

Practical-10

Orchestration of ML project containers using Kubernetes

The objective of this lab is to introduce you to the fundamentals of orchestrating applications with Kubernetes. You will learn how to define, deploy, and manage containerized applications using Kubernetes manifests.

Lab Steps:

Step 1: Verify Kubernetes Cluster Ensure your Kubernetes cluster is up and running by checking the cluster nodes

```
PS D:\Desktop\stream> kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
docker-desktop      Ready    control-plane  22m   v1.27.2
```

Step 2: Define a Deployment using YAML manifest and apply the deployment to your cluster

```
deployment.yml
1  # deployment.yml
2  apiVersion: apps/v1
3  kind: Deployment
4  metadata:
5    name: ml-deployment
6  spec:
7    replicas: 3
8    selector:
9      matchLabels:
10     app: ml-app
11  template:
12    metadata:
13     labels:
14     app: ml-app
15    spec:
16     containers:
17     - name: ml-container
18       image: your-ml-image:tag
19       ports:
20       - containerPort: 8080
21
```

Apply the deployment :

```
PS D:\Desktop\stream> kubectl apply -f deployment.yaml
deployment.apps/ml-deployment created
```

Step 3: Describe Deployment

```
PS D:\Desktop\stream> kubectl describe deployment ml-deployment
Name: ml-deployment
Namespace: default
CreationTimestamp: Thu, 23 Nov 2023 18:58:29 +0530
Labels: <none>
Annotations: deployment.kubernetes.io/revision: 1
Selector: app=ml-app
Replicas: 3 desired | 3 updated | 3 total | 0 available | 3 unavailable
StrategyType: RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels: app=ml-app
  Containers:
    ml-container:
      Image: your-ml-image:tag
      Port: 8080/TCP
      Host Port: 0/TCP
      Environment: <none>
      Mounts: <none>
  Volumes: <none>
Conditions:
  Type           Status  Reason
  ----           -
  Available      False   MinimumReplicasUnavailable
  Progressing    True    ReplicaSetUpdated
OldReplicaSets: <none>
NewReplicaSet: ml-deployment-5fcc5656fc (3/3 replicas created)
Events:
  Type    Reason             Age   From                  Message
  ----    -
  Normal  ScalingReplicaSet  24s   deployment-controller  Scaled up replica set ml-deployment-5fcc5656fc to 3
```

Step 4 : Expose Service

```
service.yaml
1  # service.yaml
2  apiVersion: v1
3  kind: Service
4  metadata:
5    name: ml-service
6  spec:
7    selector:
8      app: ml-app
9    ports:
10     - protocol: TCP
11       port: 80
12       targetPort: 8080
13    type: LoadBalancer
14
```

Step 5: Access the Service

```
PS D:\Desktop\stream> kubectl apply -f service.yaml
service/ml-service created
```

Step 6: Scale Deployment

```
PS D:\Desktop\stream> kubectl scale deployment ml-deployment --replicas=5
deployment.apps/ml-deployment scaled
```

Step 7: Update Deployment

```
deployment-updated.yaml
1 # deployment-update.yaml
2 apiVersion: apps/v1
3 kind: Deployment
4 metadata:
5   name: ml-deployment
6 spec:
7   replicas: 3
8   selector:
9     matchLabels:
10      app: ml-app
11   template:
12     metadata:
13       labels:
14         app: ml-app
15     spec:
16       containers:
17       - name: ml-container
18         image: your-updated-ml-image:tag
19         ports:
20         - containerPort: 8080
21
```

Step 8: Rollout Status

```
PS D:\Desktop\stream> kubectl rollout status deployment ml-deployment
Waiting for deployment "ml-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
```

Step 9: Rollback Deployment

```
PS D:\Desktop\stream> kubectl rollout undo deployment ml-deployment
deployment.apps/ml-deployment rolled back
```

Step 10: Delete Resources

```
PS D:\Desktop\stream> kubectl delete deployment ml-deployment
deployment.apps "ml-deployment" deleted
PS D:\Desktop\stream> kubectl delete service ml-service
service "ml-service" deleted
```