TAINTS & TOLERATIONS

Taints:

Taints are applied to nodes and signal that the node should not accept any pods that do not tolerate the taint.

Tolerations:

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

Now I list nodes

```
[node1 ~]$ kubectl get node
NAME     STATUS     ROLES           AGE   VERSION
node1    NotReady   control-plane   88s   v1.27.2
node2    NotReady   <none>          40s   v1.27.2
node3    NotReady   <none>          33s   v1.27.2
```

Now I cerate basic pod.

```
[node1 ~]$ kubectl run mypod --image=nginx
pod/mypod created
[node1 ~]$  kubectl get pod -owide
NAME    READY   STATUS              RESTARTS   AGE     IP       NODE    NOMINATED NODE   READINESS GATES
mypod   0/1     ContainerCreating   0          6m20s   <none>   node3   <none>           <none>
[node1 ~]$
```

Now I set an taint3 to one specific node

kubectl taint nodes mynode mykey=myvalue:NoSchedule

```
[node1 ~]$ kubectl taint nodes node3  mykey=myvalue:NoSchedule
node/node3 tainted
[node1 ~]$
```

Now I set an tollerance to an pod and I run the pod

Vi pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod1
spec:
  containers:
  - name: mycontainer
    image: nginx
  tolerations:
  - key: "mykey"
    operator: "Equal"
    value: "myvalue"
    effect: "NoSchedule"
~
```

```
[node1 ~]$ kubectl get pod -owide
NAME     READY   STATUS    RESTARTS   AGE     IP          NODE    NOMINATED NODE   READINESS GATES
mypod    1/1     Running   0          3m13s   10.5.2.2    node3   <none>           <none>
mypod1   1/1     Running   0          15s     10.5.2.3    node3   <none>           <none>
[node1 ~]$
```

Now I again run same pod in different image but tollerence was same

```
[node1 ~]$ vi pod1.yaml
[node1 ~]$ kubectl create -f pod1.yaml
pod/mypod3 created
```

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod3
spec:
  containers:
  - name: mycontainer
    image: httpd
  tolerations:
  - key: "mykey"
    operator: "Equal"
    value: "myvalue"
    effect: "NoSchedule"
~
```

Even though tolllerance was same the pod locted to node2 becaause we didn’t sepecify any taint to node 2

```
NAME     READY   STATUS    RESTARTS   AGE     IP          NODE    NOMINATED NODE   READINESS GATES
mypod    1/1     Running   0          9m15s   10.5.2.2    node3   <none>           <none>
mypod1   1/1     Running   0          6m17s   10.5.2.3    node3   <none>           <none>
mypod2   1/1     Running   0          3m54s   10.5.2.4    node3   <none>           <none>
mypod3   1/1     Running   0          12s     10.5.1.5    node2   <none>           <none>
[node1 ~]$ vi pod1.yaml
[node1 ~]$
```

Now I set taint to node2

```
[node1 ~]$ kubectl taint nodes node2 myname=spvp:NoSchedule
node/node2 tainted
[node1 ~]$
```

Now if I run the pod in same tollerance now it will locate in node3 because the node 2 tolerance was different

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod4
spec:
  containers:
  - name: mycontainer
    image: httpd
  tolerations:
  - key: "mykey"
    operator: "Equal"
    value: "myvalue"
    effect: "NoSchedule"
~
```

```
[node1 ~]$ kubectl get pod -owide
NAME     READY   STATUS    RESTARTS   AGE     IP          NODE    NOMINATED NODE   READINESS GATES
mypod    1/1     Running   0          39m     10.5.2.2    node3   <none>           <none>
mypod1   1/1     Running   0          36m     10.5.2.3    node3   <none>           <none>
mypod2   1/1     Running   0          33m     10.5.2.4    node3   <none>           <none>
mypod3   1/1     Running   0          30m     10.5.1.5    node2   <none>           <none>
mypod4   1/1     Running   0          3m35s   10.5.2.5    node3   <none>           <none>
```

Now I describe the tanint of the node3

```
[node1 ~]$ kubectl describe node node3
Name:               node3
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node3
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: /run/docker/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Wed, 04 Dec 2024 03:59:56 +0000
Taints:             mykey=myvalue:NoSchedule
                    myname=spvp:NoSchedule
Unschedulable:      false
Lease:
  HolderIdentity:  node3
  AcquireTime:     <unset>
  RenewTime:       Wed, 04 Dec 2024 04:39:26 +0000
```

Now I remove second taint from node3

```
[node1 ~]$ kubectl taint nodes node3  myname=spvp:NoSchedule-
node/node3 untainted
```

