DevOps Internship Task 5 Report

Build a Kubernetes Cluster Locally with Minikube

Soumen Das

November 20, 2025

1 Objective

The objective of this task is to deploy and manage applications in a Kubernetes environment using Minikube. This report documents the creation of deployment and service configurations, the execution of verification commands, and the scaling of the application.

2 Tools Used

- **Minikube**: Local Kubernetes cluster.
- **Kubectl**: Command-line tool for interacting with the cluster.
- **Docker**: Container runtime environment.

3 Step 1: Starting the Cluster

To begin the task, Minikube was installed and started to initialize the local Kubernetes cluster.

```
# command to start minikube
minikube start
```

Listing 1: Start Minikube

Status Verification:

```
kubectl cluster-info
kubectl get nodes
```

Listing 2: Verify Cluster Status

4 Step 2: Creating the Deployment

A `deployment.yaml` file was created to manage the application pods. For this task, we used the standard Nginx image as the application example.

4.1 deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Listing 3: deployment.yaml configuration

4.2 Applying the Deployment

The deployment was applied to the cluster using the following command:

```
kubectl apply -f deployment.yaml
```

5 Step 3: Exposing the App (Service)

To make the application accessible, a `service.yaml` file was created using the `NodePort` type.

5.1 service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30007
```

Listing 4: service.yaml configuration

5.2 Applying the Service

The service was applied using the following command:

```
kubectl apply -f service.yaml
```

6 Step 4: Verification and Access

We verified the status of the pods and services to ensure the application is running correctly.

```
# Check pods status
kubectl get pods

# Check service status
kubectl get services

# Get the URL to access the service in Minikube
minikube service nginx-service --url
```

Listing 5: Verification Commands

7 Step 5: Scaling the Deployment

As per the task requirements, the deployment was scaled to demonstrate Kubernetes' orchestration capabilities. We increased the replicas from 2 to 4.

```
# Scale the deployment to 4 replicas
kubectl scale deployment nginx-deployment --replicas=4

# Verify the new pods are creating
kubectl get pods
```

Listing 6: Scaling Command

8 Step 6: Logging and Debugging

To inspect the application details and logs, the `describe` command was used.

```
# Get detailed information about a specific pod
kubectl describe pod <pod-name>

# Fetch logs from a specific pod
kubectl logs <pod-name>
```

Listing 7: Describe and Logs

9 Conclusion

This task successfully demonstrated the basics of Kubernetes operations. We established a local cluster using Minikube, deployed a containerized application using YAML definitions, exposed it via a Service, and performed management tasks such as scaling and log inspection.