Constraint Pod and Node Selector Node Affinity & Anti Affinity

Introduction

Assigning Pods to Nodes

You can constrain a Pod to only be able to run on particular Node(s), or to prefer to run on particular nodes.

nodeSelector is the simplest recommended form of node selection constraint. nodeSelector is a field of PodSpec. It specifies a map of key-value pairs. For the pod to be eligible to run on a node, the node must have each of the indicated key-value pairs as labels (it can have additional labels as well).

Node affinity

Node affinity is conceptually similar to nodeSelector -- it allows you to constrain which nodes your pod is eligible to be scheduled on, based on labels on the node.

Pod affinity and anti-affinity

Inter-pod affinity and anti-affinity allow you to constrain which nodes your pod is eligible to be scheduled based on labels on pods that are already running on the node rather than based on labels on nodes.

This guide Covers:

- Constraining pods with node selector
- Constraining pods with node affinity
- Constraining pods with node anti-affinity

1 CONSTRAINING PODS WITH NODE SELECTOR

1.1 Adding Label to the nodes in the cluster

1. Check for the default labels of all the nodes. We would use one of the worker node and label going further

```
$ kubectl get nodes --show-labels
```

```
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu# kubectl get nodes --show-labels
NAME             STATUS   ROLES    AGE     VERSION   LABELS
kubeadm-master   Ready    master   5d11h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=
kubeadm-master,kubernetes.io/os=linux,node-role.kubernetes.io/master=
worker1          Ready    <none>   5d11h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=
worker1,kubernetes.io/os=linux
worker2          Ready    <none>   5d11h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=
worker2,kubernetes.io/os=linux
root@kubeadm-master:/home/ubuntu#
```

2. Add label to the worker1 node

```
$ kubectl label nodes worker1 disktype=ssd
```

```
root@kubeadm-master:/home/ubuntu#
root@kubeadm-master:/home/ubuntu# kubectl label nodes worker1 disktype=ssd
node/worker1 labeled
root@kubeadm-master:/home/ubuntu#
```

3. View the labels of the nodes again to verify labelling was done as expected

```
$ kubectl get nodes --show-labels
```

```
root@kubeadm-master:/home/ubuntu# kubectl get nodes --show-labels
NAME             STATUS   ROLES    AGE     VERSION   LABELS
kubeadm-master   Ready    master   5d11h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=
kubeadm-master,kubernetes.io/os=linux,node-role.kubernetes.io/master=
worker1          Ready    <none>   5d11h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disktype=ssd,kubernetes.io/arch=amd64,kubernetes
.io/hostname=worker1,kubernetes.io/os=linux
worker2          Ready    <none>   5d11h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=
worker2,kubernetes.io/os=linux
root@kubeadm-master:/home/ubuntu#
```

1.2 Create Deployment With Node Constraints

1. Create deployment with 2 replicas and specify the constraint with **nodeSelector** label of disktype: ssd
2. Check the content of label-deployment.yaml file

```
$ vim label-deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      nodeSelector:
        disktype: ssd
~
~
~
~
```

3. Create deployment with kubectl create command

```
$ kubectl create -f label-deployment.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl create -f label-deployment.yaml
deployment.apps/nginx-deployment created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

4. Grep the node details for ssd labelled node

```
$ kubectl get nodes --show-labels  | grep ssd
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get nodes --show-labels | grep ssd
worker1         Ready    <none>   5d12h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disktype=ssd,kubernetes.io/arch=amd64,kubernetes
.io/hostname=worker1,kubernetes.io/os=linux
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

5. Verify that both the replicas of the nginx-deployment is scheduled on worker node “worker1” which was labelled as disktype: ssd

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

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-5cdb5745d-c7c7p   1/1     Running   0          68s   10.46.0.3   worker1   <none>           <none>
nginx-deployment-5cdb5745d-pd4md   1/1     Running   0          68s   10.46.0.2   worker1   <none>           <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

1.3 Clean-Up Resources & Labels

1. Delete the deployment using kubectl delete command with filename

```
$ kubectl delete -f label-deployment.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete -f label-deployment.yaml
deployment.apps "nginx-deployment" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -o wide
NAME                                READY   STATUS        RESTARTS   AGE    IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-5cdb5745d-c7c7p    0/1     Terminating   0          2m3s   10.46.0.3   worker1   <none>           <none>
nginx-deployment-5cdb5745d-pd4md    1/1     Terminating   0          2m3s   10.46.0.2   worker1   <none>           <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Remove the label added to worker1 node with kubectl label command and verify the label is removed

