

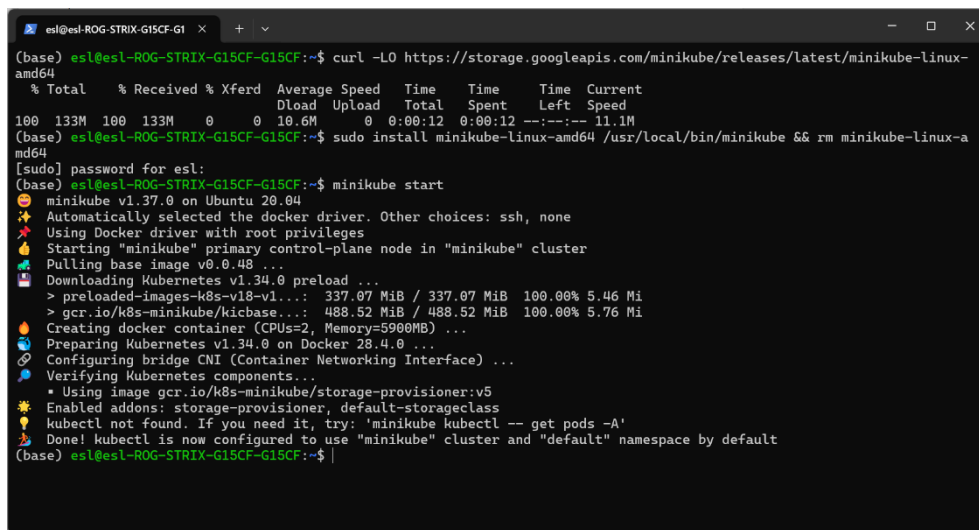
開源軟體開發 MLOps 作業

姓名：張健勳

學號：614410073

1. 基礎環境建置與檢測

首先啟動本地 Kubernetes 叢集。使用 `minikube start` 成功初始化控制平面 (Control Plane)，並下載基礎映像檔。隨後透過 `kubectl get pods -A` 檢查系統元件，確認 `etcd`、`kube-apiserver`、`coredns` 等核心服務皆處於 `Running` 狀態，叢集運作正常。



```
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
100 133M 100 133M 0 0 10.6M 0 0:00:12 0:00:12 --:--:-- 11.1M
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
[sudo] password for esl:
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$ minikube start
minikube v1.37.0 on Ubuntu 20.04
🔧 Automatically selected the docker driver. Other choices: ssh, none
👉 Using Docker driver with root privileges
🌟 Starting "minikube" primary control-plane node in "minikube" cluster
📡 Pulling base image v0.0.48 ...
📦 Downloading Kubernetes v1.34.0 preload ...
> preloaded-images-k8s-v18-v1...: 337.07 MiB / 337.07 MiB 100.00% 5.46 Mi
> gcr.io/k8s-minikube/kicbase...: 488.52 MiB / 488.52 MiB 100.00% 5.76 Mi
🔥 Creating docker container (CPUs=2, Memory=5900MB) ...
📦 Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
🔗 Configuring bridge CNI (Container Networking Interface) ...
🔍 Verifying Kubernetes components...
▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
💡 kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🎉 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$
```

2. KServe 服務元件檢查

在部署模型前，驗證 KServe 及其依賴元件的狀態。

- KServe Controller: 確認 `kserve-controller-manager` 處於 `Running` 且狀態為 `2/2`，表示控制器已就緒。
- Cert-Manager: 檢查 `cert-manager`、`cainjector` 與 `webhook` 等 Pod 皆正常運作，確保 `Webhook` 憑證管理功能無誤。

```
esl@esl-ROG-STRIX-G15CF-G1 x + - x
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$ kubectl get pods -A
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE
kube-system   coredns-66bc5c9577-ltplh             1/1     Running   0           2m55s
kube-system   etcd-minikube                         1/1     Running   0           3m
kube-system   kube-apiserver-minikube               1/1     Running   0           3m2s
kube-system   kube-controller-manager-minikube      1/1     Running   0           3m
kube-system   kube-proxy-y4zvt                     1/1     Running   0           2m55s
kube-system   kube-scheduler-minikube               1/1     Running   0           3m
kube-system   storage-provisioner                  1/1     Running   1 (2m24s ago) 2m59s
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$

esl@esl-ROG-STRIX-G15CF-G1 x + - x
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$ kubectl get pods -n kserve
NAME                                     READY   STATUS    RESTARTS   AGE
kservice-controller-manager-bc4466cf6-c5nhf 2/2     Running   0           2m52s
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$ kubectl get pods -n cert-manager
NAME                                     READY   STATUS    RESTARTS   AGE
cert-manager-6996b77986-jrvpt             1/1     Running   0           4m6s
cert-manager-cainjector-84c4464cc7-rq7nt   1/1     Running   0           4m6s
cert-manager-webhook-7c79c64bb4-zmjwp     1/1     Running   0           4m6s
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~$
```

