Pod Scheduling (Node Selector Node Affinity, Anti-Affinity, Taint & Toleration)

[Edition 3]

[Last Update 211018]

For any issues/help contact : ***support@k21academy.com***

Contents

1 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.

Taint and Tolerations

Node affinity, is a property of Pods that attracts them to a set of nodes (either as a preference or a hard requirement). Taints are the opposite -- they allow a node to repel a set of pods.

Tolerations are applied to pods, and allow (but do not require) the pods to schedule onto nodes with matching taints.

Taints and tolerations work together to ensure that pods are not scheduled onto inappropriate nodes. One or more taints are applied to a node; this marks that the node should not accept any pods that do not tolerate the taints.

This guide Covers:

- Constraining pods with node selector
- Constraining pods with node affinity
- Constraining pods with node anti-affinity
- Constraining pods with taint and toleration

2 DOCUMENTATION

2.1 Kubernetes Documentation

1. DaemonSet Controller
   https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/
2. Assign Pods to Nodes NodeSelector
   https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/
   https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/
3. Assign Pods to Nodes using Node Affinity
   https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes-using-node-affinity/
4. Advanced Scheduling in Kubernetes
   https://kubernetes.io/blog/2017/03/advanced-scheduling-in-kubernetes/#:~:text=Node%20affinity%2Fanti%2Daffinity%20allows,to%20pods%20in%20other%20services%3F
5. Taints and Tolerations
   https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/

2.2 Linux Commands and VIM Commands

Note: If you are new to Linux and wanted to learn basic linux for Kubernetes, then drop us a mail at support@k21academy.com and get Bonus Linux for Beginners course free.

1. Basic Linux Commands
   https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners
   https://www.hostinger.in/tutorials/linux-commands
2. Basic VIM Commands
   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started
3. Popular VIM Commands
   https://www.keycdn.com/blog/vim-commands

3 PRE-REQUISITE

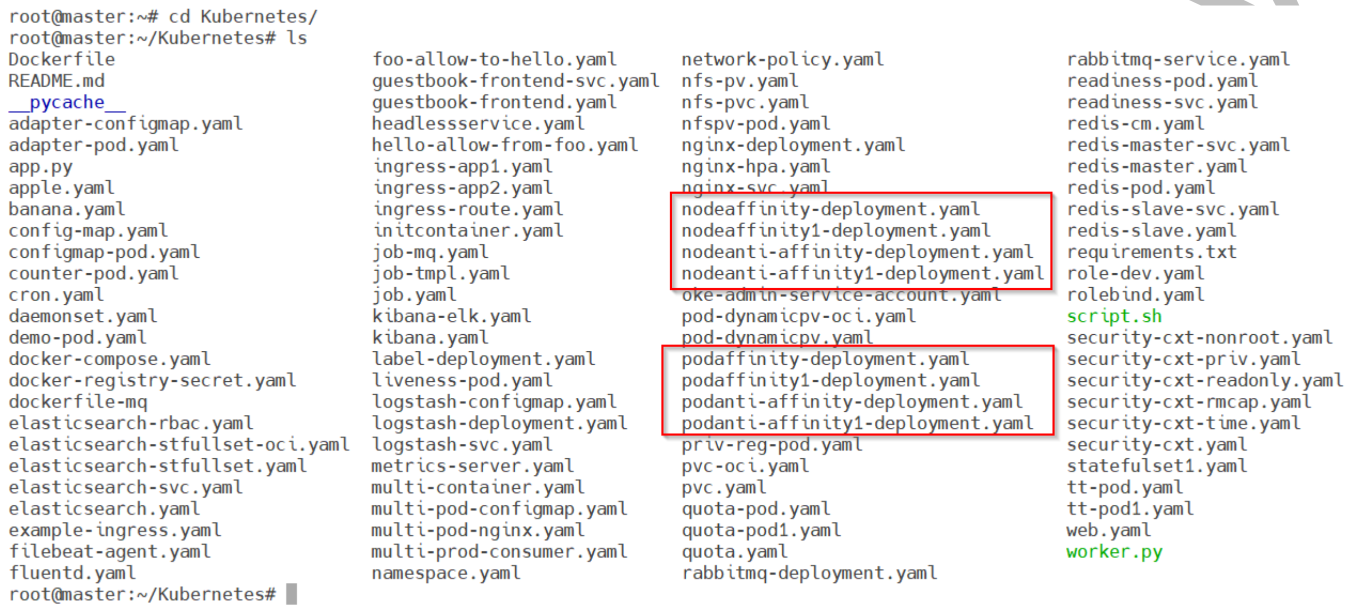
Ensure that you have completed following three activity guides (or you have an Ubuntu Server)

- **Note**: *Follow Activity Guide* ***AG_Bootstrap_Kubernetes_Cluster_Using_Kubeadm_Guide_ed***** *from portal*
- **Note**: *Follow Activity Guide* ***AG_ Deploy_App_On_Pod_&_Basic_Networking_ed***** *from portal*
- **Note**: *Follow Activity Guide* ***AG_Deploying_Scalable_and_Configuring_Autoscaling_For_Stateless_Application_ed***** *from portal*
- **Note**: *Follow Activity Guide* ***AG_Configuring_NFS_Storage_Persistence_Volume_ed***** *from portal*

4 CONSTRAINING PODS WITH NODE SELECTOR

Note: In below Sections we are going to use YAML files no need write complete yaml file because in CKA exam you can official Kubernetes documentation use Below GIT url to clone repo and use yaml files

```
$ git clone https://github.com/k21academyuk/Kubernetes
$ cd Kubernetes
```

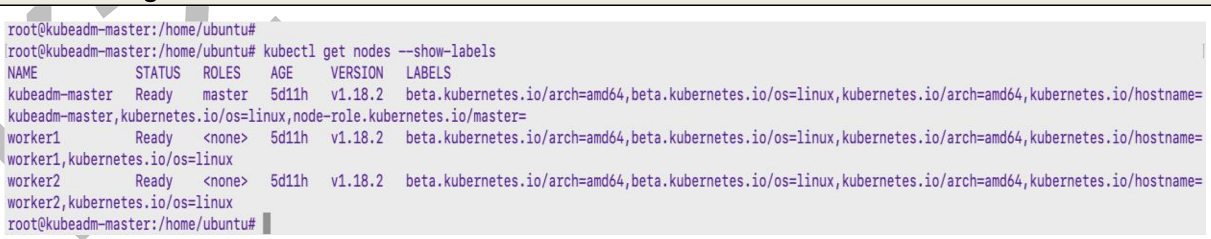


```
root@master:~# cd Kubernetes/
root@master:~/Kubernetes# ls
Dockerfile                        foo-allow-to-hello.yaml      network-policy.yaml                 rabbitmq-service.yaml
README.md                         guestbook-frontend-svc.yaml  nfs-pv.yaml                         readiness-pod.yaml
__pycache__                       guestbook-frontend.yaml      nfs-pvc.yaml                        readiness-svc.yaml
adapter-configmap.yaml            headlessservice.yaml         nfspv-pod.yaml                      redis-cm.yaml
adapter-pod.yaml                  hello-allow-from-foo.yaml    nginx-deployment.yaml               redis-master-svc.yaml
app.py                            ingress-app1.yaml            nginx-hpa.yaml                      redis-master.yaml
apple.yaml                        ingress-app2.yaml            nginx-svc.yaml                      redis-pod.yaml
banana.yaml                       ingress-route.yaml           nodeaffinity-deployment.yaml        redis-slave-svc.yaml
config-map.yaml                   initcontainer.yaml           nodeaffinity1-deployment.yaml       redis-slave.yaml
configmap-pod.yaml                job-mq.yaml                  nodeanti-affinity-deployment.yaml   requirements.txt
counter-pod.yaml                  job-tmpl.yaml                nodeanti-affinity1-deployment.yaml  role-dev.yaml
cron.yaml                         job.yaml                     oke-admin-service-account.yaml      rolebind.yaml
daemonset.yaml                    kibana-elk.yaml              pod-dynamicpv-oci.yaml              script.sh
demo-pod.yaml                     kibana.yaml                  pod-dynamicpv.yaml                  security-cxt-nonroot.yaml
docker-compose.yaml               label-deployment.yaml        podaffinity-deployment.yaml         security-cxt-priv.yaml
docker-registry-secret.yaml       liveness-pod.yaml            podaffinity1-deployment.yaml        security-cxt-readonly.yaml
dockerfile-mq                     logstash-configmap.yaml      podanti-affinity-deployment.yaml    security-cxt-rmcap.yaml
elasticsearch-rbac.yaml           logstash-deployment.yaml     podanti-affinity1-deployment.yaml   security-cxt-time.yaml
elasticsearch-stfullset-oci.yaml  logstash-svc.yaml            priv-reg-pod.yaml                   security-cxt.yaml
elasticsearch-stfullset.yaml      metrics-server.yaml          pvc-oci.yaml                        statefulset1.yaml
elasticsearch-svc.yaml            multi-container.yaml         pvc.yaml                            tt-pod.yaml
elasticsearch.yaml                multi-pod-configmap.yaml     quota-pod.yaml                      tt-pod1.yaml
example-ingress.yaml              multi-pod-nginx.yaml         quota-pod1.yaml                     web.yaml
filebeat-agent.yaml               multi-prod-consumer.yaml     quota.yaml                          worker.py
fluentd.yaml                      namespace.yaml               rabbitmq-deployment.yaml
root@master:~/Kubernetes# 
```

