CLOUD-NATIVE APPLICATIONS DEPLOYMENT

Overview

Cloud-native containerized applications require multiple Kubernetes resources due to their complex nature. In this lab, you will create an instance of WordPress application on Kubernetes by using one statefulset, one deployment, and two service resources. At the end, you will check the status of the Pods and connect to the Service using Kubectl and minikube.

Procedure

Creating a StatefulSet definition

- Create a StatefulSet definition in a file, named database.yaml, with the following content. Your task is to ensure YAML file has an appropriate indentation so that there are no syntax errors.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
name: database
spec:
selector:
matchLabels:
app: mysql
serviceName: mysql
replicas: 1
template:
metadata:
labels:
app: mysql
spec:
containers:
- name: mysql
image: mysql:5.7
env:
- name: MYSQL_ROOT_PASSWORD
value: "root"
ports:
- name: mysql
containerPort: 3306
volumeMounts:
- name: data
mountPath: /var/lib/mysql
subPath: mysql
volumeClaimTemplates:
- metadata:
```

```
name: data
spec:
accessModes: ["ReadWriteOnce"]
resources:
requests:
storage: 2Gi
```

This StatefulSet resource defines a database to be used by WordPress. There is only one container named mysql with the Docker image of mysql:5.7. There is one environment variable for the root password and one port defined in the container specification.

- Deploy the StatefulSet to the cluster by running the following command in your terminal:

```
kubectl apply -f database.yaml
```

This command will apply the definition in the database.yaml file since it is passed with the -f flag.

Creating a Database service

- Create a database-service.yaml file with the following content. Your task is to ensure YAML file has an appropriate indentation so that there are no syntax errors.

```
apiVersion: v1
kind: Service
metadata:
name: database-service
spec:
selector:
app: mysql
ports:
- protocol: TCP
port: 3306
targetPort: 3306
```

This Service resource defines a Service abstraction over database instances. WordPress instances will connect to the database by using the specified Service.

- Deploy the Service resource with the following command:

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

This command deploys the resource defined in the database-service.yaml file.

- List all the services with the following command:

```
kubectl get service
```

This command lists all the services with cluster IP address and Ports assigned by Kubernetes. **(Provide a screenshot in your lab report.)**

- Describe the database-service with the following command:

kubectl describe service/database-service
This command provides detailed information about the database service. **(Provide a screenshot in your lab report.)**

Application Deployment

- Create a file with the name wordpress.yaml and the following content. Your task is to ensure YAML file has an appropriate indentation so that there are no syntax errors.

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: wordpress
labels:
app: wordpress
spec:
replicas: 3
selector:
matchLabels:
app: wordpress
template:
metadata:
labels:
app: wordpress
spec:
containers:
- image: wordpress:4.8-apache
name: wordpress
env:
- name: WORDPRESS_DB_HOST
value: database-service
- name: WORDPRESS_DB_PASSWORD
value: root
ports:
- containerPort: 80
name: wordpress
```

This Deployment resource defines a three-replica WordPress installation. There is one container defined with the wordpress image and database-service is passed to the application as an environment variable. With the help of this environment variable, WordPress connects to the database.

- Deploy the WordPress Deployment with the following command:

```
kubectl apply -f wordpress.yaml
```

This command deploys the resource defined in the wordpress.yaml file.

- List all the deployment with the following command:

```
kubectl get deployment
```

This command lists the deployment with status and the number of replicaset. **(Provide a screenshot in your lab report.)**

- Describe the wordpress deployment with the following command:

  kubectl describe deployment/wordpress
  This command provides detailed information about the wordpress deployment. **(Provide a screenshot in your lab report.)**

Creating an Application Service

- Create a wordpress-service.yaml file with the following content. Your task is to ensure YAML file has an appropriate indentation so that there are no syntax errors.

  apiVersion: v1
  kind: Service
  metadata:
  name: wordpress-service
  spec:
  type: LoadBalancer
  selector:
  app: wordpress
  ports:
  - protocol: TCP
  port: 80
  targetPort: 80

This Service resource defines a Service abstraction over the WordPress instances. The Service will be used to connect to WordPress from the outside world via port 80.

- Deploy the Service resource with the following command:

  kubectl apply -f wordpress-service.yaml

This command deploys the resource defined in the wordpress-service.yaml file.

- List all the services with the following command:

  kubectl get service

This command lists all the services with cluster IP address and Ports assigned by Kubernetes. **(Provide a screenshot in your lab report.)**

- Describe the wordpress-service with the following command:

  kubectl describe service/wordpress-service
  This command provides detailed information about the wordpress service. **(Provide a screenshot in your lab report.)**

- Check the status of all running Pods with the following command:

  kubectl get pods

This command lists all the Pods with their statuses, and there are one database and three WordPress Pods with the Running status. **(Provide a screenshot in your lab report.)**

- Get the URL of wordpress-service by running the following command:

  minikube service wordpress-service --url

This command lists the URL of the Service, accessible from the host machine. **(Provide a screenshot in your lab report.)**