```
$ kubectl label nodes worker1 disktype-
$ kubectl get nodes --show-labels  | grep ssd
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl label nodes worker1 disktype-
node/worker1 labeled
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get nodes --show-labels | grep ssd
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2 CONSTRAINING PODS WITH NODE AFFINITY

2.1 Create Deployment With Node Affinity Constraint

1. Create deployment with 2 replicas and specify the constraint with node affinity constraint defined
2. Check the content of nodeaffinity-deployment.yaml file and see the constraint is **"preferredDuringSchedulingIgnoredDuringExecution"**

```
$ vim nodeaffinity-deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 1
            preference:
              matchExpressions:
              - key: disktype
                operator: In
                values:
                - ssd
~
~
~
~
~
~
~
~
~
~
"nodeaffinity-deployment.yaml" 31L, 619C
```

3. Create deployment with kubectl create command and verify

```
$ kubectl create -f nodeaffinity-deployment.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl create -f nodeaffinity-deployment.yaml
deployment.apps/nginx-deployment created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
$ kubectl get deployment
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           12s
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

4. List the pods and notice that it has been created despite none of the nodes had the specified label

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

```
root@kubeadm-master:/home/ubuntu/Kubernetes#  kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-b956d8fb7-6289t    1/1     Running   0          108s   10.46.0.2   worker1   <none>           <none>
nginx-deployment-b956d8fb7-dvll8    1/1     Running   0          108s   10.40.0.2   worker2   <none>           <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2.2 Delete Deployment

```
$ kubectl delete -f nodeaffinity-deployment.yaml
```

1. Define **"requiredDuringSchedulingIgnoredDuringExecution" Constraint**

   a. Check the content of nodeaffinity1-deployment.yaml file and see the constraint is "requiredDuringSchedulingIgnoredDuringExecution"

```
$ vim nodeaffinity1-deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: disktype
                operator: In
                values:
                - ssd
~
~
~
~
```

   b. Create deployment with kubectl create command. List the deployment and pods, see that the pods are in **pending state** as none of the nodes have the required label

```
$ kubectl create -f nodeaffinity1-deployment.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl create -f nodeaffinity1-deployment.yaml
deployment.apps/nginx-deployment created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
$ kubectl get deployment
$ kubectl get pods -o wide
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   0/2     2            0           12s
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NODE   READINESS GATES
nginx-deployment-6d46875998-48649   0/1     Pending   0          25s   <none>   <none>   <none>           <none>
nginx-deployment-6d46875998-7j8ks   0/1     Pending   0          25s   <none>   <none>   <none>           <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2.3 Verify pod scheduling

1. Label node “worker2” and notice that pending pods get scheduled on the labelled node
2. Label worker node “worker2” disktype=ssd

```
$ kubectl label nodes worker2 disktype=ssd
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl label nodes worker2 disktype=ssd
node/worker2 labeled
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3. List the deployment and pods to verify pods get scheduled on worker2 node

```
$ kubectl get deployment


$ kubectl get pods -o wide
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           68s
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-6d46875998-48649   1/1     Running   0          71s   10.40.0.2   worker2   <none>           <none>
nginx-deployment-6d46875998-7j8ks   1/1     Running   0          71s   10.40.0.3   worker2   <none>           <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2.4 Clean-Up Resources & Label added in this Exercise

1. Delete the deployment

```
$ kubectl delete -f nodeaffinity1-deployment.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete -f nodeaffinity1-deployment.yaml
deployment.apps "nginx-deployment" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Remove the label added to worker2 node with kubectl label command and verify the label is removed

```
$ kubectl label nodes worker2 disktype-
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl label nodes worker2 disktype-
node/worker2 labeled
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3 CONSTRAINING PODS WITH NODE ANTI-AFFINITY

3.1 Label nodes “worker2” and “worker1”

1. Label worker node “worker1” and “worker2” disktype=ssd

```
$ kubectl label nodes worker1 disktype=ssd
$ kubectl label nodes worker2 disktype=ssd
```