4.1 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

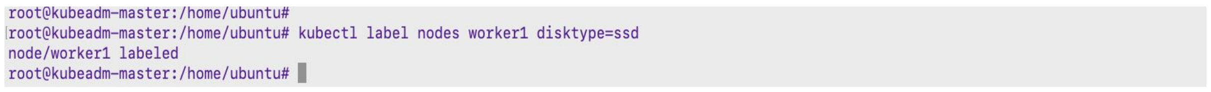
```
$ kubectl get nodes --show-labels
```



```
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

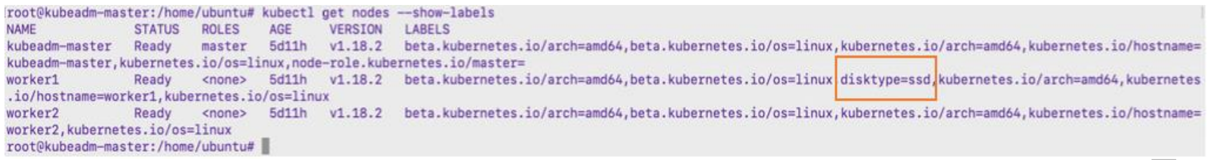
```
$ kubectl label nodes worker1 disktype=ssd
```



```
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
```

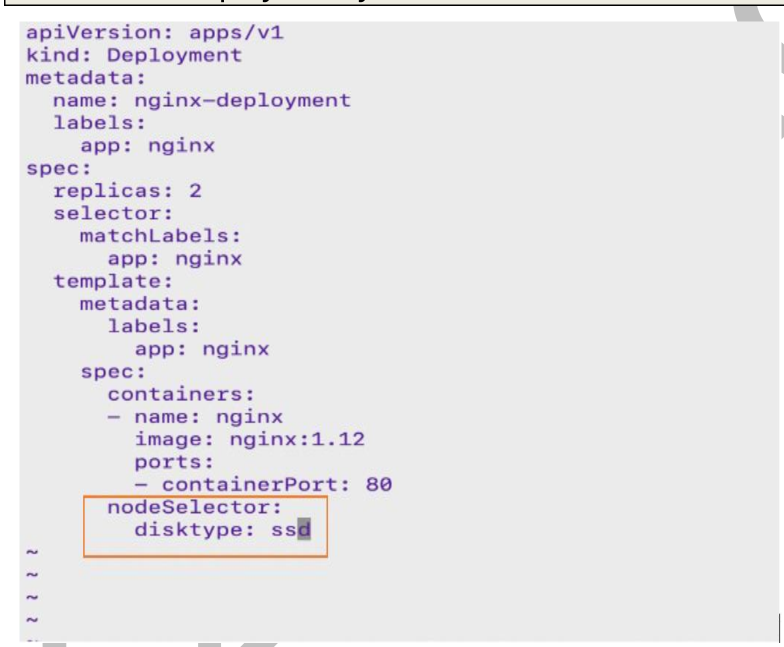


```
root@kubeadm-master:/home/ubuntu# kubectl get nodes --show-labels
NAME             STATUS   ROLES    AGE     VERSION   LABELS
kubeadm-master   Ready    master   5d11h   v1.18.2   beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=
kubeadm-master,kubernetes.io/os=linux,node-role.kubernetes.io/master=
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#
```

4.2 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

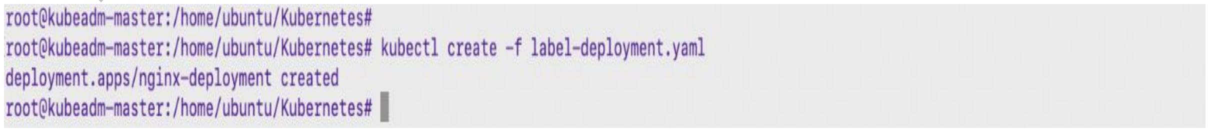
```
$ vim label-deployment.yaml
```



```
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

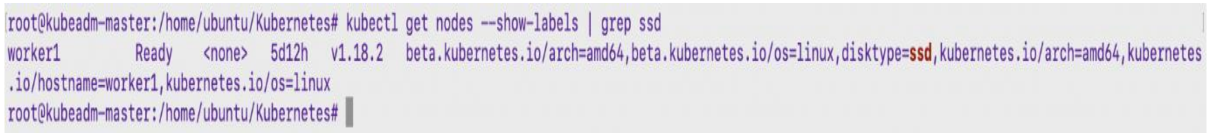
```
$ kubectl create -f label-deployment.yaml
```



```
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

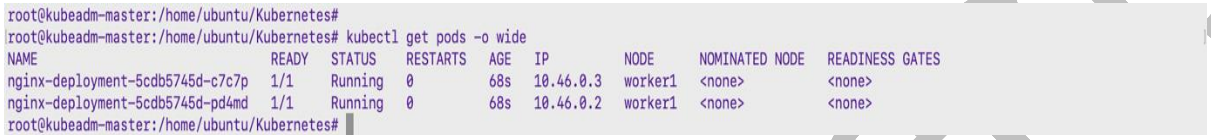
```
$ kubectl get nodes --show-labels  | grep ssd
```



```
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
```

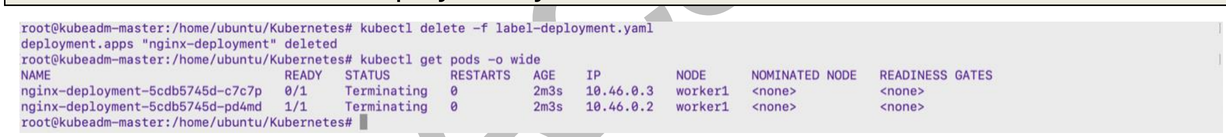


```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP           NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-5cdb5745d-c7c7p   1/1     Running   0          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#
```

4.3 Clean-Up Resources & Labels

1. Delete the deployment using kubectl delete command with filename

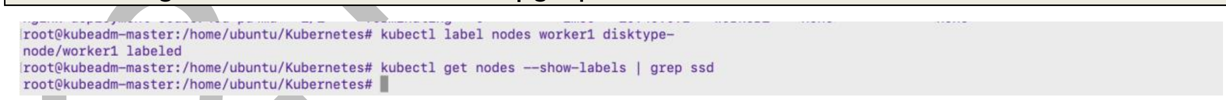
```
$ kubectl delete -f label-deployment.yaml
```



```
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
```

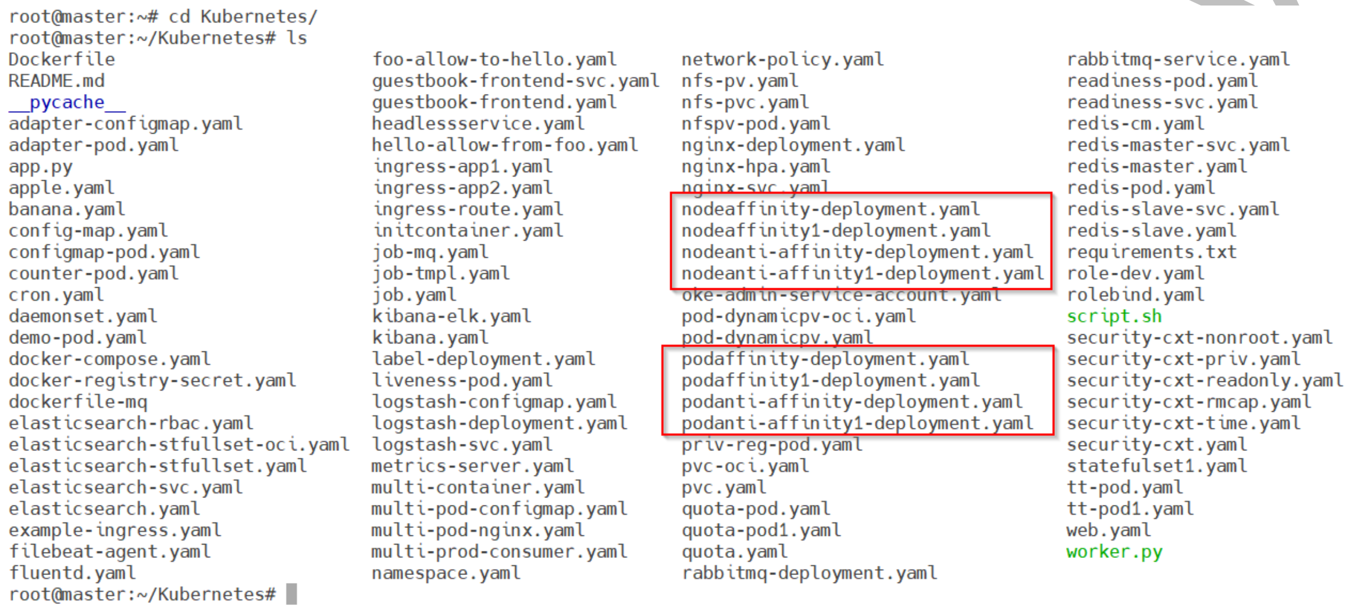


