Kubernetes volumes

To design a resilient and scalable deployment for a web application with a MySQL database using Docker and Kubernetes, we'll address each requirement step-by-step. Here, I'll provide Kubernetes YAML configurations for each part of the system:

### A) Create Persistent Volume (PV)

A Persistent Volume (PV) in Kubernetes is a piece of storage in the cluster that has been provisioned by an administrator. It is a resource in the cluster just like a node is a cluster resource.

Here’s an example of a Persistent Volume configuration:

Step 1 : create a volume yaml file

vim pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-pv

labels:

type: local

spec:

storageClassName: manual

capacity:

storage: 10Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/mnt/data/mysql"

step 2: deploy the volume

kubectl create -f pv.yaml

step 3: list the volume

kubectl get pv

### B) Create Persistent Volume Claim (PVC)

A Persistent Volume Claim (PVC) is a request for storage by a user. It is similar to a pod. Pods consume node resources and PVCs consume PV resources.

Here’s how you can define a PVC to match the PV:

Step 4: create PVC yaml file

vim pvc.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pvc

spec:

storageClassName: manual

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 10Gi

selector:

matchLabels:

type: local

Step 5: Deploy the PVC

kubectl create -f pvc.yaml

step 6 : list the PVC

kubectl get pvc

### C) MySQL Deployment with a Single Replica

This configuration defines a Kubernetes Deployment for MySQL. It uses the PVC to ensure the database data persists across pod restarts and failures.

Step 7: create the mysql yaml file

vim mysql.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql

strategy:

type: Recreate

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: mysql:5.7

env:

- name: MYSQL\_ROOT\_PASSWORD

value: "yourpassword" # Use a secret in production

ports:

- containerPort: 3306

volumeMounts:

- name: mysql-storage

mountPath: /var/lib/mysql

volumes:

- name: mysql-storage

persistentVolumeClaim:

claimName: mysql-pvc

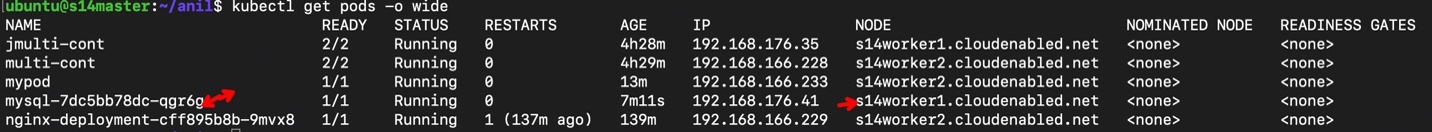
step 8: deploy the mysql with pv

kubectl create -f mysql.yaml

step 9 : validate on which node POD is deployed and make a note

kubectl get pods -o wide

validate on which node its running



Step 10 : Ssh to the node on which POD is running ( worker14 in my scenario)

ssh ubuntu@s14worker1

Step 11 : You can verify the mysql database in actually on volume and not inside pods

ls -l /mnt/data/mysql

A screenshot of a computer

Description automatically generated

=============== Lab completes =================