Lesson 04 Demo 02

Configuring Pod Affinity and Anti-affinity in Kubernetes

Objective: To configure pod affinity and anti-affinity rules in a Kubernetes cluster to ensure specific deployment patterns of pods across nodes

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Steps to be followed:

1. Deploy redis-cache with anti-affinity
2. Colocate the web server with redis-cache using affinity

Step 1: Deploy redis-cache with anti-affinity

1.1 Create the **redis-cache-deployment.yaml** configuration file using the following command:
vi redis-cache-deployment.yaml

```
labsuser@master:~$ vi redis-cache-deployment.yaml
labsuser@master:~$
```

1.2 Enter the following code in the **redis-cache-deployment.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-cache
spec:
  selector:
    matchLabels:
      app: store

  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: redis-server
        image: redis:3.2-alpine
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-cache
spec:
  selector:
    matchLabels:
      app: store
  replicas: 3
  template:
    metadata:
      labels:
        app: store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - store
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: redis-server
        image: redis:3.2-alpine

:wq
```

1.3 Apply the **redis-cache-deployment.yaml** configuration file using the following command:
kubectl apply -f redis-cache-deployment.yaml

```
labsuser@master:~$ kubectl apply -f redis-cache-deployment.yaml
deployment.apps/redis-cache created
labsuser@master:~$
```

1.4 Verify the deployment of **redis-cache** using the following commands:
kubectl get deploy redis-cache
kubectl get pod -l app=store -o wide

```
labsuser@master:~$ kubectl apply -f redis-cache-deployment.yaml
deployment.apps/redis-cache created
labsuser@master:~$ kubectl get deploy redis-cache
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
redis-cache   1/3     3            1           6m43s
labsuser@master:~$ kubectl get pod -l app=store -o wide
NAME                           READY   STATUS    RESTARTS   AGE     IP               NODE                      NOMINATED NODE   READINESS GATES
redis-cache-8478cbdc86-74wnv   0/1     Pending   0          6m56s   <none>           <none>                    <none>           <none>
redis-cache-8478cbdc86-qw8jb   0/1     Pending   0          6m56s   <none>           <none>                    <none>           <none>
redis-cache-8478cbdc86-wldjq   1/1     Running   0          6m57s   172.16.232.202   worker-node-2.example.com <none>           <none>
labsuser@master:~$
```

Step 2: Colocate the web server with redis-cache using affinity

2.1 Create the **web-server-deployment.yaml** configuration file using the following command:
vi web-server-deployment.yaml

```
labsuser@master:~$ kubectl apply -f redis-cache-deployment.yaml
deployment.apps/redis-cache created
labsuser@master:~$ kubectl get deploy redis-cache
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
redis-cache   1/3     3            1           6m43s
labsuser@master:~$ kubectl get pod -l app=store -o wide
NAME                           READY   STATUS    RESTARTS   AGE     IP              NODE                       NOMINATED NODE   READINESS GAT
redis-cache-8478cbdc86-74wnv   0/1     Pending   0          6m56s   <none>          <none>                     <none>           <none>
redis-cache-8478cbdc86-qw8jb   0/1     Pending   0          6m56s   <none>          <none>                     <none>           <none>
redis-cache-8478cbdc86-wldjq   1/1     Running   0          6m57s   172.16.232.202  worker-node-2.example.com  <none>           <none>
labsuser@master:~$ kubectl get pod -l app=store -o wide
NAME                           READY   STATUS    RESTARTS   AGE   IP              NODE                       NOMINATED NODE   READINESS GATES
redis-cache-8478cbdc86-74wnv   0/1     Pending   0          17m   <none>          <none>                     <none>           <none>
redis-cache-8478cbdc86-qw8jb   0/1     Pending   0          17m   <none>          <none>                     <none>           <none>
redis-cache-8478cbdc86-wldjq   1/1     Running   0          17m   172.16.232.202  worker-node-2.example.com  <none>           <none>
labsuser@master:~$ vi web-server-deployment.yaml
labsuser@master:~$
```

2.2 Enter the following code in the **web-server-deployment.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-server
spec:
  selector:
    matchLabels:
      app: web-store
  replicas: 3
  template:
    metadata:
      labels:
        app: web-store
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - web-store
            topologyKey: "kubernetes.io/hostname"
        podAffinity:
```

```
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app
          operator: In
          values:
          - store
      topologyKey: "kubernetes.io/hostname"
  containers:
  - name: web-app
    image: nginx:1.16-alpine
```

```
template:
  metadata:
    labels:
      app: web-store
  spec:
    affinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - web-store
          topologyKey: "kubernetes.io/hostname"
      podAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
            - key: app
              operator: In
              values:
              - store
          topologyKey: "kubernetes.io/hostname"
    containers:
    - name: web-app
      image: nginx:1.16-alpine

:wq
```

