

Ragul R R
Date: 07.02.2025

Kubernetes Deployment Guide

Introduction

This document provides a guide on deploying a sample application using Kubernetes. The process includes creating a Deployment and a Service configuration, applying them using kubectl, and accessing the deployed service through Minikube.

UNIX/Linux Commands

- **cat**: Short for "concatenate," used to display the content of files.
- **vim**: Opens the Vim text editor, a highly configurable and powerful text editor commonly used in UNIX/Linux environments.

```
ubuntu@LAPTOP-DEQKQVPU:~$ minikube start
🐳 minikube v1.35.0 on Ubuntu 24.04 (amd64)
🌟 Using the docker driver based on existing profile
👉 Starting "minikube" primary control-plane node in "minikube" cluster
🔄 Pulling base image v0.0.46 ...
🔄 Updating the running docker "minikube" container ...
🔄 Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
🔍 Verifying Kubernetes components...
   ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟 Enabled addons: storage-provisioner, default-storageclass
🏁 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Kubernetes Commands

Apply Deployment Configuration

```
kubectl apply -f t1.txt
```

```
ubuntu@LAPTOP-DEQKQVPU:~$ cat file1.txt
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: test
  name: test
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test
  template:
    metadata:
      labels:
        app: test
    spec:
      containers:
      - name: test
        image: rrragul/sample
        imagePullPolicy: Always
        ports:
        - containerPort: 80
          name: http
          protocol: TCP
```

```
ubuntu@LAPTOP-DEQKQVPU:~$ kubectl apply -f file1.txt
deployment.apps/test unchanged
```

This command applies the Deployment configuration and creates the deployment named test.

Apply Service Configuration

kubectl apply -f t2.txt

```
ubuntu@LAPTOP-DEQKQVPU:~$ cat file2.txt
apiVersion: v1
kind: Service
metadata:
  name: test-service # Corrected the name field
  labels:
    app: test
spec:
  selector:
    app: test # Ensures it matches the label in the corresponding deployment/pod
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: 80
  type: NodePort # Exposes service on a node port
```

```
ubuntu@Harz-PC:~$ kubectl apply -f t2.txt
service/test-service created
```

This command applies the Service configuration and creates the service named test-service.

Access the Service

minikube service test-service

```
ubuntu@LAPTOP-DEQKQVPU:~$ minikube service test-service
-----
| NAMESPACE | NAME       | TARGET PORT | URL                |
|-----|-----|-----|-----|
| default | test-service | http/80      | http://192.168.49.2:32279 |
|-----|-----|-----|-----|
🚀 Starting tunnel for service test-service.
-----
| NAMESPACE | NAME       | TARGET PORT | URL                |
|-----|-----|-----|-----|
| default | test-service |             | http://127.0.0.1:40657 |
|-----|-----|-----|-----|
🌐 Opening service default/test-service in default browser...
👉 http://127.0.0.1:40657
⚠️ Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

This command starts a tunnel for the service and provides a URL to access the deployed application.

Output

