Node Mgmt

12 January 2021 11:43

Node Drain

- Can purposefully drain the node of all the workloads so that the workloads are moved to other nodes.
    - The node is also cordoned or marked as **unschedulable**.

```
$ kubectl drain worker1
```

- There is also another command called cordon. Cordon simply marks a node unschedulable. Unlike drain it does not terminate or move the pods on an existing node.

```
$ kubectl cordon worker1
```

=====================================

- When the node is back online after a maintenance, it is still unschedulable. You then need to uncordon it.

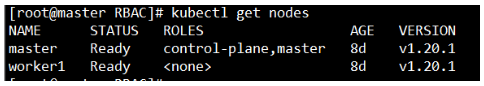
$ kubectl uncordon **worker-1**

2 worker nodes
1) Deploy 6 replicas (assumption is that we will have 3 replicas on each worker node)
2) $ kubectl drain worker1
    a. Checking the results

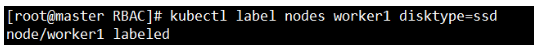
$ kubectl cordon worker1

Node Selector:

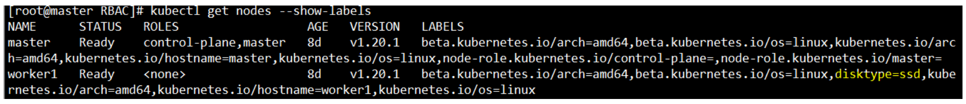
Firstly apply label to Node



```
[root@master RBAC]# kubectl get nodes
NAME      STATUS   ROLES                  AGE   VERSION
master    Ready    control-plane,master   8d    v1.20.1
worker1   Ready    <none>                 8d    v1.20.1
```

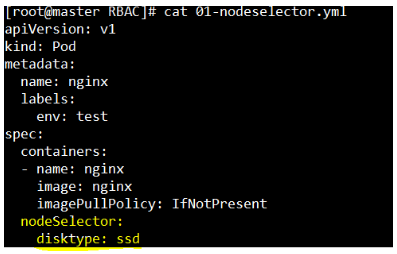


```
[root@master RBAC]# kubectl label nodes worker1 disktype=ssd
node/worker1 labeled
```

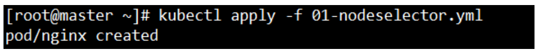


```
[root@master RBAC]# kubectl get nodes --show-labels
NAME      STATUS   ROLES                  AGE   VERSION   LABELS
master    Ready    control-plane,master   8d    v1.20.1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arc
h=amd64,kubernetes.io/hostname=master,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=
worker1   Ready    <none>                 8d    v1.20.1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disktype=ssd,kube
rnetes.io/arch=amd64,kubernetes.io/hostname=worker1,kubernetes.io/os=linux
```

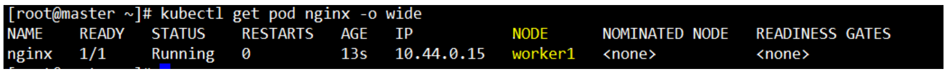
Secondly define Node selector in pod config fie



```
[root@master RBAC]# cat 01-nodeselector.yml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
```



```
[root@master ~]# kubectl apply -f 01-nodeselector.yml
pod/nginx created
```



```
[root@master ~]# kubectl get pod nginx -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP           NODE      NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          13s   10.44.0.15   worker1   <none>           <none>
```

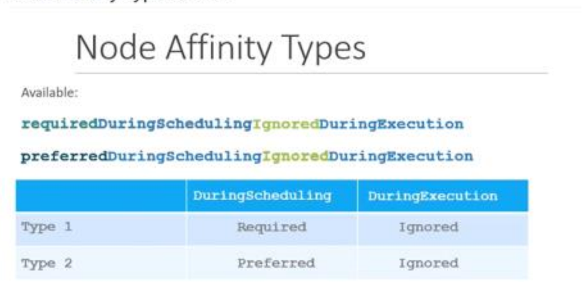
Limitation
- **Cannot provide advance expressions**
- **We used a single label and selector to achieve our goal here. But what if our requirement is much more complex.**
    - **e.g. disktype not ssd or size like large, medium or not small etc.**
- **For this we have Node Affinity and Anti affinity**