```
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl label nodes worker1 disktype-
node/worker1 labeled
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl get nodes --show-labels | grep ssd
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

5 CONSTRAINING PODS WITH NODE AFFINITY

Note: In below Sections we are going to use YAML files no need write complete yaml file because in CKA exam you can official Kubernetes documentation use Below GIT url to clone repo and use yaml files

```
$ git clone https://github.com/k21academyuk/Kubernetes
$ cd Kubernetes
```

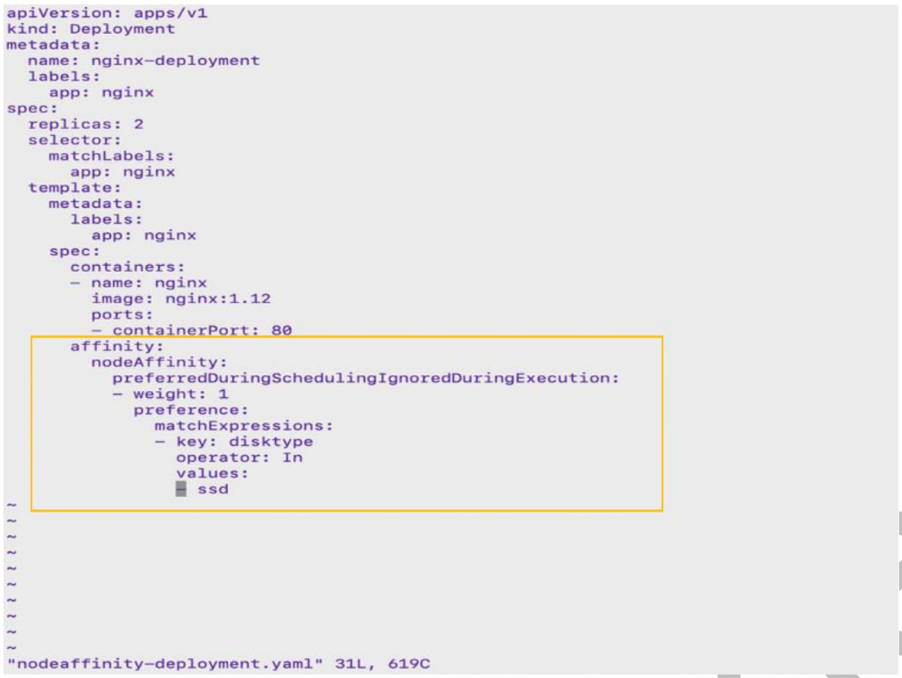


```
root@master:~# cd Kubernetes/
root@master:~/Kubernetes# ls
Dockerfile                        foo-allow-to-hello.yaml      network-policy.yaml                 rabbitmq-service.yaml
README.md                         guestbook-frontend-svc.yaml  nfs-pv.yaml                         readiness-pod.yaml
__pycache__                       guestbook-frontend.yaml      nfs-pvc.yaml                        readiness-svc.yaml
adapter-configmap.yaml            headlessservice.yaml         nfspv-pod.yaml                      redis-cm.yaml
adapter-pod.yaml                  hello-allow-from-foo.yaml    nginx-deployment.yaml               redis-master-svc.yaml
app.py                            ingress-app1.yaml            nginx-hpa.yaml                      redis-master.yaml
apple.yaml                        ingress-app2.yaml            nginx-svc.yaml                      redis-pod.yaml
banana.yaml                       ingress-route.yaml           nodeaffinity-deployment.yaml        redis-slave-svc.yaml
config-map.yaml                   initcontainer.yaml           nodeaffinity1-deployment.yaml       redis-slave.yaml
configmap-pod.yaml                job-mq.yaml                  nodeanti-affinity-deployment.yaml   requirements.txt
counter-pod.yaml                  job-tmpl.yaml                nodeanti-affinity1-deployment.yaml  role-dev.yaml
cron.yaml                         job.yaml                     oke-admin-service-account.yaml      rolebind.yaml
daemonset.yaml                    kibana-elk.yaml              pod-dynamicpv-oci.yaml              script.sh
demo-pod.yaml                     kibana.yaml                  pod-dynamicpv.yaml                  security-cxt-nonroot.yaml
docker-compose.yaml               label-deployment.yaml        podaffinity-deployment.yaml         security-cxt-priv.yaml
docker-registry-secret.yaml       liveness-pod.yaml            podaffinity1-deployment.yaml        security-cxt-readonly.yaml
dockerfile-mq                     logstash-configmap.yaml      podanti-affinity-deployment.yaml    security-cxt-rmcap.yaml
elasticsearch-rbac.yaml           logstash-deployment.yaml     podanti-affinity1-deployment.yaml   security-cxt-time.yaml
elasticsearch-stfullset-oci.yaml  logstash-svc.yaml            priv-reg-pod.yaml                   security-cxt.yaml
elasticsearch-stfullset.yaml      metrics-server.yaml          pvc-oci.yaml                        statefulset1.yaml
elasticsearch-svc.yaml            multi-container.yaml         pvc.yaml                            tt-pod.yaml
elasticsearch.yaml                multi-pod-configmap.yaml     quota-pod.yaml                      tt-pod1.yaml
example-ingress.yaml              multi-pod-nginx.yaml         quota-pod1.yaml                     web.yaml
filebeat-agent.yaml               multi-prod-consumer.yaml     quota.yaml                          worker.py
fluentd.yaml                      namespace.yaml               rabbitmq-deployment.yaml
root@master:~/Kubernetes# 
```

5.1 Create Deployment with Node Affinity preferred 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"**

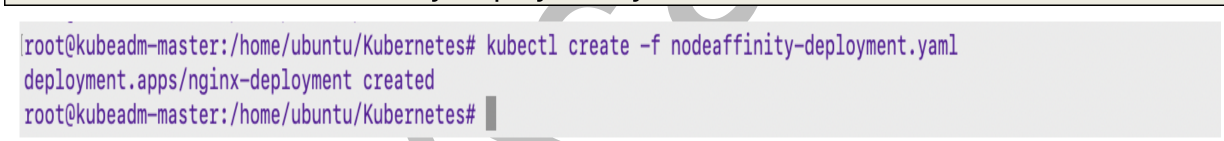
```
$ vim nodeaffinity-deployment.yaml
```



```
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
```

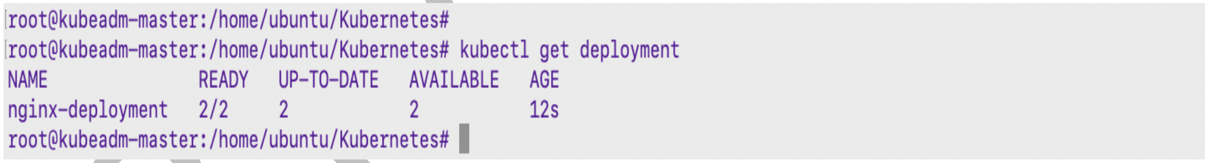
3. Create deployment with kubectl create command and verify

```
$ 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#
```

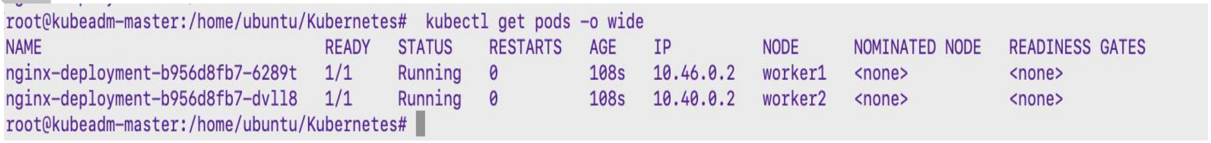
```
$ kubectl get deployment
```



```
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#
```

