Kubernetes : Node Affinity in Pod Scheduling

Objective:

Learn how to use **Node Affinity rules** to control which nodes a Pod can be scheduled on based on node labels. This exercise uses both matching and presence-based affinity.

Prerequisites:

- A running Kubernetes cluster (e.g., KIND, minikube, kOps)
- `kubectl` installed and configured
- Access to at least 2 worker nodes (`worker01`, `worker02`)

Task 1: Create a Pod with Node Affinity for `disktype=ssd`

1. Create a file `nginx-affinity.yaml`:

2. Apply the pod manifest:

kubectl apply -f nginx-affinity.yaml

3. Check the pod status:

```
apiVersion: v1
    kind: Pod
    metadata:
      name: nginx-affinity
    spec:
      containers:
      - name: nginx
        image: nginx
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
```

```
            nodeSelectorTerms:
            - matchExpressions:
              - key: disktype
                operator: In
                values:
                - ssd
    ```
```

kubectl get pods -o wide

kubectl describe pod nginx-affinity

```
controlplane ~ ➜ sudo vi nginx-affinity.yaml

controlplane ~ ➜ kubectl apply -f nginx-affinity.yaml
pod/nginx-affinity created

controlplane ~ ➜ kubectl get pods -o wide
NAME   READY   STATUS   RESTARTS   AGE   IP   NODE   NOMINATED NODE   READIN
ESS GATES
nginx-affinity   0/1   Pending   0   9s   <none>   <none>   <none>   <none>
```
```

```
controlplane ~ ➜ cat nginx-affinity.yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-affinity
  labels:
    app: nginx-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: disktype
            operator: In
            values:
            - ssd
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

```
controlplane ~ ➜ kubectl describe pod nginx-affinity
Name:             nginx-affinity
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           app=nginx-affinity
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  nginx:
    Image:        nginx
    Port:         80/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-k9z4q (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-k9z4q:
    Type:                    Projected (a volume that contains injected data from multiple so
urces)
```


Expected:** Pod should remain in `Pending` state because no node is labeled with disktype=ssd`.

Task 2: Label Node and Trigger Scheduling:

1. Label the node `worker01` with the required label:

```
controlplane ~ ✖ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   40m   v1.33.0
node01         Ready    <none>          39m   v1.33.0
node02         Ready    <none>          39m   v1.33.0

controlplane ~ ➜ kubectl edit node node01
node/node01 edited
```

kubectl label node worker01 disktype=ssd

```
apiVersion: v1
kind: Node
metadata:
  annotations:
    flannel.alpha.coreos.com/backend-data: '{"VNI":1,"VtepMAC":"de:ab:b1:c2:24:30"}'
    flannel.alpha.coreos.com/backend-type: vxlan
    flannel.alpha.coreos.com/kube-subnet-manager: "true"
    flannel.alpha.coreos.com/public-ip: 192.168.102.166
    kubeadm.alpha.kubernetes.io/cri-socket: unix:///var/run/containerd/containerd.sock
    node.alpha.kubernetes.io/ttl: "0"
    volumes.kubernetes.io/controller-managed-attach-detach: "true"
  creationTimestamp: "2025-07-28T11:56:40Z"
  labels:
    beta.kubernetes.io/arch: amd64
    beta.kubernetes.io/os: linux
    disktype: ssd
    kubernetes.io/arch: amd64
    kubernetes.io/hostname: node01
    kubernetes.io/os: linux
  name: node01
  resourceVersion: "4725"
  uid: db91e458-8d87-41de-acc8-017215ecf1e2
"/tmp/kubectl-edit-2374353021.yaml" 183L, 6987B
```

2. Recheck the status of the pod:

kubectl get pods -o wide

Expected:** The pod `nginx-affinity` should now be scheduled on `worker01`.

```
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-7btsd:
    Type:                     Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:   3607
    ConfigMapName:            kube-root-ca.crt
    Optional:                 false
    DownwardAPI:              true
QoS Class:                    BestEffort
Node-Selectors:               <none>
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  2m16s  default-scheduler  Successfully assigned default/nginx-affinity to node01
  Normal  Pulling    2m15s  kubelet            Pulling image "nginx"
  Normal  Pulled     2m15s  kubelet            Successfully pulled image "nginx" in 161ms (161ms including waiting). Image size: 72223946 bytes.
  Normal  Created    2m15s  kubelet            Created container: nginx
  Normal  Started    2m15s  kubelet            Started container nginx
```

Task 3: Create Pod with Presence-Based Node Affinity

1. Create a file `redis-affinity.yaml`:

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: redis-affinity
spec:
  containers:
  - name: redis
    image: redis
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: disktype
            operator: Exists
```

2. Apply the pod manifest:

```
kubectl apply -f redis-affinity.yaml
```

3. Check the pod status:

kubectl get pods -o wide
kubectl describe pod redis-affinity

Expected: Pod will not schedule yet if no node has the `disktype` label.

Task 4: Label Node with Empty Value

1. Remove any existing `disktype` label from `worker02` (if any):

kubectl label node worker02 disktype-

2. Add `disktype` label with an empty value on `worker02`:

```
kubectl label node worker02 disktype=""
```

3. Verify node labels:

```
kubectl get node worker02 --show-labels
```

4. Recreate `redis-affinity` pod:

```
kubectl delete pod redis-affinity
kubectl apply -f redis-affinity.yaml
```

5. Check pod scheduling:

```
kubectl get pods -o wide
```

Expected:** Pod `redis-affinity` should be scheduled on `worker02`, since only the key is checked.

Task 5: Cleanup (Optional):

1. Delete the created pods:

```
kubectl delete pod nginx-affinity redis-affinity
```

2. Remove labels (optional):

```
kubectl label node worker01 disktype-
kubectl label node worker02 disktype-
```

```
controlplane ~ ✖ kubectl delete pod redis-affinity
kubectl apply -f redis-affinity-pod.yaml
pod "redis-affinity" deleted
pod/redis-affinity created

controlplane ~ ➜  kubectl get pods -o wide
NAME             READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
nginx-affinity   1/1     Running   0          17m   172.17.1.3   node01   <none>           <none>
redis-affinity   0/1     Pending   0          8s    <none>       <none>   <none>           <none>
```

```
controlplane ~ ✖ kubectl delete pod nginx-affinity redis-affinity
pod "nginx-affinity" deleted
pod "redis-affinity" deleted
```