2.3 Apply the **web-server-deployment.yaml** configuration using the following command:
kubectl apply -f web-server-deployment.yaml

```
redis-cache-8478cbdc86-qw8jb   0/1     Pending   0          17m   <none>
redis-cache-8478cbdc86-wldjq   1/1     Running   0          17m   172.16.232.202
labsuser@master:~$ vi web-server-deployment.yaml
labsuser@master:~$ kubectl apply -f web-server-deployment.yaml
deployment.apps/web-server created
labsuser@master:~$
```

2.4 Verify the deployment of the web server using the following commands:
kubectl get deploy web-server
kubectl get pod -l app=web-store -o wide

```
labsuser@master:~$ kubectl get deploy web-server
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
web-server   1/3     3            1           95s
labsuser@master:~$ kubectl get pod -l app=web-store -o wide
NAME                          READY   STATUS    RESTARTS   AGE    IP               NODE                      NOMINATED NODE   READINESS GATES
web-server-55f57c89d4-8lnnb   0/1     Pending   0          103s   <none>           <none>                    <none>           <none>
web-server-55f57c89d4-kh5st   1/1     Running   0          103s   172.16.232.203   worker-node-2.example.com <none>           <none>
web-server-55f57c89d4-rbxrf   0/1     Pending   0          103s   <none>           <none>                    <none>           <none>
labsuser@master:~$
```

2.5 Check the information of the pods using the following commands:
kubectl get pods -l app=store -o wide
kubectl get pods -l app=web-store -o wide

```
labsuser@master:~$ kubectl get deploy web-server
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
web-server   1/3     3            1           95s
labsuser@master:~$ kubectl get pod -l app=web-store -o wide
NAME                          READY   STATUS    RESTARTS   AGE    IP               NODE                      NOMINATED NODE   READINESS GATES
web-server-55f57c89d4-8lnnb   0/1     Pending   0          103s   <none>           <none>                    <none>           <none>
web-server-55f57c89d4-kh5st   1/1     Running   0          103s   172.16.232.203   worker-node-2.example.com <none>           <none>
web-server-55f57c89d4-rbxrf   0/1     Pending   0          103s   <none>           <none>                    <none>           <none>
labsuser@master:~$ kubectl get pods -l app=store -o wide
NAME                          READY   STATUS    RESTARTS   AGE    IP               NODE                      NOMINATED NODE   READINESS GATES
redis-cache-8478cbdc86-74wnv  0/1     Pending   0          27m    <none>           <none>                    <none>           <none>
redis-cache-8478cbdc86-qw8jb  0/1     Pending   0          27m    <none>           <none>                    <none>           <none>
redis-cache-8478cbdc86-wldjq  1/1     Running   0          27m    172.16.232.202   worker-node-2.example.com <none>           <none>
labsuser@master:~$ kubectl get pods -l app=web-store -o wide
NAME                          READY   STATUS    RESTARTS   AGE    IP               NODE                      NOMINATED NODE   READINESS GATES
web-server-55f57c89d4-8lnnb   0/1     Pending   0          2m48s  <none>           <none>                    <none>           <none>
web-server-55f57c89d4-kh5st   1/1     Running   0          2m48s  172.16.232.203   worker-node-2.example.com <none>           <none>
web-server-55f57c89d4-rbxrf   0/1     Pending   0          2m48s  <none>           <none>                    <none>           <none>
labsuser@master:~$
```

By following these steps, you have successfully configured pod affinity and anti-affinity in Kubernetes. This ensures that your redis-cache instances are spread across different hosts and your web-server instances are colocated with the redis-cache in the same nodes for optimal performance and resilience.