Kubernetes 6

WorkerNode 1

Control plane

kubectl

WorkerNode 2

WorkerNode N

EKS Host VM

Whenever we create a pod to deploy the application, are you sure out of all worker nodes we have which worker node our Pod will be in. No. Can you be sure your pod will be in a specific worker node? No. I want my Pods to be created in all Worker Nodes. DaemonSet (Create a Pod in each worker node). To get Logs from each of the Worker nodes, there is one concept called as Kibana, FluentD, ElasticSearch (EFK).

<https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>

A DaemonSet ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

Stateless pod (no storage, no data is storage) and Stateful pod (all data will be maintained)

StatefulSet: It will create Stateful application (Ex: Database pods)

PV & PVC: Storage (Persistent Volume & Persistent Volume Clain) --> Used to manage the persistent storage --> To retain the data even if the pod is deleted or restarted (data is restored)

ConfigMap & Secrets --> To supply environment variables (Ex: DB credentials)

Docker image

JAR / WAR file

Source code

Production

UAT

SIT

Dev

UAT DB

SIT DB

Dev DB

We package the application source code into a JAR or WAR file then we create the DockerImage. Can we use the same DockerImage in all environments? Yes if we don’t hard-code. We can make application loosely coupled so the same DockerImage could be deployed in all environments. that’s where ConfigMaps and Secrets come into picture. We can externalize environment-specific values like Database credentials, URLs, and keys. DockerImage gets deployed into Kubernetes Cluster Pod.

Pod gets deployed into different envoronments

Kubernetes

DockerImage

Pod

Secrets

ConfigMap

Sensitive data

TextData which is non-sensitive

We can deploy same Docker image into multiple environments (Dev, SIT, UAT etc) without modifying the image itself.

The other concept is Ingress controller:

It’s like the Front controller, the one that faces the request. Ingress controller receives the request and which request should go to which Service. Basically the routing work is done by Ingress controller. To route incoming traffic to a particular service in the cluster. Readyness and Liveness probe to make sure Pods are ready and alive to receive requests.

Create K8s cluster

eksctl create cluster --name my-eks-cluster --region ca-central-1 --node-type t2.medium --zones ca-central-1a,ca-central-1b

ConfigMap manifest:

1. YML file  
   Database deployment
2. YML file

Application Pod

Manifest

1. Create a YML file

ConfigMap manifest

1. YML file

Secret manifest

App Pod will be created with App Pod manifest Yml file

DB Pod

End goal is Application Pod must be able to connect to DB Pod. Config values are passed through ConfigMap and Secret manifests. To be able to connect App Pod with DB Pod, we require ConfigMap and Secret manifest to be passed into App Pod manifest as well. that’s where our Pod will be able to make a connection with Database

If you go to this application.properties file, we can see Config values are passed dynamically. Environmental variables with default values if not passed

<https://github.com/Haider7214/spring-boot-mysql/blob/main/src/main/resources/application.properties>

# Datasource settings

spring.datasource.url=${DB\_URL:jdbc:mysql://mysqldb:3306/sbms}

spring.datasource.username=${DB\_USERNAME:root}

spring.datasource.password=${DB\_PASSWORD:root123}

# JPA settings

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

ubuntu@ip-172-31-9-165:~$ mkdir config-map-secret-manifest

ubuntu@ip-172-31-9-165:~$ cd config-map-secret-manifest/

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ ls -l

total 0

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 1-demo-db-configmap.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 1-demo-db-configmap.yml

apiVersion: v1

kind: ConfigMap

metadata:

name: demo-db-config-map

namespace: default # Change if needed

labels:

storage: demo-db-storage

data:

DB\_HOST\_SERVICE\_NAME\_VALUE: demo-app-db-service

DB\_PORT\_VALUE: "3306" # Or "3306" for MySQL

DB\_SCHEMA\_VALUE: demo-mkdapp

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 2-demo-db-secret.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 2-demo-db-secret.yml

---

apiVersion: v1

kind: Secret

metadata:

name: demo-db-config-secret

namespace: default # Change namespace if needed

labels:

secrete: demo-db-config-secrete

type: Opaque

data:

DB\_USER: cm9vdA== # base64 for "root"

DB\_PASSWORD: cm9vdDEyMw== # base64 for "root123"

...

