnv has successfully created a shim for helm.exe
The install of kubernetes-helm was successful.
Deployed to 'C:\ProgramData\chocolatey\lib\kubernetes-helm\tools'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\WINDOWS\system32>

```

curl version verification

```

PS C:\WINDOWS\system32> curl.exe --version
curl 8.13.0 (Windows) libcurl/8.13.0 Schannel zlib/1.3.1 WinIDN
Release-Date: 2025-04-02
Protocols: dict file ftp ftps http https imap imaps ipfs ipns mqtt pop3 pop3s smb smbs smtp smtps telnet tftp ws wss
Features: alt-svc AsynchDNS HSTS HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM SPNEGO SSL SSPI threadsafe Unicode UnixSockets
PS C:\WINDOWS\system32>

```

Step 2: Start Minikube Cluster

```

PS C:\WINDOWS\system32> minikube start
* Microsoft Windows 11 Home China 10.0.22631.6199 Build 22631.6199 上的 minikube v1.37.0
* 根据现有的配置文件使用 docker 驱动程序
* 在集群中 "minikube" 启动节点 "minikube" primary control-plane
* 正在拉取基础镜像 v0.0.48 ...
* 正在下载 Kubernetes v1.34.0 的预加载文件...
* 正在更新运行中的 docker "minikube" container ...
! 从 Minikube 的 container 内部连接到 https://registry.k8s.io/ 失败
* 要获取新的外部镜像, 可能需要配置代理: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
! The image 'gcr.io/k8s-minikube/storage-provisioner:v5' was not found; unable to add it to cache.
! The image 'registry.k8s.io/kube-controller-manager:v1.34.0' was not found; unable to add it to cache.
! The image 'registry.k8s.io/kube-apiserver:v1.34.0' was not found; unable to add it to cache.
! The image 'registry.k8s.io/kube-scheduler:v1.34.0' was not found; unable to add it to cache.
! The image 'registry.k8s.io/kube-proxy:v1.34.0' was not found; unable to add it to cache.
! The image 'registry.k8s.io/etcd:3.6.4-0' was not found; unable to add it to cache.
! The image 'registry.k8s.io/pause:3.10.1' was not found; unable to add it to cache.
! The image 'registry.k8s.io/coredns/coredns:v1.12.1' was not found; unable to add it to cache.
* 正在 Docker 28.4.0 中准备 Kubernetes v1.34.0...
* 正在验证 Kubernetes 组件...
- 正在使用镜像 gcr.io/k8s-minikube/storage-provisioner:v5
* 启用插件: storage-provisioner, default-storageclass
! C:\Program Files\Docker\resources\bin\kubectl.exe 的版本为 1.32.2, 可能与 Kubernetes 1.34.0 不兼容。
- 想要使用 kubectl v1.34.0 吗? 尝试使用 'minikube kubectl -- get pods -A' 命令
* 完成! minikube 现在已配置, 默认使用"minikube"集群和"default"命名空间
PS C:\WINDOWS\system32>

```

件, 这些组件较耗资源。

Start Minikube with sufficient resources (at least 4 CPUs).

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

```

PS C:\WINDOWS\system32> kubectl get nodes
NAME      STATUS    ROLES          AGE     VERSION
minikube  Ready     control-plane  16h    v1.34.0
PS C:\WINDOWS\system32>

```

Run KServe Quick Installation Script

Install the required tools in WSL.

```
sunyn@LAPTOP-S64S0TQ6:/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:  
. libl1m19  
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-S64S0TQ6:/mnt/c/WINDOWS/system32$
```

```
sunyn@LAPTOP-S64S0TQ6:/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-S64S0TQ6:/mnt/c/WINDOWS/system32$ chmod 700 get_helm.sh  
sunyn@LAPTOP-S64S0TQ6:/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-S64S0TQ6:/mnt/c/WINDOWS/system32$
```

```
helm installed into /usr/local/bin/helm  
sunyn@LAPTOP-S64S0TQ6:/mnt/c/WINDOWS/system32$ kubectl version --client  
Client Version: v1.32.2  
Kustomize Version: v5.0.0  
sunyn@LAPTOP-S64S0TQ6:/mnt/c/WINDOWS/system32$ helm version  
version.ClientVersion: v3.19.4  
version.GitCommit: 026202556836b910c37d4847793e  
version.GitTreeState: "clean"  
version.GitURL: https://github.com/helm/helm.git  
version.License: Apache-2.0  
version.Major: 3  
version.Minor: 19  
version.Patch: 4  
version.PreRelease: ""  
version.ReleaseDate: 2023-12-06  
version.SecurityPatched: 8.5.0-2ubuntu10.6  
Protocol: dict file ftp ftps http https imap maps ldap ldaps mailer pop3 pop3s rtsp rtspss sftp smb smbs smtp smtppst telnet tftp  
Features: alias auth basic digest gzip http https https proxy idn IPv6 Kerberos Largefile libz NTLM PSL SPNEGO SSL threadsafe TLS-SRP UnixSockets zstd  
sunyn@LAPTOP-S64S0TQ6:/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.

```

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:

```
PS C:\nebula> kubectl get pods -n default
[...]
11231 22:07:03.704417 31092 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
11231 22:07:03.731187 31092 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
11231 22:07:03.733855 31092 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
11231 22:07:03.736015 31092 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
11231 22:07:03.741738 31092 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
unable to connect to the server: dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
PS C:\nebula>
```

6. Access InferenceService Externally

Expose the service using port-forward:

```
PS C:\nebula> kubectl port-forward svc/iris-classifier-predictor-default 8080:80 -n default
[...]
11231 22:09:05.674563 26544 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
11231 22:09:05.702964 26544 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
11231 22:09:05.705657 26544 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
11231 22:09:05.708971 26544 nemcache.go:265] couldn't get current server API group list: Get "http://localhost:8080/api/v1/namespaces/default": dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
unable to connect to the server: dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.
```

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.