Kubernetes Deployment Setup for `ic-webapp`

Introduction

This document outlines the steps to set up and deploy the `ic-webapp` application on a Kubernetes cluster using Minikube with Docker as the driver.

Prerequisites

- **Docker**: Ensure Docker is installed and running.
- **Minikube**: Installed and configured to use Docker driver.
- **kubectl**: Command-line tool for interacting with the Kubernetes cluster.

Installation Steps

Installing Docker

1. Download Docker Desktop from the Docker website. 2. Follow the installation instructions on the website. 3. After installation, ensure Docker is running.

Installing Minikube

1. Download the Minikube installer for Windows from the Minikube website. 2. Install Minikube by running the installer. 3. Ensure Minikube uses Docker as the driver:

```
minikube config set driver docker
```

Installing kubectl

1. Download the kubectl binary for Windows from the Kubernetes website. 2. Add the binary to your system's PATH by following the installation instructions on the website. 3. Verify the installation:

```
kubectl version --client
```

Setup Steps

1. Start Minikube

```
minikube start --driver=docker --kubernetes-version=v1
    .20.0
```

2. Create Namespace

Create a namespace for organizing resources.

```
# namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: icgroup
```

Apply the namespace:

```
kubectl apply -f namespace.yaml
```

3. Create Deployment

Define the deployment for `ic-webapp`.

```
# deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ic-webapp
  namespace: icgroup
  labels:
    env: prod
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ic-webapp
  template:
    metadata:
      labels:
        app: ic-webapp
    spec:
      containers:
        - name: ic-webapp
          image: imane123456788/file_rouge:v1
          ports:
            - containerPort: 8080
          resources:
            requests:
              memory: "256Mi"
              cpu: "500m"
            limits:
              memory: "512Mi"
              cpu: "1"
```

Apply the deployment:

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

4. Create Services

NodePort Service

Expose the deployment using NodePort.

```
# nodeport-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: ic-webapp-nodeport
  namespace: icgroup
spec:
  selector:
    app: ic-webapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
      nodePort: 30007
  type: NodePort
```

Apply the service:

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

LoadBalancer Service

Optionally, expose the deployment using LoadBalancer.

```
# loadbalancer-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: ic-webapp-loadbalancer
  namespace: icgroup
spec:
  selector:
    app: ic-webapp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

Apply the service:

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

5. Verify Services

Check the status of the services:

```
kubectl get services -n icgroup
```

6. Access the Application

Using NodePort

Find the Minikube IP:

```
minikube ip
```

Access the application at `http://<minikube-ip>:30007`.

Using Minikube Service Command

```
minikube service ic-webapp-nodeport -n icgroup --url
```

Using Port Forwarding

```
kubectl port-forward svc/ic-webapp-nodeport 8080:80 -n
    icgroup
```

Access the application at `http://localhost:8080`.

Troubleshooting

External IP Pending

If the external IP for LoadBalancer is pending, use NodePort or `minikube service` command.

Cannot Access NodePort

Ensure Docker for Windows networking is configured correctly. Use port forwarding as an alternative.

References

- Minikube Documentation
- Kubernetes Documentation
- Docker Documentation