Encoded DB credentials using Base64 only: <https://www.base64encode.org/>

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 2-demo-db-secret.yml

---

apiVersion: v1

kind: Secret

metadata:

name: demo-db-config-secret

namespace: default # Change namespace if needed

labels:

secrete: demo-db-config-secrete

type: Opaque

data:

DB\_USER: cm9vdA== # base64 for "root"

DB\_PASSWORD: cm9vdDEyMw== # base64 for "root123"

…

To make DB pod persistent, we add PV and PVC yml files. Even when the Pod is deleted, I want the data to be there

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 3-demo-db-pv.yml

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: demo-db-pv

labels:

name: demo-db-pv

spec:

capacity:

storage: 4Gi

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Retain

storageClassName: local-storage

hostPath:

path: /opt/mysql

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 4-demo-db-pvc.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 4-demo-db-pvc.yml

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: demo-db-pvc

spec:

volumeName: demo-db-pv

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 4Gi

storageClassName: local-storage

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 5-demo-db-deployment.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 5-demo-db-deployment.yml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: demo-app-db-deployment

labels:

app: demo-app-db

spec:

replicas: 1

selector:

matchLabels:

app: demo-app-db-pod

template:

metadata:

labels:

app: demo-app-db-pod

spec:

containers:

- name: demo-app-db

image: mysql:8.0

ports:

- containerPort: 3306

volumes:

- name: demo-app-db-volume

persistentVolumeClaim:

claimName: demo-db-pvc

---

apiVersion: v1

kind: Service

metadata:

name: demo-app-db-service

labels:

app: demo-app-db-service

spec:

type: ClusterIP

selector:

app: demo-app-db-pod

ports:

- protocol: TCP

port: 3306

targetPort: 3306

clusterIP: None # Headless service for stable DNS (optional, for StatefulSets or direct pod access)

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 5-demo-db-deployment.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 5-demo-db-deployment.yml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: demo-app-db-deployment

labels:

app: demo-app-db

spec:

replicas: 1

selector:

matchLabels:

app: demo-app-db-pod

template:

metadata:

labels:

app: demo-app-db-pod

spec:

containers:

- name: mysql

image: mysql:8.0

ports:

- containerPort: 3306

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: demo-db-config-secrete

key: DB\_PASSWORD\_VALUE

- name: MYSQL\_DATABASE

valueFrom:

configKeyRef:

name: demo-db-config-map

key: DB\_SCHEMA\_VALUE

volumeMounts:

- name: demo-app-db-volume

mountPath: /var/lib/mysql

volumes:

- name: demo-app-db-volume

persistentVolumeClaim:

claimName: demo-db-pvc

---

apiVersion: v1

kind: Service

metadata:

name: demo-app-db-service

labels:

app: demo-app-db-service

spec:

type: ClusterIP

selector:

app: demo-app-db-pod

ports:

- protocol: TCP

port: 3306

targetPort: 3306

clusterIP: None # Headless service for stable DNS (optional, for StatefulSets or direct pod access)

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 6-app-deployment.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 6-app-deployment.yml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: spring-boot-mysql

spec:

replicas: 1

selector:

matchLabels:

app: spring-boot-mysql

template:

metadata:

labels:

app: spring-boot-mysql

spec:

containers:

- name: spring-boot-mysql

image: hacker123shiva/springboot-mysql-app:latest

ports:

- containerPort: 8080

env:

- name: DB\_HOST

valueFrom:

configMapKeyRef:

name: demo-db-config-map

key: DB\_HOST\_SERVICE\_NAME\_VALUE

- name: DB\_NAME

valueFrom:

configMapKeyRef:

name: demo-db-config-map

key: DB\_SCHEMA\_VALUE

- name: DB\_USERNAME

valueFrom:

secretKeyRef:

name: demo-db-config-secrete

key: DB\_USER\_NAME\_VALUE

- name: DB\_PASSWORD

valueFrom:

secretKeyRef:

name: demo-db-config-secrete

key: DB\_PASSWORD\_VALUE

---

apiVersion: v1

kind: Service

metadata:

name: springboot-mysql-svc

spec:

type: NodePort

selector:

app: spring-boot-mysql

ports:

- protocol: TCP

port: 8080

targetPort: 8080

nodePort: 30785

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ ls -l

total 24

-rw-rw-r-- 1 ubuntu ubuntu 295 Jun 8 22:28 1-demo-db-configmap.yml

-rw-rw-r-- 1 ubuntu ubuntu 295 Jun 8 22:54 2-demo-db-secret.yml

-rw-rw-r-- 1 ubuntu ubuntu 284 Jun 8 23:18 3-demo-db-pv.yml

-rw-rw-r-- 1 ubuntu ubuntu 228 Jun 8 23:22 4-demo-db-pvc.yml

-rw-rw-r-- 1 ubuntu ubuntu 1258 Jun 9 00:01 5-demo-db-deployment.yml

-rw-rw-r-- 1 ubuntu ubuntu 1331 Jun 9 00:41 6-app-deployment.yml

Creating configMap

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 1-demo-db-configmap.yml

configmap/demo-db-config-map created

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get cm

NAME DATA AGE

demo-db-config-map 3 23s

kube-root-ca.crt 1 3h18m

Creating secret

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 2-demo-db-secret.yml

secret/demo-db-config-secret created

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get secret

NAME TYPE DATA AGE

demo-db-config-secret Opaque 2 18s

Creating PV

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 3-demo-db-pv.yml

persistentvolume/demo-db-pv created

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIBUTESCLASS REASON AGE

demo-db-pv 4Gi RWO Retain Available local-storage <unset> 22s

Creating PVC

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 4-demo-db-pvc.yml

persistentvolumeclaim/demo-db-pvc created

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE

demo-db-pvc Pending demo-db-pv 0 local-storage <unset> 11s

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 5-demo-db-deployment.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 5-demo-db-deployment.yml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: demo-app-db-deployment