```
[node1 ~]$ kubectl describe node node3
Name:               node3
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node3
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: /run/docker/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Wed, 04 Dec 2024 03:59:56 +0000
Taints:             mykey=myvalue:NoSchedule
Unschedulable:      false
```

Different effect we used

NoSchedule: This is the default effect. If a pod does not tolerate this taint, it will not be scheduled on the node.

PreferNoSchedule: This is the recommended effect to use if you want the system to avoid scheduling a pod onto a node, but it will still schedule the pod on the node if it is necessary.

NoExecute: This effect means that the pod will be evicted from the node if it is already running on the node, and will not be scheduled onto the node if it is not yet running on the node.

Before giving noexecute effect to node3 the running pod in node3

```
[node1 ~]$ kubectl get pod -owide
NAME     READY   STATUS    RESTARTS   AGE   IP          NODE    NOMINATED NODE   READINESS GATES
mypod    1/1     Running   0          67m   10.5.2.2    node3   <none>           <none>
mypod1   1/1     Running   0          64m   10.5.2.3    node3   <none>           <none>
mypod2   1/1     Running   0          61m   10.5.2.4    node3   <none>           <none>
mypod3   1/1     Running   0          58m   10.5.1.5    node2   <none>           <none>
mypod4   1/1     Running   0          31m   10.5.2.5    node3   <none>           <none>
mypod5   1/1     Running   0          12s   10.5.2.6    node3   <none>           <none>
[node1 ~]$
```

Now I use the effect **noexecute**

```
[node1 ~]$ vi pod2.yaml
[node1 ~]$ kubectl create -f pod2.yaml
pod/mypod5 created
[node1 ~]$
```

Change the taint to node3

```
[node1 ~]$ kubectl taint nodes node3 mykey=myvalue:NoExecute
node/node3 tainted
```

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod5
spec:
  containers:
  - name: mycontainer
    image: httpd
  tolerations:
  - key: "mykey"
    operator: "Equal"
    value: "myvalue"
    effect: "NoExecute"
~
~
```

If I give the noexecute effect the before running pod will remove from the node3.and the new node will execute.

```
[node1 ~]$ kubectl get pod -owide
NAME     READY   STATUS    RESTARTS   AGE    IP         NODE    NOMINATED NODE   READINESS GATES
mypod3   1/1     Running   0          65m    10.5.1.5   node2   <none>           <none>
mypod5   1/1     Running   0          5m7s   10.5.2.7   node3   <none>           <none>
[node1 ~]$
```

Now I give the tolleernce effect to pod6 from taint2 node2

Now I see taint effect in node2

```
[node1 ~]$ kubectl describe nodes node2
Name:               node2
Roles:              <none>
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=node2
                    kubernetes.io/os=linux
Annotations:        kubeadm.alpha.kubernetes.io/cri-socket: /run/docker/containerd/containerd.sock
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Wed, 04 Dec 2024 03:59:34 +0000
Taints:             myname=spvp:NoSchedule
```

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod6
spec:
  containers:
  - name: mycontainer
    image: nginx
  tolerations:
  - key: "myname"
    operator: "Equal"
    value: "spvp"
    effect: "NoSchedule"
~
```

```
[node1 ~]$ vi pod.yaml
[node1 ~]$ kubectl create -f pod.yaml
pod/mypod6 created
```

Now I get the pods

```
[node1 ~]$ kubectl get pod -owide
NAME     READY   STATUS    RESTARTS   AGE     IP         NODE    NOMINATED NODE   READINESS GATES
mypod3   1/1     Running   0          69m     10.5.1.5   node2   <none>           <none>
mypod5   1/1     Running   0          8m52s   10.5.2.7   node3   <none>           <none>
mypod6   1/1     Running   0          8s      10.5.1.6   node2   <none>           <none>
```

Now I mannually shedule the pod to node2.even though I not set the tollerence. it will shedule the pods.

[node1 ~]$ vi pod3.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: manually-scheduled-pod
spec:
  nodeName: node2
  containers:
  - name: my-container
    image: nginx
~
```

[node1 ~]$ kubectl create -f pod3.yaml

pod/manually-scheduled-pod created

```
[node1 ~]$ kubectl create -f pod3.yaml
pod/manually-scheduled-pod created
[node1 ~]$ kubectl get pod -owide
NAME                    READY   STATUS    RESTARTS   AGE   IP         NODE    NOMINATED NODE   READINESS GATES
manually-scheduled-pod  1/1     Running   0          11s   10.5.1.7   node2   <none>           <none>
mypod3                  1/1     Running   0          79m   10.5.1.5   node2   <none>           <none>
mypod5                  1/1     Running   0          19m   10.5.2.7   node3   <none>           <none>
mypod6                  1/1     Running   0          10m   10.5.1.6   node2   <none>           <none>
```