**❏Create pods, deployments, services.**

**1. Create Pods, Deployments, Services**

**Pod Definition**:

kubectl run nginx --image=nginx

**Deployment Definition**:

kubectl create deployment nginx-deployment --image=nginx

**Service Definition**:

kubectl expose deployment nginx-deployment --port=80 --target-port=80 --name=nginx-service

**❏Deploy pods on specific node using concept of taint & tolerance, node selector & node affinity**

Apply a taint to the node where you want to schedule the Pods:

kubectl taint nodes nginx key=value:NoSchedule

#### **Adding a Toleration to a Pod**

"apiVersion": "v1",

"spec": {

"tolerations": [

{

"key": "key",

"operator": "Equal",

"value": "value",

"effect": "NoSchedule"

}

]

**❏Explore node affinity, node selector, anti affinity**

#### **Creating a Pod with Node Selector**

"apiVersion": "v1",

"spec": {

"nodeSelector": {

"disktype": "ssd"

#### **Creating a Pod with Node Affinity**

"apiVersion": "v1",

"spec": {

"affinity": {

"nodeAffinity": {

"requiredDuringSchedulingIgnoredDuringExecution": {

"nodeSelectorTerms": [

{

"matchExpressions": [

{

"key": "disktype",

"operator": "In",

"values": [

"ssd"

**❏Deploy mysql pod and store the data in a persistent volume. Delete the pod and attach the volume to a new pod. Check if the data is available to the new pod.**

Create a file named mysql-pv.yaml:

yaml

Copy code

apiVersion: v1

kind: PersistentVolume

metadata:

name: mysql-pv

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: /mnt/data/mysql

storageClassName: manual

Apply the Persistent Volume:

kubectl apply -f mysql-pv.yaml

### 2. Create a Persistent Volume Claim (PVC)

Create a file named mysql-pvc.yaml:

yaml

Copy code

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: mysql-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

storageClassName: manual

Apply the Persistent Volume Claim:

kubectl apply -f mysql-pvc.yaml

### 3. Deploy a MySQL Pod

Create a file named mysql-pod.yaml:

yaml

Copy code

apiVersion: v1

kind: Pod

metadata:

name: mysql-pod

spec:

containers:

- name: mysql

image: mysql:5.7

env:

- name: MYSQL\_ROOT\_PASSWORD

value: password

ports:

- containerPort: 3306

volumeMounts:

- mountPath: /var/lib/mysql

name: mysql-storage

volumes:

- name: mysql-storage

persistentVolumeClaim:

claimName: mysql-pvc

Apply the Pod configuration:

kubectl apply -f mysql-pod.yaml

### 4. Verify Data Persistence

1. **Access the MySQL Pod** and create some test data:

kubectl exec -it mysql-pod -- mysql -u root -ppassword -e "CREATE DATABASE testdb; USE testdb; CREATE TABLE test\_table (id INT PRIMARY KEY, value VARCHAR(100)); INSERT INTO test\_table (id, value) VALUES (1, 'test\_value');"

1. **Check the data**:

kubectl exec -it mysql-pod -- mysql -u root -ppassword -e "USE testdb; SELECT \* FROM test\_table;"

### 5. Delete the MySQL Pod

kubectl delete pod mysql-pod

### 6. Deploy a New MySQL Pod

Create a new Pod configuration file named mysql-pod-new.yaml:

yaml

Copy code

apiVersion: v1

kind: Pod

metadata:

name: mysql-pod-new

spec:

containers:

- name: mysql

image: mysql:5.7

env:

- name: MYSQL\_ROOT\_PASSWORD

value: password

ports:

- containerPort: 3306

volumeMounts:

- mountPath: /var/lib/mysql

name: mysql-storage

volumes:

- name: mysql-storage

persistentVolumeClaim:

claimName: mysql-pvc

Apply the new Pod configuration:

kubectl apply -f mysql-pod-new.yaml

### 7. Verify Data Availability

**Access the new MySQL Pod** and check if the data is still available:

kubectl exec -it mysql-pod-new -- mysql -u root -ppassword -e "USE testdb; SELECT \* FROM test\_table;"

**❏Deploy Wordpress to kubernetes and expose the service to outside world using nodeport, loadbalancerservices.**

**❏Define Readiness & liveness probes for the wordpress container.**