Node Affinity

- Conceptually similar to nodeSelector
    - Allows you to constrain which nodes your pod is eligible to be scheduled on, based on labels on the node

Two Types :
- **requiredDuringSchedulingIgnoredDuringExecution**
- **preferredDuringSchedulingIgnoredDuringExecution**

Node Affinity Types States


Node Affinity Types

Available:

requiredDuringSchedulingIgnoredDuringExecution

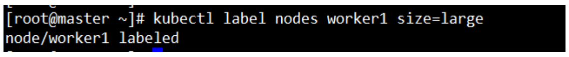
preferredDuringSchedulingIgnoredDuringExecution

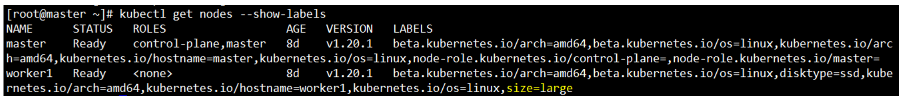
	DuringScheduling	DuringExecution
Type 1	Required	Ignored
Type 2	Preferred	Ignored

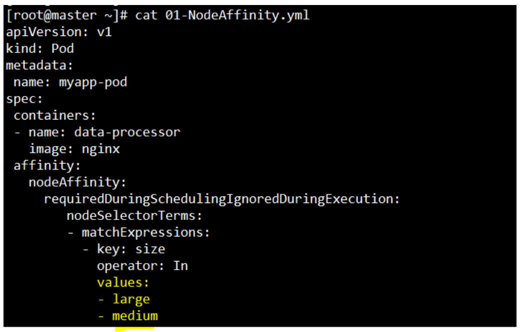
-- Labeled node as "**size: large**"

-- Pod operators:

- Large
- medium

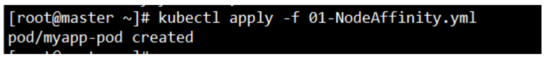

```
[root@master ~]# kubectl label nodes worker1 size=large
node/worker1 labeled
```


```
[root@master ~]# kubectl get nodes --show-labels
NAME      STATUS   ROLES                  AGE   VERSION   LABELS
master    Ready    control-plane,master   8d    v1.20.1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arc
h=amd64,kubernetes.io/hostname=master,kubernetes.io/os=linux,node-role.kubernetes.io/control-plane=,node-role.kubernetes.io/master=
worker1   Ready    <none>                 8d    v1.20.1   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disktype=ssd,kube
rnetes.io/arch=amd64,kubernetes.io/hostname=worker1,kubernetes.io/os=linux,size=large
```

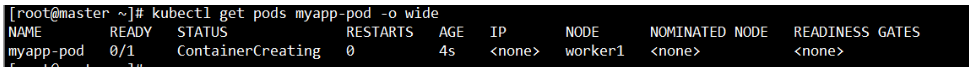

```
[root@master ~]# cat 01-NodeAffinity.yml
apiVersion: v1
kind: Pod
metadata:
 name: myapp-pod
spec:
 containers:
 - name: data-processor
   image: nginx
 affinity:
   nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: size
            operator: In
            values:
            - large
            - medium
```

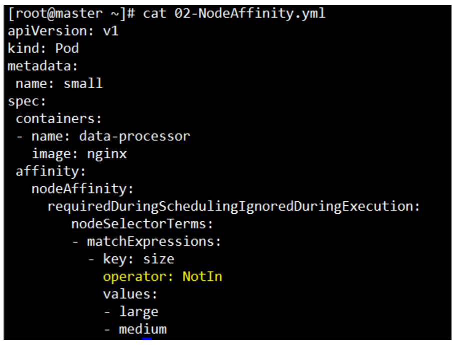
.