3. 模型部署 (InferenceService)

使用定義好的 YAML 設定檔部署模型：

- 執行 `kubectl apply -f tensorflow.yaml` 建立資源，系統回傳 `inferenceservice.serving.kservice.io/flower-sample created`。
- 透過 `kubectl get isvc flower-sample` 監控部署進度。初期狀態可能顯示 `Unknown`，待模型容器下載並初始化完成後，`READY` 欄位轉為 `True`，且 `URL` 已生成 (`http://flower-sample.default.example.com`)。
- 同步確認 `Pod` 狀態，`flower-sample-predictor` 顯示 `2/2 Running`，代表模型伺服器已上線。

```
esl@esl-ROG-STRIX-G15CF-G1 X + v
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~/kserve_flowerv$ kubectl apply -f tensorflow.yaml
inferenceservice.serving.kserve.io/flower-sample created
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~/kserve_flowerv$
```

4. 網路配置與端口轉發 (Port-Forwarding)

為了從本地端存取叢集內的模型服務，進行以下網路設定：

- 獲取 Istio Ingress Gateway 的服務名稱與端口資訊。
- 設定環境變數 INGRESS_HOST 與 INGRESS_PORT。
- 執行 `kubectl port-forward` 將本地的 8080 端口映射至 Istio Ingress Gateway 的 80 端口，建立連線通道。

```
esl@esl-ROG-STRIX-G15CF-G1 X + v
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~/kserve_flowerv$ kubectl get isvc flower-sample
NAME URL AGE READY PREV LATEST PREVROLLOUTREVISION LATESTREADYRE
flower-sample http://flower-sample.default.example.com True 100 flower-sample
-predictor-00001 97s
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~/kserve_flowerv$ kubectl get pods
NAME READY STATUS RESTARTS AGE
flower-sample-predictor-00001-deployment-75bd6f86cd-8wcr1 2/2 Running 0 96s
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~/kserve_flowerv$ kubectl get svc istio-ingressgateway -n istio-system
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
istio-ingressgateway LoadBalancer 10.98.128.238 <pending> 15021:30938/TCP,80:31818/TCP,443:31134/TCP 8m16s
(base) esl@esl-ROG-STRIX-G15CF-G15CF:~/kserve_flowerv$
```

5. 推論測試 (Inference Testing)

最後進行實際的 API 請求測試：

- 準備包含圖片數據的 `input.json`。
- 使用 `curl` 發送 POST 請求至模型預測端點 (`/v1/models/flower-sample:predict`)。

- 測試結果： 伺服器回傳 HTTP/1.1 200 OK，表示請求成功。回傳的 JSON 數據中包含 predictions 物件，顯示分類預測值為 0，信心分數 (Score) 高達 0.9991。

```
(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ kubectl get svc istio-ingressgateway -n istio-system
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)                                     AGE
istio-ingressgateway               LoadBalancer        10.98.128.238    <pending>         15021:30938/TCP,80:31818/TCP,443:31134/TCP  8m28s
(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ export INGRESS_HOST=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ export INGRESS_PORT=$(kubectl -n istio-system get service istio-ingressgateway -o jsonpath='{.spec.ports[?(@.name=="http2")].port}')
(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ INGRESS_GATEWAY_SERVICE=$(kubectl get svc --namespace istio-system --selector="app=istio-ingressgateway" --output jsonpath='{.items[0].metadata.name}')
(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ kubectl port-forward --namespace istio-system svc/${INGRESS_GATEWAY_SERVICE} 8080:80
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from ::1:8080 -> 8080

(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ kubectl get isvc flower-sample
NAME      URL                                     READY  PREV  LATEST  PREVROLLOUTREVISION  LATESTREADYRE
VISION    AGE
flower-sample  http://flower-sample.default.example.com  True    100
-predictor-00001  52s
(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ |

(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ curl -v -H "Host: ${SERVICE_HOSTNAME}" http://${INGRESS_HOST}:${INGRESS_PORT}/v1/models/${MODEL_NAME}:predict -d @./input.json
* Trying 127.0.0.1:8080...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /v1/models/flower-sample:predict HTTP/1.1
> Host: flower-sample.default.example.com
> User-Agent: curl/7.65.3
> Accept: */*
> Content-Length: 16201
> Content-Type: application/x-www-form-urlencoded
> Expect: 100-continue
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 100 Continue
* We are completely uploaded and fine
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< content-length: 221
< content-type: application/json
< date: Mon, 08 Dec 2025 07:31:28 GMT
< x-envoy-upstream-service-time: 956
< server: istio-envoy
<
{
  "predictions": [
    {
      "scores": [0.999114931, 9.20989623e-05, 0.000136786344, 0.00033725804, 0.000300533167, 1.84813962e-05],
      "prediction": 0,
      "key": " 1"
    }
  ]
}
* Connection #0 to host localhost left intact
(base) esl@esl-ROG-STRIX-G15CF-G1:~/kserve_flow$ |
```