5. Delete Deployment

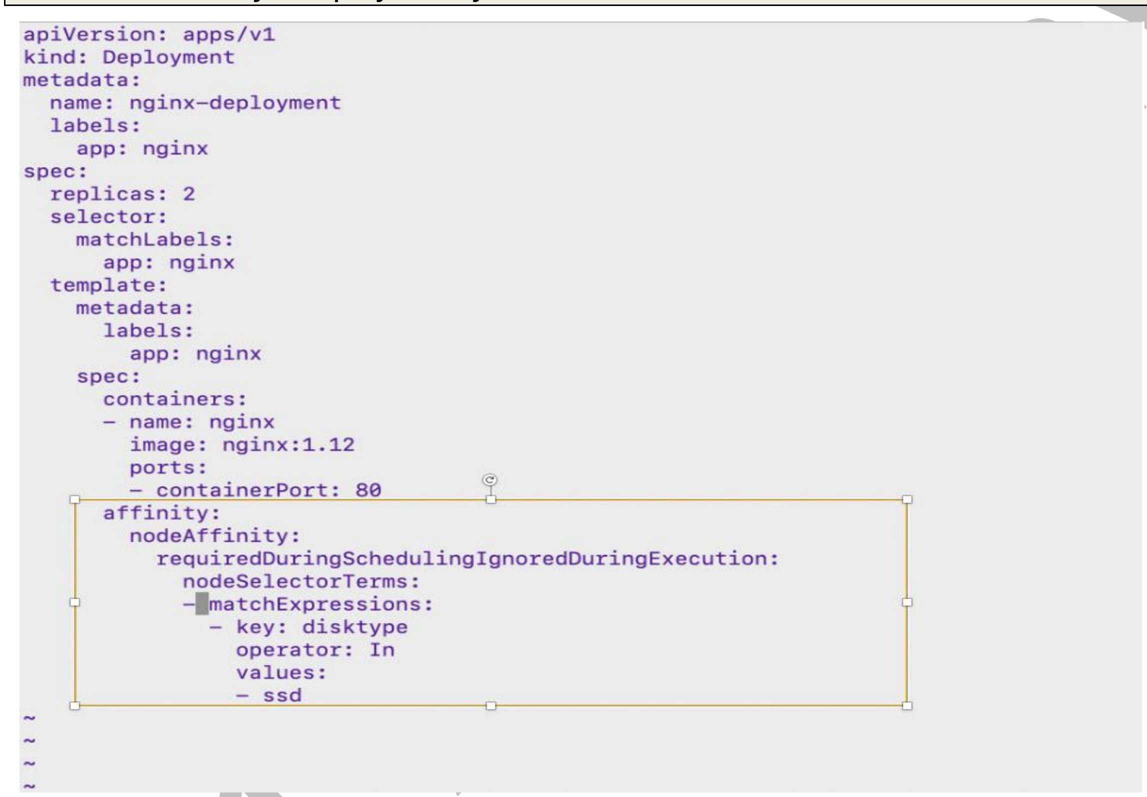
```
$ kubectl delete -f nodeaffinity-deployment.yaml
```

5.2 Creating Pod with Node Affinity required constraint

1. Define **"requiredDuringSchedulingIgnoredDuringExecution" Constraint**

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

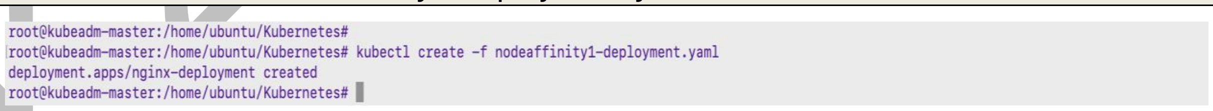
```
$ vim nodeaffinity1-deployment.yaml
```



```
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
~
~
~
~
```

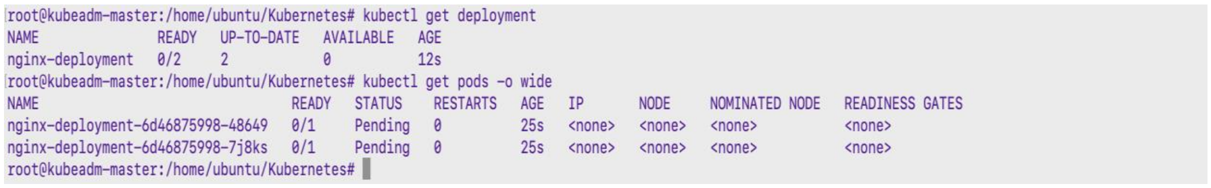
   b. Create deployment with kubectl create command. List the deployment and pods, see that the pods are in **pending state** as none of the nodes have the required label

```
$ 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#
```

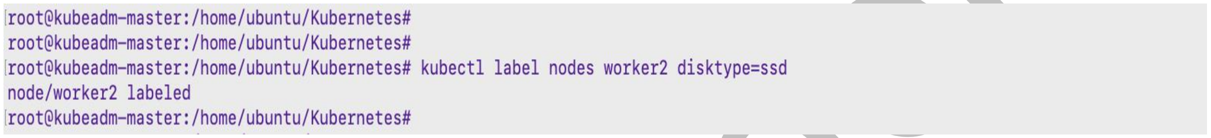
```
$ kubectl get deployment

$ 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#
```

5.3 Verify pod scheduling

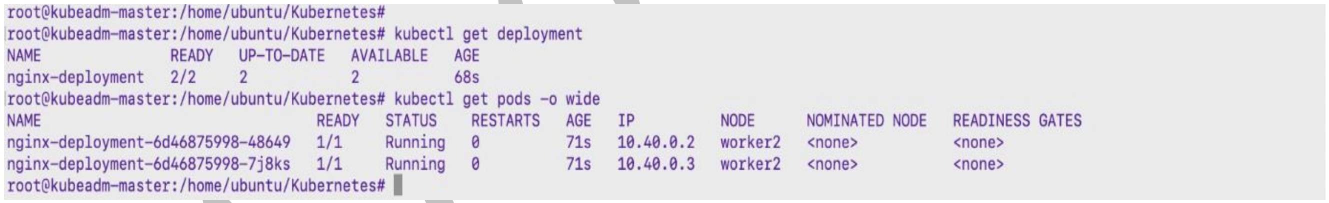
1. Label node “worker2” and notice that pending pods get scheduled on the labelled node
2. Label worker node “worker2” disk type=ssd

```
$ 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

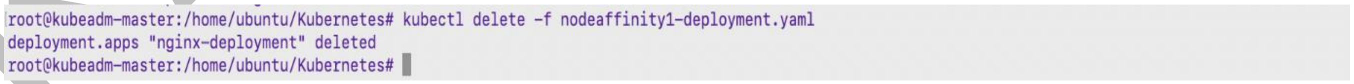
```
$ kubectl get deployment


$ 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#
```

5.4 Clean-Up Resources & Label added in this Exercise

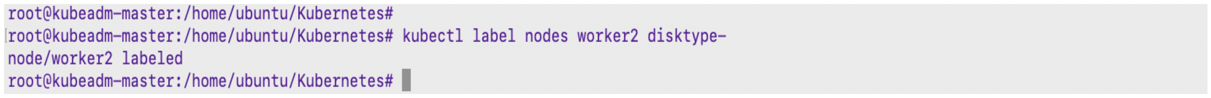
1. Delete the deployment

```
$ 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

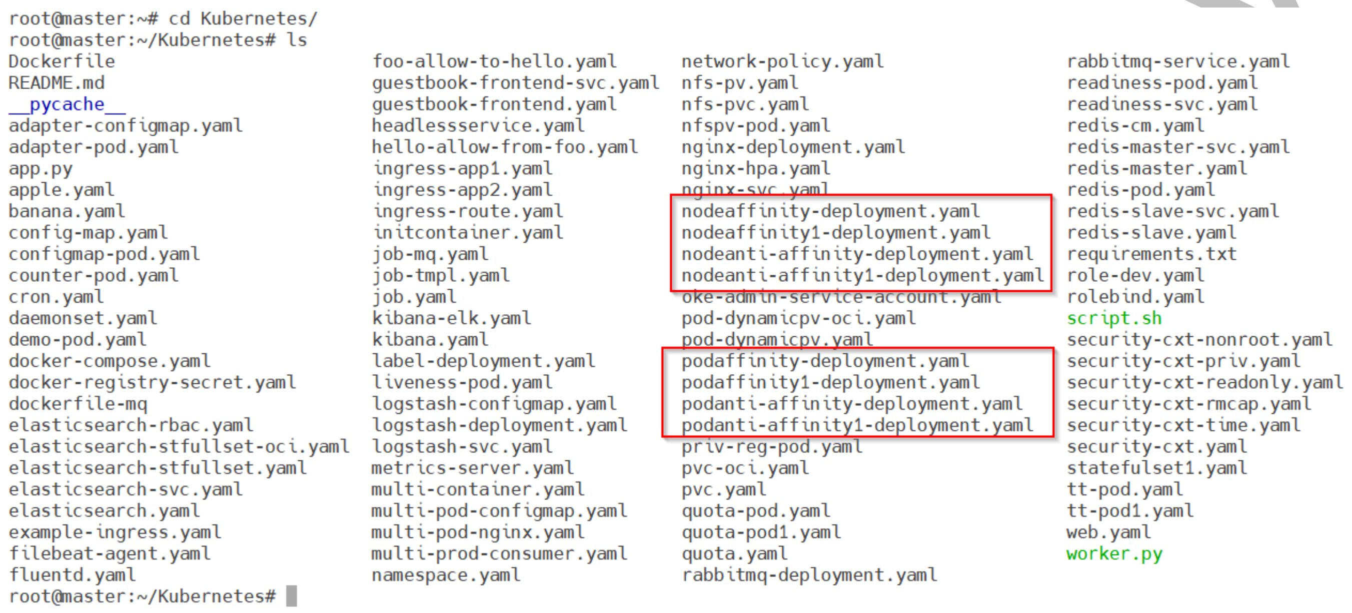
```
$ kubectl label nodes worker2 disktype-
```


```
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#
```