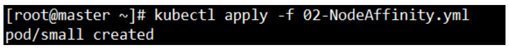
kubectl apply -f 01-NodeAffinity.yml

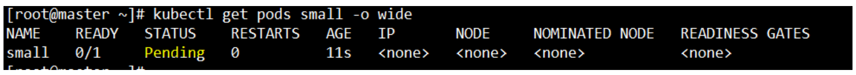

```
[root@master ~]# kubectl apply -f 01-NodeAffinity.yml
pod/myapp-pod created
```

kubectl get pods myapp-pod -o wide


```
[root@master ~]# kubectl get pods myapp-pod -o wide
NAME        READY   STATUS              RESTARTS   AGE   IP       NODE      NOMINATED NODE   READINESS GATES
myapp-pod   0/1     ContainerCreating   0          4s    <none>   worker1   <none>           <none>
```


```
[root@master ~]# cat 02-NodeAffinity.yml
apiVersion: v1
kind: Pod
metadata:
 name: small
spec:
 containers:
 - name: data-processor
   image: nginx
 affinity:
   nodeAffinity:
     requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: size
            operator: NotIn
            values:
            - large
            - medium
```


```
[root@master ~]# kubectl apply -f 02-NodeAffinity.yml
pod/small created
```


```
[root@master ~]# kubectl get pods small -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NODE   READINESS GATES
small   0/1     Pending   0          11s   <none>   <none>   <none>           <none>
```

Taints (node) and Toleration (pod)

- Pod to node relationship and how you can restrict what pods are placed on what nodes.
- Taints and Tolerations are used to set restrictions on what pods can be scheduled on a node.
- Only pods which are tolerant to the particular taint on a node will get scheduled on that node.

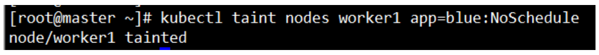
3 Taint Effects

- **NoSchedule** - Kubernetes will not schedule the pod onto that node
- **PreferNoSchedule** - Kubernetes will try to not schedule the pod onto the node
- **NoExecute**: pods that do not tolerate the taint will be evicted immediately

- Key:
- Value:
- Effect: explained above (behavior)
- Operation: condition

Apply Taint

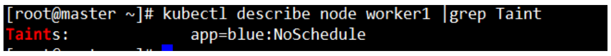
# kubectl taint nodes worker1 app=blue:NoSchedule



```
[root@master ~]# kubectl taint nodes worker1 app=blue:NoSchedule
node/worker1 tainted
```

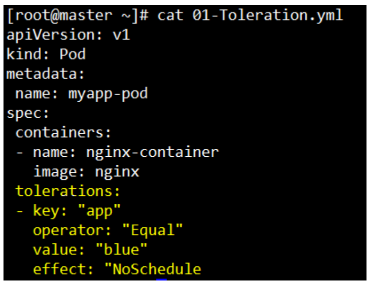
Check Taint:

# kubectl describe node worker1 |grep Taint

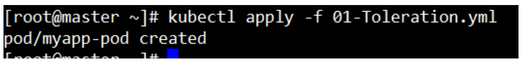


```
[root@master ~]# kubectl describe node worker1 |grep Taint
Taints:             app=blue:NoSchedule
```

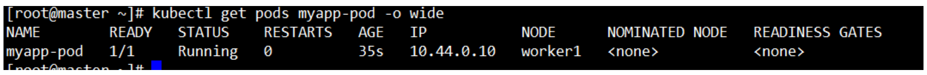
Tolerations are added to pods by adding a tolerations **section in pod definition.**