labels:

app: demo-app-db

spec:

replicas: 1

selector: # This is important and was misplaced in your YAML

matchLabels:

app: demo-app-db-pod

template:

metadata:

labels:

app: demo-app-db-pod

spec:

volumes:

- name: demo-app-db-volume

persistentVolumeClaim:

claimName: demo-db-pvc

containers:

- name: demo-app-db

image: mysql:8.0 # Add a version tag to ensure consistency

ports:

- containerPort: 3306

volumeMounts:

- name: demo-app-db-volume

mountPath: /var/lib/mysql # ✅ MySQL expects data here, not /opt/mysql

env:

- name: MYSQL\_ROOT\_PASSWORD

valueFrom:

secretKeyRef:

name: demo-db-config-secrete

key: DB\_PASSWORD\_VALUE

- name: MYSQL\_DATABASE

valueFrom:

configMapKeyRef:

name: demo-db-config-map

key: DB\_SCHEMA\_VALUE

---

apiVersion: v1

kind: Service

metadata:

name: demo-app-db-service

labels:

app: demo-app-db-service

spec:

type: ClusterIP

ports:

- port: 3306

targetPort: 3306

protocol: TCP

selector:

app: demo-app-db-pod

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 5-demo-db-deployment.yml

deployment.apps/demo-app-db-deployment created

service/demo-app-db-service configured

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get pods

NAME READY STATUS RESTARTS AGE

demo-app-db-deployment-5667b4fdd9-wll8d 0/1 CreateContainerConfigError 0 33s

Some error trying to fix

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 1-demo-db-configmap.yml

---

apiVersion: v1

kind: ConfigMap

metadata:

name: demo-db-config-map # Name of the ConfigMap

labels:

storage: demo-db-storage # Optional: label for categorization or selection

data:

DB\_HOST\_SERVICE\_NAME\_VALUE: demo-app-db-service

DB\_SCHEMA\_VALUE: demo-mkdapp

DB\_PORT\_VALUE: "3306"

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 2-demo-db-secret.yml

---

apiVersion: v1

kind: Secret

metadata:

name: demo-db-config-secrete # Name of the secret

labels:

secrete: demo-db-config-secrete # (Optional) label for identification

data:

DB\_USER\_NAME\_VALUE: cm9vdA== # Base64 for "root"

DB\_PASSWORD\_VALUE: cm9vdDEyMw== # Base64 for "root"

type: Opaque

...

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ vi 3-demo-db-pv.yml

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ cat 3-demo-db-pv.yml

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: demo-db-pv

labels:

name: demo-db-pv

spec:

capacity:

storage: 4Gi # 4 GiB of storage

accessModes:

- ReadWriteOnce # Only one node can mount it read-write

persistentVolumeReclaimPolicy: Retain # Keeps the data even after PVC is deleted

storageClassName: local-storage # Must match with the PVC's storageClassName

hostPath:

path: /opt/mysql

...

Now everything is up and running

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 1-demo-db-configmap.yml

configmap/demo-db-config-map unchanged

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 2-demo-db-secret.yml

secret/demo-db-config-secrete created

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 3-demo-db-pv.yml

persistentvolume/demo-db-pv unchanged

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 4-demo-db-pvc.yml

persistentvolumeclaim/demo-db-pvc unchanged

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl apply -f 5-demo-db-deployment.yml

deployment.apps/demo-app-db-deployment unchanged

service/demo-app-db-service unchanged

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get pods

NAME READY STATUS RESTARTS AGE

demo-app-db-deployment-5667b4fdd9-wll8d 1/1 Running 0 61m

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

demo-app-db-service ClusterIP None <none> 3306/TCP 68m

kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 4h36m

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get cm

NAME DATA AGE

demo-db-config-map 3 79m

kube-root-ca.crt 1 4h38m

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get secret

NAME TYPE DATA AGE

demo-db-config-secret Opaque 2 78m

demo-db-config-secrete Opaque 2 3m16s

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS VOLUMEATTRIBUTESCLASS REASON AGE

demo-db-pv 4Gi RWO Retain Bound default/demo-db-pvc local-storage <unset> 77m

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS VOLUMEATTRIBUTESCLASS AGE

demo-db-pvc Bound demo-db-pv 4Gi RWO local-storage <unset> 75m

To check whether MySQL is running inside this Pod

ubuntu@ip-172-31-9-165:~/config-map-secret-manifest$ kubectl exec -it demo-app-db-deployment-5667b4fdd9-wll8d -- bash

bash-5.1# mysql -h localhost -u root -p root

Enter password:

ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

bash-5.1# mysql -u root -p

Enter password:

ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)

bash-5.1# mysql -u root -p root

Enter password:

ERROR 1049 (42000): Unknown database 'root'

bash-5.1# mysql -u root -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 11

Server version: 8.0.42 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its

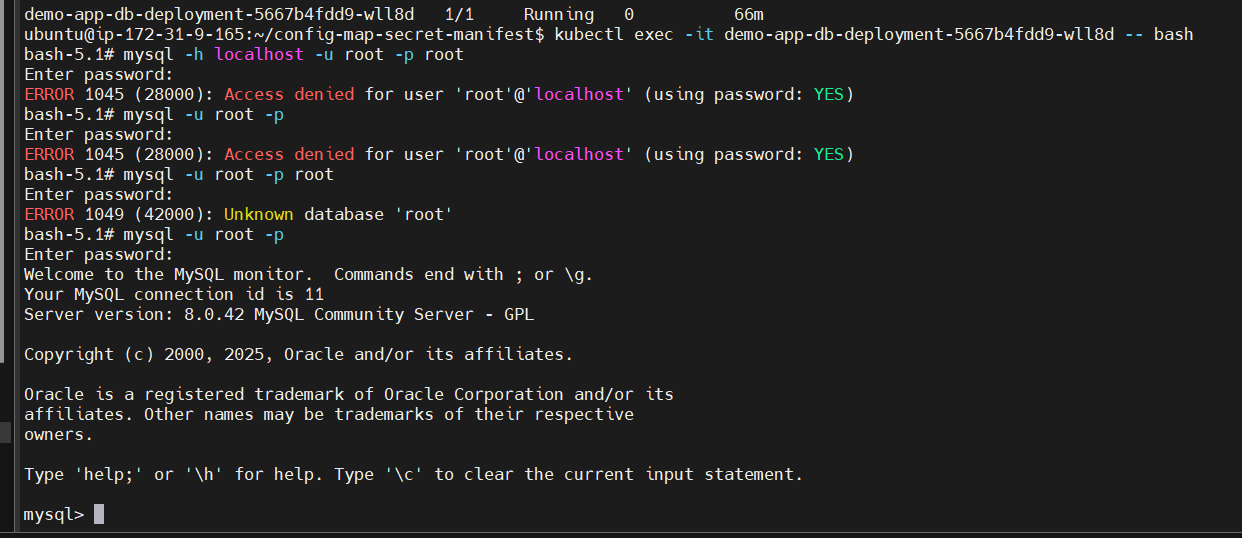
affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

Password is root123



mysql> show databases;

+--------------------+

| Database |

+--------------------+

| demo-mkdapp |

| information\_schema |

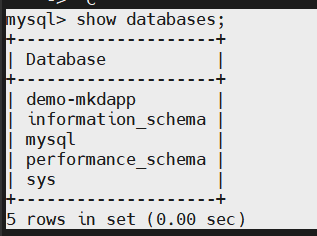
| mysql |

| performance\_schema |

| sys |

+--------------------+

5 rows in set (0.00 sec)



mysql> show databases;

+--------------------+

| Database |

+--------------------+

| demo-mkdapp |

| information\_schema |

| mysql |

| performance\_schema |

| sys |

+--------------------+

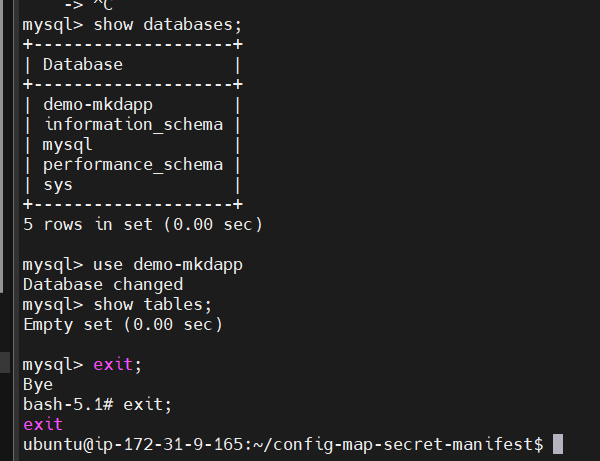
5 rows in set (0.00 sec)

mysql> use demo-mkdapp

Database changed

mysql> show tables;

Empty set (0.00 sec)



1:45:40