6 CONSTRAINING PODS WITH NODE ANTI-AFFINITY

Note: In below Sections we are going to use YAML files no need write complete yaml file because in CKA exam you can official Kubernetes documentation use Below GIT url to clone repo and use yaml files

```
$ git clone https://github.com/k21academyuk/Kubernetes
$ cd Kubernetes
```

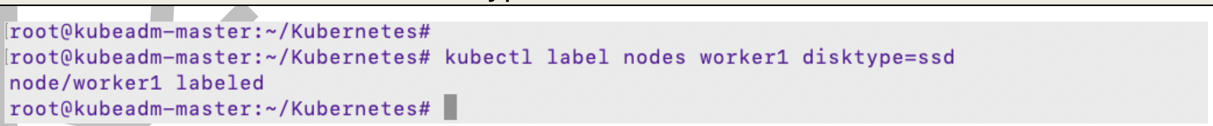


```
root@master:~# cd Kubernetes/
root@master:~/Kubernetes# ls
Dockerfile                        foo-allow-to-hello.yaml      network-policy.yaml                 rabbitmq-service.yaml
README.md                         guestbook-frontend-svc.yaml  nfs-pv.yaml                         readiness-pod.yaml
__pycache__                       guestbook-frontend.yaml      nfs-pvc.yaml                        readiness-svc.yaml
adapter-configmap.yaml            headlessservice.yaml         nfspv-pod.yaml                      redis-cm.yaml
adapter-pod.yaml                  hello-allow-from-foo.yaml    nginx-deployment.yaml               redis-master-svc.yaml
app.py                            ingress-app1.yaml            nginx-hpa.yaml                      redis-master.yaml
apple.yaml                        ingress-app2.yaml            nginx-svc.yaml                      redis-pod.yaml
banana.yaml                       ingress-route.yaml           nodeaffinity-deployment.yaml        redis-slave-svc.yaml
config-map.yaml                   initcontainer.yaml           nodeaffinity1-deployment.yaml       redis-slave.yaml
configmap-pod.yaml                job-mq.yaml                  nodeanti-affinity-deployment.yaml   requirements.txt
counter-pod.yaml                  job-tmpl.yaml                nodeanti-affinity1-deployment.yaml  role-dev.yaml
cron.yaml                         job.yaml                     oke-admin-service-account.yaml      rolebind.yaml
daemonset.yaml                    kibana-elk.yaml              pod-dynamicpv-oci.yaml              script.sh
demo-pod.yaml                     kibana.yaml                  pod-dynamicpv.yaml                  security-cxt-nonroot.yaml
docker-compose.yaml               label-deployment.yaml        podaffinity-deployment.yaml         security-cxt-priv.yaml
docker-registry-secret.yaml       liveness-pod.yaml            podaffinity1-deployment.yaml        security-cxt-readonly.yaml
dockerfile-mq                     logstash-configmap.yaml      podanti-affinity-deployment.yaml    security-cxt-rmcap.yaml
elasticsearch-rbac.yaml           logstash-deployment.yaml     podanti-affinity1-deployment.yaml   security-cxt-time.yaml
elasticsearch-stfullset-oci.yaml  logstash-svc.yaml            priv-reg-pod.yaml                   security-cxt.yaml
elasticsearch-stfullset.yaml      metrics-server.yaml          pvc-oci.yaml                        statefulset1.yaml
elasticsearch-svc.yaml            multi-container.yaml         pvc.yaml                            tt-pod.yaml
elasticsearch.yaml                multi-pod-configmap.yaml     quota-pod.yaml                      tt-pod1.yaml
example-ingress.yaml              multi-pod-nginx.yaml         quota-pod1.yaml                     web.yaml
filebeat-agent.yaml               multi-prod-consumer.yaml     quota.yaml                          worker.py
fluentd.yaml                      namespace.yaml               rabbitmq-deployment.yaml
root@master:~/Kubernetes# 
```

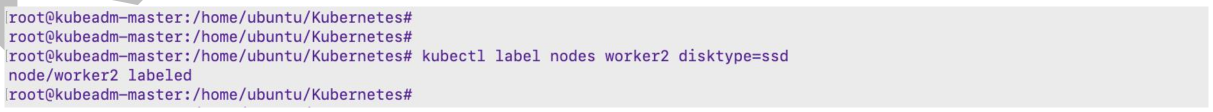
6.1 Label nodes “worker2” and “worker1”

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# 
```

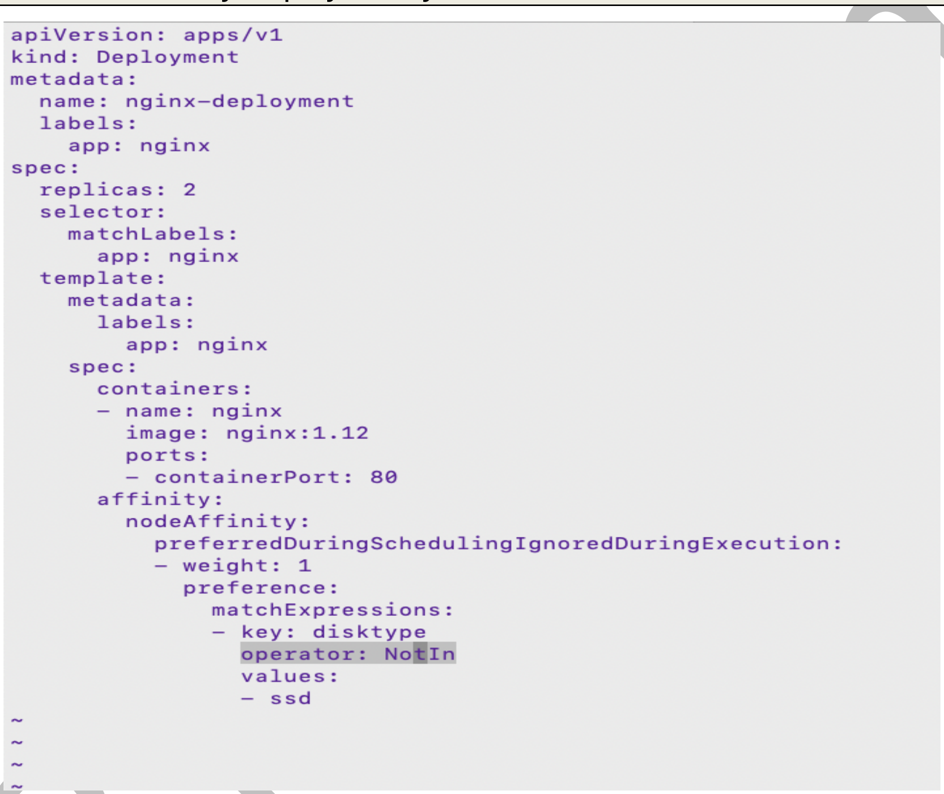


```
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes#
root@kubeadm-master:/home/ubuntu/Kubernetes# kubectl label nodes worker2 disktype=ssd
node/worker2 labeled
root@kubeadm-master:/home/ubuntu/Kubernetes#
```

6.2 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”**

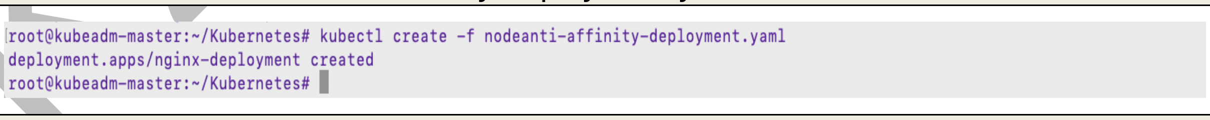
```
$ vim nodeanti-affinity-deployment.yaml
```