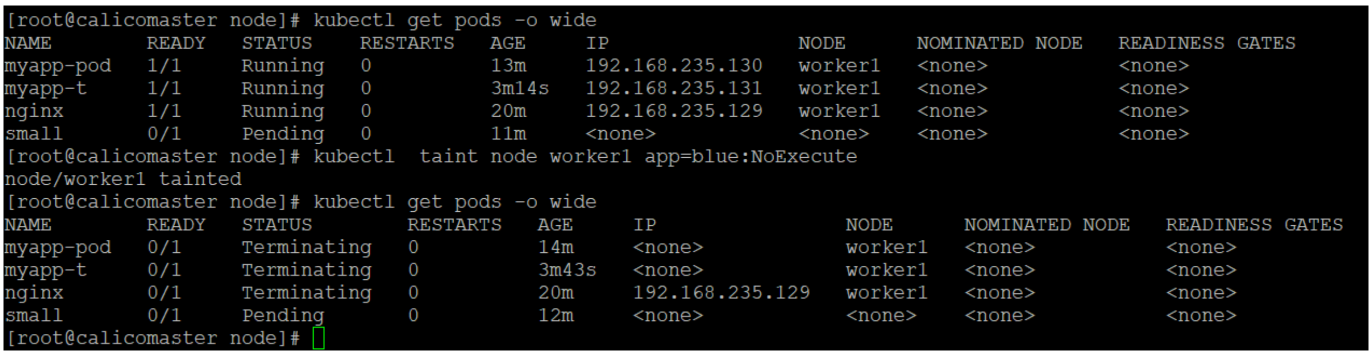
```
[root@master ~]# cat 01-Toleration.yml
apiVersion: v1
kind: Pod
metadata:
 name: myapp-pod
spec:
 containers:
 - name: nginx-container
   image: nginx
 tolerations:
 - key: "app"
   operator: "Equal"
   value: "blue"
   effect: "NoSchedule
```



```
[root@master ~]# kubectl apply -f 01-Toleration.yml
pod/myapp-pod created
```



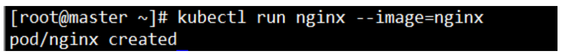
```
[root@master ~]# kubectl get pods myapp-pod -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP           NODE      NOMINATED NODE   READINESS GATES
myapp-pod   1/1     Running   0          35s   10.44.0.10   worker1   <none>           <none>
```



```
[root@calicomaster node]# kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE     IP                NODE      NOMINATED NODE   READINESS GATES
myapp-pod   1/1     Running   0          13m     192.168.235.130   worker1   <none>           <none>
myapp-t     1/1     Running   0          3m14s   192.168.235.131   worker1   <none>           <none>
nginx       1/1     Running   0          20m     192.168.235.129   worker1   <none>           <none>
small       0/1     Pending   0          11m     <none>            <none>    <none>           <none>
[root@calicomaster node]# kubectl  taint node worker1 app=blue:NoExecute
node/worker1 tainted
[root@calicomaster node]# kubectl get pods -o wide
NAME        READY   STATUS        RESTARTS   AGE     IP                NODE      NOMINATED NODE   READINESS GATES
myapp-pod   0/1     Terminating   0          14m     <none>            worker1   <none>           <none>
myapp-t     0/1     Terminating   0          3m43s   <none>            worker1   <none>           <none>
nginx       0/1     Terminating   0          20m     192.168.235.129   worker1   <none>           <none>
small       0/1     Pending       0          12m     <none>            <none>    <none>           <none>
[root@calicomaster node]#
```

Now create pod without toleration,

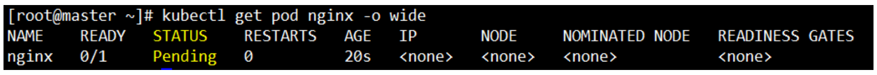
# kubectl run nginx --image=nginx



```
[root@master ~]# kubectl run nginx --image=nginx
pod/nginx created
```

Not assigned to worker node and taint is applied on it.

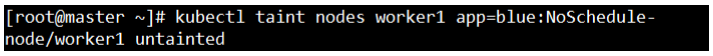
# kubectl get pod nginx -o wide



```
[root@master ~]# kubectl get pod nginx -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NODE   READINESS GATES
nginx   0/1     Pending   0          20s   <none>   <none>   <none>           <none>
```

Now we are removing taint from node. After removing nginx pod will be assigned to worker node

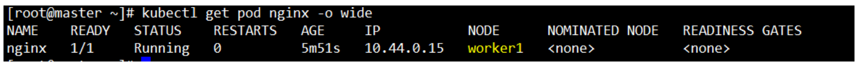
# kubectl taint nodes worker1 app=blue:NoSchedule-



```
[root@master ~]# kubectl taint nodes worker1 app=blue:NoSchedule-
node/worker1 untainted
```

Now Pod is assigned to worker node

# kubectl get pod nginx -o wide



```
[root@master ~]# kubectl get pod nginx -o wide
NAME    READY   STATUS    RESTARTS   AGE     IP           NODE      NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          5m51s   10.44.0.15   worker1   <none>           <none>
```