```
root@kubeadm-master:~/Kubernetes#
root@kubeadm-master:~/Kubernetes# kubectl label nodes worker1 disktype=ssd
node/worker1 labeled
root@kubeadm-master:~/Kubernetes#
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl label nodes worker2 disktype=ssd
node/worker2 labeled
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3.2 Create Deployment with Node Anti-Affinity preferred constraint

1. Create deployment with 2 replicas and specify the constraint with node anti-affinity constraint defined
2. Check the content of nodeantiaffinity-deployment.yaml file and see the constraint is **“preferredDuringSchedulingIgnoredDuringExecution”**

```
$ vim nodeanti-affinity-deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 1
            preference:
              matchExpressions:
              - key: disktype
                operator: NotIn
                values:
                - ssd
~
~
~
~
```

3. Create deployment with kubectl create command and verify that inspite of the node label pod gets placed as the condition is preferred one

```
$ kubectl create -f nodeanti-affinity-deployment.yaml
```

```
root@kubeadm-master:~/Kubernetes# kubectl create -f nodeanti-affinity-deployment.yaml
deployment.apps/nginx-deployment created
root@kubeadm-master:~/Kubernetes#
```

```
$ kubectl get deployment
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           12s
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

4. List the pods and notice that it has been created despite none of the nodes had the specified label

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

```
root@kubeadm-master:/home/ubuntu/Kubernetes#  kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-b956d8fb7-6289t   1/1     Running   0          108s   10.46.0.2   worker1   <none>           <none>
nginx-deployment-b956d8fb7-dvll8   1/1     Running   0          108s   10.40.0.2   worker2   <none>           <none>
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

3.3 Delete the deployment

```
$ kubectl delete -f nodeanti-affinity-deployment.yaml
```

3.4 Creating Pod with Node Anti-Affinity required constraint

1. Check the content of nodeaffinity1-deployment.yaml file and see the constraint is “requiredDuringSchedulingIgnoredDuringExecution”

```
$ vim nodeanti-affinity1-deployment.yaml
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.12
        ports:
        - containerPort: 80
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: disktype
                operator: NotIn
                values:
                - ssd
~
~
~
```

2. Create deployment with kubectl create command. List the deployment and pods, see that the pods are in pending state as all the nodes in the cluster have the label

```
$ kubectl create -f nodeanti-affinity1-deployment.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl create -f nodeaffinity1-deployment.yaml
deployment.apps/nginx-deployment created
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

```
$ kubectl get deployment
$ kubectl get pods -o wide
```

```
root@kubeadm-master:~/Kubernetes# kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   0/2     2            0           101s
root@kubeadm-master:~/Kubernetes# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP       NODE     NOMINATED NODE   READINESS GATES
nginx-deployment-6ff667f855-ddhxm   0/1     Pending   0          103s   <none>   <none>   <none>           <none>
nginx-deployment-6ff667f855-mjj4v   0/1     Pending   0          103s   <none>   <none>   <none>           <none>
root@kubeadm-master:~/Kubernetes#
```

3.5 Verify Pod Scheduling

1. Remove label from node “worker2” and notice that pending pods get scheduled on it, as its not labelled
2. Remove label from worker node “worker2” disktype-

```
$ kubectl label nodes worker2 disktype-
```

3. List the deployment and pods to verify pods get scheduled on worker2 node

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

```
root@kubeadm-master:~/Kubernetes# kubectl label nodes worker2 disktype-
node/worker2 labeled
root@kubeadm-master:~/Kubernetes# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-6ff667f855-ddhxm   1/1     Running   0          3m2s   10.38.0.3   worker2   <none>           <none>
nginx-deployment-6ff667f855-mjj4v   1/1     Running   0          3m2s   10.38.0.2   worker2   <none>           <none>
root@kubeadm-master:~/Kubernetes#
```

3.6 Clean-Up Resources & Label added in this Exercise

1. Delete the deployment

```
$ kubectl delete -f nodeanti-affinity1-deployment.yaml
```

```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl delete -f nodeaffinity1-deployment.yaml
deployment.apps "nginx-deployment" deleted
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

2. Remove the label added to worker1 node with kubectl label command and verify the label is removed

```
$ kubectl label nodes worker1 disktype-
```