```
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
~
~
~
~
```

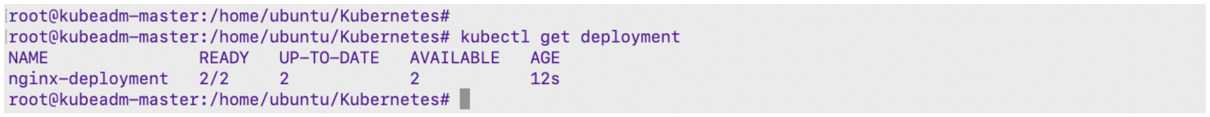
3. Create deployment with kubectl create command and verify that inspite of the node label pod gets placed as the condition is preferred one

```
$ 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#
```

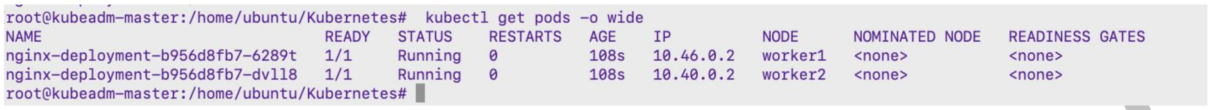
```
$ kubectl get deployment
```



```
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#
```

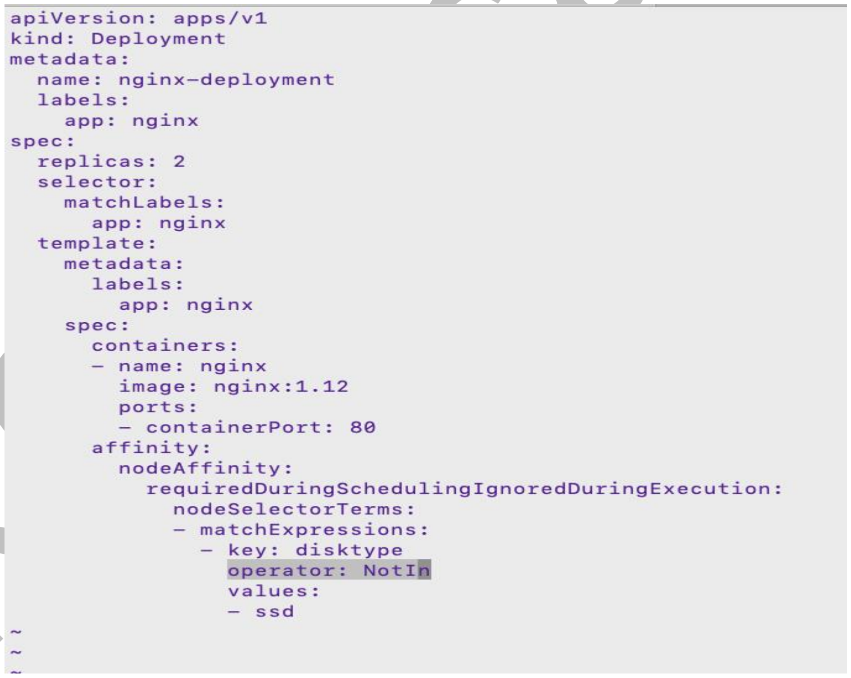
6.3 Delete the deployment

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

6.4 Creating Pod with Node Anti-Affinity required constraint

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

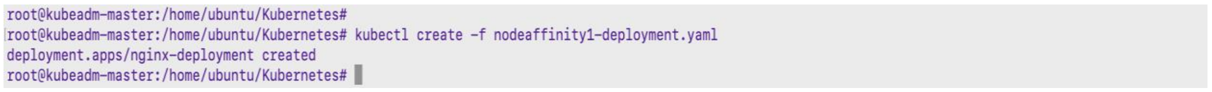
```
$ vim nodeanti-affinity1-deployment.yaml
```



```
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
~
~
~
```

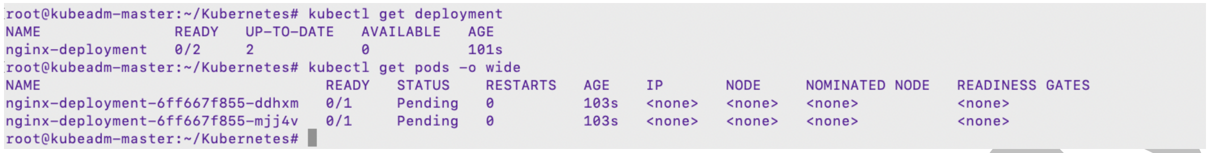
2. Create deployment with kubectl create command. List the deployment and pods, see that the pods are in pending state as all the nodes in the cluster have the label

```
$ 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#
```

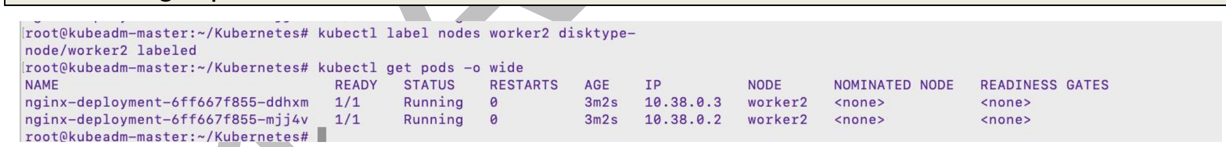
6.5 Verify Pod Scheduling

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
```

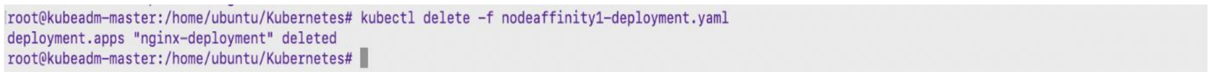


```
root@kubeadm-master:~/Kubernetes# kubectl label nodes worker2 disktype-
node/worker2 labeled
root@kubeadm-master:~/Kubernetes# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP          NODE      NOMINATED NODE   READINESS GATES
nginx-deployment-6ff667f855-ddhxm   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#
```

6.6 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#
```

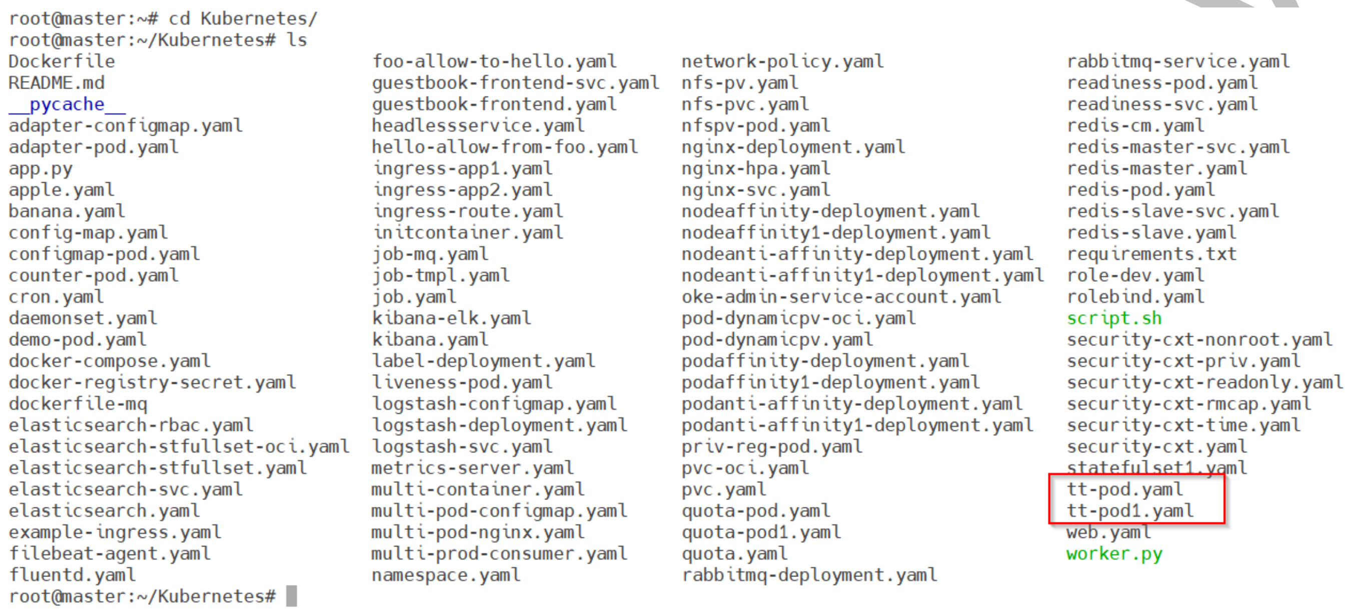
2. Remove the label added to worker1 node with kubectl label command and verify the label is removed

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

7 ADVANCED SCHEDULING WITH TAINT AND TOLERATIONS

Note: In below Sections we are going to use YAML files no need write complete yaml file because in CKA exam you can official Kubernetes documentation use Below GIT url to clone repo and use yaml files

```
$ git clone https://github.com/k21academyuk/Kubernetes

$ cd Kubernetes
```

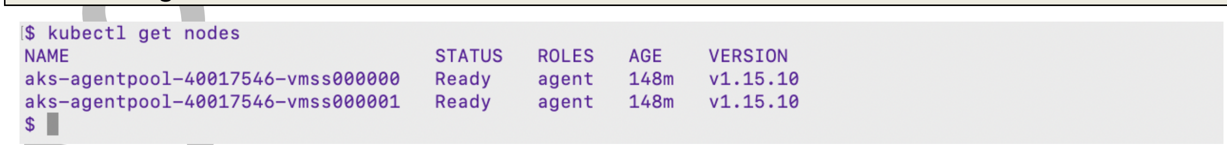


```
root@master:~# cd Kubernetes/
root@master:~/Kubernetes# ls
Dockerfile                        foo-allow-to-hello.yaml      network-policy.yaml                 rabbitmq-service.yaml
README.md                         guestbook-frontend-svc.yaml  nfs-pv.yaml                         readiness-pod.yaml
__pycache__                       guestbook-frontend.yaml      nfs-pvc.yaml                        readiness-svc.yaml
adapter-configmap.yaml            headlessservice.yaml         nfspv-pod.yaml                      redis-cm.yaml
adapter-pod.yaml                  hello-allow-from-foo.yaml    nginx-deployment.yaml               redis-master-svc.yaml
app.py                            ingress-app1.yaml            nginx-hpa.yaml                      redis-master.yaml
apple.yaml                        ingress-app2.yaml            nginx-svc.yaml                      redis-pod.yaml
banana.yaml                       ingress-route.yaml           nodeaffinity-deployment.yaml        redis-slave-svc.yaml
config-map.yaml                   initcontainer.yaml           nodeaffinity1-deployment.yaml       redis-slave.yaml
configmap-pod.yaml                job-mq.yaml                  nodeanti-affinity-deployment.yaml   requirements.txt
counter-pod.yaml                  job-tmpl.yaml                nodeanti-affinity1-deployment.yaml  role-dev.yaml
cron.yaml                         job.yaml                     oke-admin-service-account.yaml      rolebind.yaml
daemonset.yaml                    kibana-elk.yaml              pod-dynamicpv-oci.yaml              script.sh
demo-pod.yaml                     kibana.yaml                  pod-dynamicpv.yaml                  security-cxt-nonroot.yaml
docker-compose.yaml               label-deployment.yaml        podaffinity-deployment.yaml         security-cxt-priv.yaml
docker-registry-secret.yaml       liveness-pod.yaml            podaffinity1-deployment.yaml        security-cxt-readonly.yaml
dockerfile-mq                     logstash-configmap.yaml      podanti-affinity-deployment.yaml    security-cxt-rmcap.yaml
elasticsearch-rbac.yaml           logstash-deployment.yaml     podanti-affinity1-deployment.yaml   security-cxt-time.yaml
elasticsearch-stfullset-oci.yaml  logstash-svc.yaml            priv-reg-pod.yaml                   security-cxt.yaml
elasticsearch-stfullset.yaml      metrics-server.yaml          pvc-oci.yaml                        statefulset1.yaml
elasticsearch-svc.yaml            multi-container.yaml         pvc.yaml                            tt-pod.yaml
elasticsearch.yaml                multi-pod-configmap.yaml     quota-pod.yaml                      tt-pod1.yaml
example-ingress.yaml              multi-pod-nginx.yaml         quota-pod1.yaml                     web.yaml
filebeat-agent.yaml               multi-prod-consumer.yaml     quota.yaml                          worker.py
fluentd.yaml                      namespace.yaml               rabbitmq-deployment.yaml
root@master:~/Kubernetes#
```

7.1 Tainting a Node to Simulate Advanced Scheduling

1. View all the nodes in the cluster

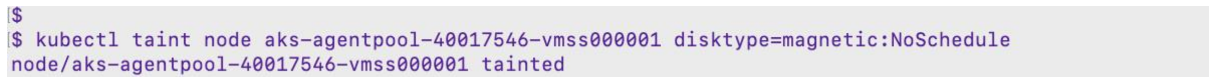
```
$ kubectl get nodes
```



```
$ kubectl get nodes
NAME                                STATUS   ROLES   AGE    VERSION
aks-agentpool-40017546-vmss000000   Ready    agent   148m   v1.15.10
aks-agentpool-40017546-vmss000001   Ready    agent   148m   v1.15.10
$
```

2. Taint one of the nodes by using its name

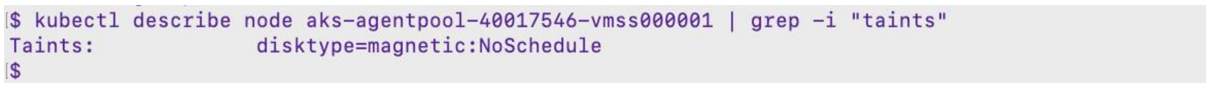
```
$ kubectl taint node aks-agentpool-40017546-vmss000001
disktype=magnetic:NoSchedule
```



```
$
$ kubectl taint node aks-agentpool-40017546-vmss000001 disktype=magnetic:NoSchedule
node/aks-agentpool-40017546-vmss000001 tainted
```

3. Verify that the taint was applied to the desired node

```
$ kubectl describe node aks-agentpool-40017546-vmss000001 | grep -i "taints"
```

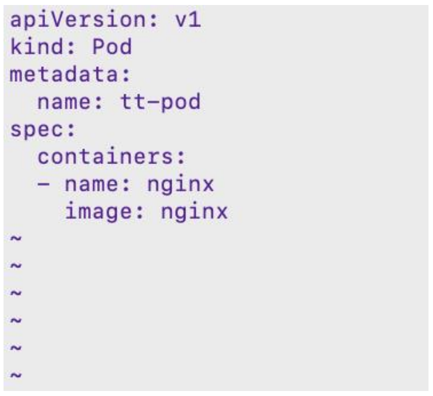


```
$ kubectl describe node aks-agentpool-40017546-vmss000001 | grep -i "taints"
Taints:             disktype=magnetic:NoSchedule
$
```

7.2 Creating Pod without Toleration

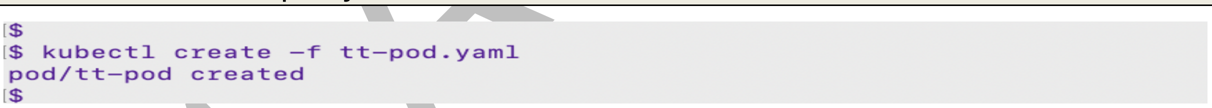
1. View the content of tt-pod.yaml file and create pod using the yaml file

```
$ vi tt-pod.yaml
```



```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod
spec:
  containers:
  - name: nginx
    image: nginx
~
~
~
~
~
~
```

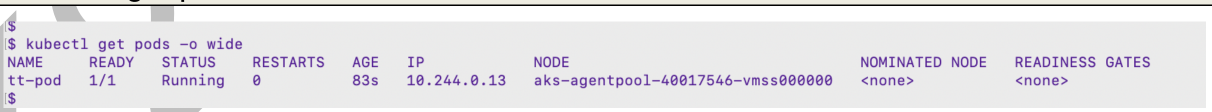
```
$ kubectl create -f tt-pod.yaml
```



```
$
$ kubectl create -f tt-pod.yaml
pod/tt-pod created
$
```

2. Verify the pod status. Notice that it was scheduled on the node which is not tainted

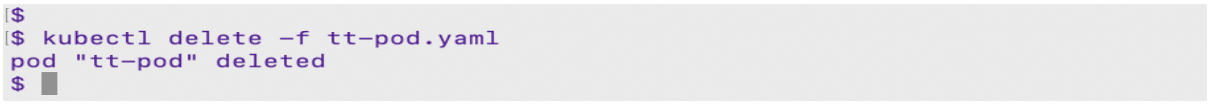
```
$ kubectl get pods -o wide
```



```
$
$ kubectl get pods -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP            NODE                                NOMINATED NODE   READINESS GATES
tt-pod   1/1     Running   0          83s   10.244.0.13   aks-agentpool-40017546-vmss000000   <none>           <none>
$
```

3. Delete the pod created in this task

```
$ kubectl delete -f tt-pod.yaml
```

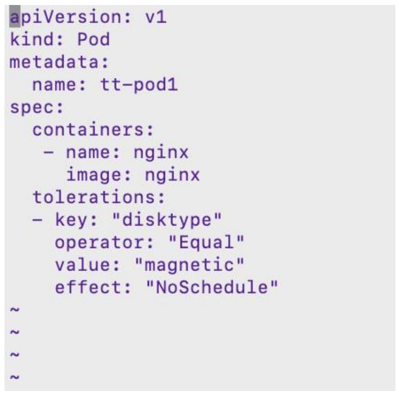


```
$
$ kubectl delete -f tt-pod.yaml
pod "tt-pod" deleted
$
```

7.3 Creating a Pod with Toleration

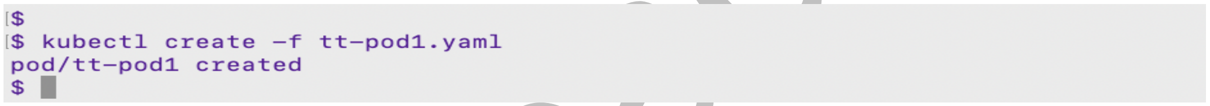
1. View the content of tt-pod1.yaml file and create pod using the yaml file

```
$ vi tt-pod1.yaml
```



```
apiVersion: v1
kind: Pod
metadata:
  name: tt-pod1
spec:
  containers:
   - name: nginx
     image: nginx
  tolerations:
  - key: "disktype"
    operator: "Equal"
    value: "magnetic"
    effect: "NoSchedule"
~
~
~
~
```

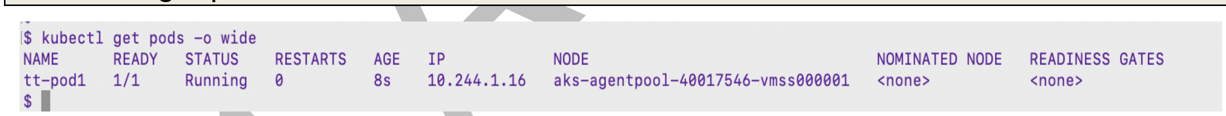
```
$ kubectl create -f tt-pod1.yaml
```



```
$
$ kubectl create -f tt-pod1.yaml
pod/tt-pod1 created
$
```

2. Verify the pod status. Notice that it was scheduled on the tainted node

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



```
$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE                                NOMINATED NODE   READINESS GATES
tt-pod1   1/1     Running   0          8s    10.244.1.16   aks-agentpool-40017546-vmss000001   <none>           <none>
$
```

3. Delete the pod created in this task

```
$ kubectl delete -f tt-pod1.yaml
```

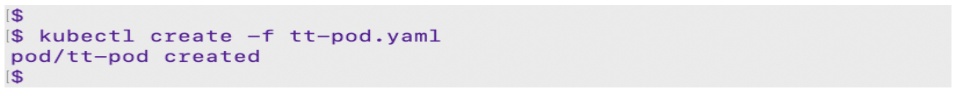
4. Delete the taint from the node

```
$ kubectl taint node <node_name> disktype-
$ kubectl describe nodes <node_name> | grep -i taint
```

7.4 Simulate eviction of Pod using NoSchedule effect

1. Again create a pod using tt-pod.yaml file. It doesn't have any toleration defined

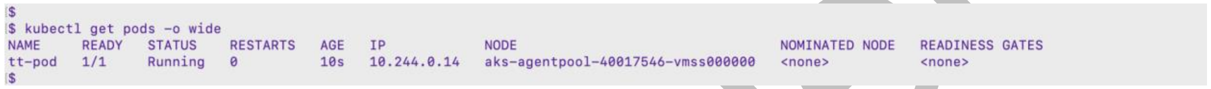
```
$ kubectl create -f tt-pod.yaml
```



```
$
$ kubectl create -f tt-pod.yaml
pod/tt-pod created
$
```

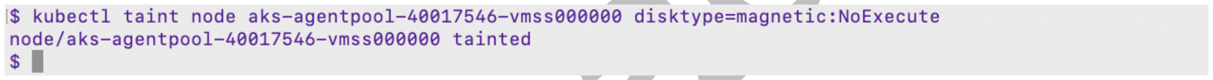
2. Taint the node on which the Pod was scheduled

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

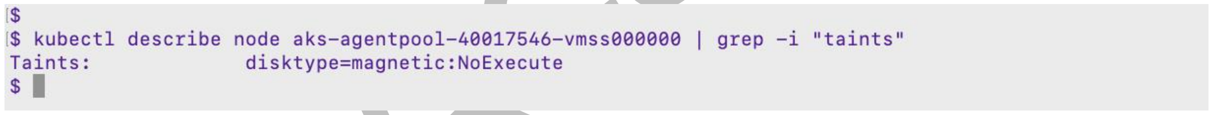


```
$
$ kubectl get pods -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP            NODE                                NOMINATED NODE   READINESS GATES
tt-pod   1/1     Running   0          10s   10.244.0.14   aks-agentpool-40017546-vmss000000   <none>           <none>
$
```

```
$ kubectl taint node <node_name> disktype=magnetic:NoExecute
```



```
$ kubectl taint node aks-agentpool-40017546-vmss000000 disktype=magnetic:NoExecute
node/aks-agentpool-40017546-vmss000000 tainted
$
```



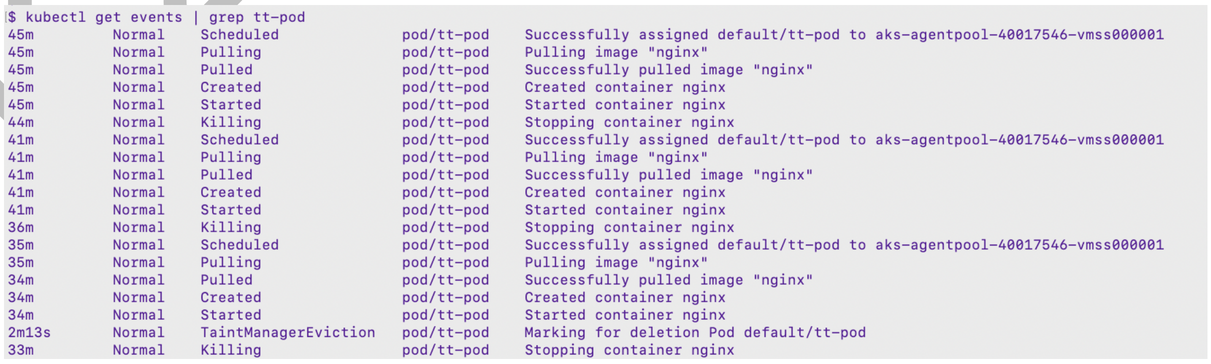
```
$
$ kubectl describe node aks-agentpool-40017546-vmss000000 | grep -i "taints"
Taints:             disktype=magnetic:NoExecute
$
```

3. Verify the pods status again and see that the pod is evicted

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

4. View recent events to see that the pod was evicted due to the taint

```
$ kubectl get events
```



```
$ kubectl get events | grep tt-pod
45m         Normal    Scheduled              pod/tt-pod    Successfully assigned default/tt-pod to aks-agentpool-40017546-vmss000001
45m         Normal    Pulling                pod/tt-pod    Pulling image "nginx"
45m         Normal    Pulled                 pod/tt-pod    Successfully pulled image "nginx"
45m         Normal    Created                pod/tt-pod    Created container nginx
45m         Normal    Started                pod/tt-pod    Started container nginx
44m         Normal    Killing                pod/tt-pod    Stopping container nginx
41m         Normal    Scheduled              pod/tt-pod    Successfully assigned default/tt-pod to aks-agentpool-40017546-vmss000001
41m         Normal    Pulling                pod/tt-pod    Pulling image "nginx"
41m         Normal    Pulled                 pod/tt-pod    Successfully pulled image "nginx"
41m         Normal    Created                pod/tt-pod    Created container nginx
41m         Normal    Started                pod/tt-pod    Started container nginx
36m         Normal    Killing                pod/tt-pod    Stopping container nginx
35m         Normal    Scheduled              pod/tt-pod    Successfully assigned default/tt-pod to aks-agentpool-40017546-vmss000001
35m         Normal    Pulling                pod/tt-pod    Pulling image "nginx"
34m         Normal    Pulled                 pod/tt-pod    Successfully pulled image "nginx"
34m         Normal    Created                pod/tt-pod    Created container nginx
34m         Normal    Started                pod/tt-pod    Started container nginx
2m13s       Normal    TaintManagerEviction   pod/tt-pod    Marking for deletion Pod default/tt-pod
33m         Normal    Killing                pod/tt-pod    Stopping container nginx
```

5. Delete the pod and taint from the node

```
$ kubectl delete -f tt-pod.yaml

$ kubectl taint node <node_name> disktype-
$ kubectl describe nodes <node_name> | grep -i taint
```

8 TROUBLESHOOTING

8.1 Nodes are remain tainted even after untaint them

Issue: Nodes are remain tainted even after removing the taint.

```
root@master:/home/azureuser/Kubernetes# kubectl taint nodes worker1
node.kubernetes.io/unreachable:NoExecute-
node/worker1 untainted
root@master:/home/azureuser/Kubernetes# kubectl describe nodes worker1 | grep -i Taints
Taints:             node.kubernetes.io/unreachable:NoExecute
```

Reason: Tainted nodes are down

Fix: Tainted nodes are down, make sure they are in running state. Sometimes it takes few minutes to come up.

9 SUMMARY

In this guide we Covered:

- Constraining pods with node selector
- Constraining pods with node affinity
- Constraining pods with node anti-affinity
- Constraining pods with taint and toleration