TAINTS AND TOLERATIONS

Taints and Tolerations help control which pods can be scheduled on which nodes. Taints are applied to nodes whereas Tolerations are applied to pods.

Nodes in a Kubernetes cluster often have different hardware specifications—some may be memory-optimized, CPU-optimized, GPU-enabled. Taints and tolerations allow us to ensure that workloads are scheduled onto the appropriate nodes based on these capabilities.

→ Taints and Tolerations Use cases:

1. **Dedicated Nodes:** Pods with specific hardware needs—such as GPU-intensive or memory-heavy workloads—are scheduled onto nodes optimized for those requirements.
4. **Isolating Critical Workloads:** Critical applications are deployed on production nodes, while new features are tested on development nodes. This separation is enforced using taints and tolerations.
7. **Preventing Scheduling on Control Plane Nodes:** Workloads are not deployed on control plane nodes unless explicitly scheduled. This is because control plane nodes are tainted by default, and regular pods do not include the required tolerations to run on them.
8. **Maintenance Mode:** When a node is undergoing maintenance or upgrades, it can be tainted (e.g., maintenance=true) to prevent new pods from being scheduled on it during that period.
9. **Node Resource Constraints:** Taints and tolerations can be used to prevent high-compute workloads from being scheduled on standard compute nodes, ensuring they run only on nodes with adequate resources.

Lets Explore Taints and Tolerations Practically:

```
kubectl get nodes
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get nodes
NAME                           STATUS   ROLES           AGE   VERSION
rayeez-cluster-control-plane   Ready    control-plane   91m   v1.31.4
rayeez-cluster-worker          Ready    <none>          91m   v1.31.4
rayeez-cluster-worker2         Ready    <none>          91m   v1.31.4
```

Lets apply taint to worker nodes:

```
kubectl taint node rayeez-cluster-worker storage=ssd:NoSchedule
kubectl taint node rayeez-cluster-worker2 storage=hdd:NoSchedule
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl taint node rayeez-cluster-worker storage=ssd:NoSchedule
node/rayeez-cluster-worker tainted
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl taint node rayeez-cluster-worker2 storage=hdd:NoSchedule
node/rayeez-cluster-worker2 tainted
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$
```

Here **key**=storage, **operator** is "=", **value**=ssd or hdd, **effect**=NoSchedule

Lets run a test-pod imperatively and see on which node will it get scheduled:

```
kubectl run test-pod --image=nginx
```

verify;

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

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP       NODE     NOMINATED NODE   READINESS GATES
test-pod   0/1     Pending   0          10s   <none>   <none>   <none>           <none>
```

This pod is not scheduled on any node of the cluster. As we know, Worker Nodes are tainted. But what about Control plane node?

→ Control plane Node is tainted already by the Cluster:

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl describe node rayeez-cluster-worker | grep -i taint
Taints:             storage=ssd:NoSchedule
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl describe node rayeez-cluster-worker2 | grep -i taint
Taints:             storage=hdd:NoSchedule
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl describe node rayeez-cluster-control-plane | grep -i taint
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
```

Test-pod do not have any tolerations. Hence its not getting scheduled on any of the node.

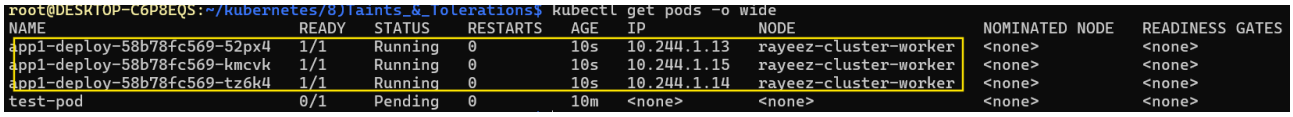
Lets deploy a deployment with proper tolerations:

```
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: app1-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      tolerations:
        - key: "storage"
          operator: "Equal"
          value: "ssd"
          effect: "NoSchedule"
      containers:
        - name: nginx-container
          image: nginx
```

Lets apply it:

```
kubectl apply -f deploy.yaml
```



Pods are deployed on worker node-1:

Lets deploy a pod manually on worker node 2:

```
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  tolerations:
    - key: "storage"
      operator: "Equal"
      value: "hdd"
```

```
      effect: "NoSchedule"
  containers:
    - name: nginx-container
      image: nginx
```

Apply;

```
kubectl apply -f pod.yaml
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE     IP            NODE                      NOMINATED NODE   READINESS GATES
app1-deploy-58b78fc569-52px4  1/1     Running   0          4m22s   10.244.1.13   rayeez-cluster-worker     <none>           <none>
app1-deploy-58b78fc569-kmcvk  1/1     Running   0          4m22s   10.244.1.15   rayeez-cluster-worker     <none>           <none>
app1-deploy-58b78fc569-tz6k4  1/1     Running   0          4m22s   10.244.1.14   rayeez-cluster-worker     <none>           <none>
nginx-pod                     1/1     Running   0          31s     10.244.2.3    rayeez-cluster-worker2    <none>           <none>
test-pod                      0/1     Pending   0          15m     <none>        <none>                    <none>           <none>
```

A node can have more than one taints and a pod can have more than one tolerations.

Lets add one more taint to both nodes to segregate resources as per the environment.

```
kubectl taint node rayeez-cluster-worker
env=prod:NoSchedule
kubectl taint node rayeez-cluster-worker2
env=dev:NoSchedule
```

Let deploy pods individually on both environments:

```
apiVersion: v1
kind: Pod
metadata:
  name: prod-pod
spec:
  tolerations:
    - key: "storage"
      operator: "Exists"
      effect: "NoSchedule"
    - key: "env"
      operator: "Equal"
      value: "prod"
      effect: "NoSchedule"
  containers:
    - name: nginx-container
      image: nginx
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP            NODE                   NOMINATED NODE   READINESS GATES
prod-pod   1/1     Running   0          11s   10.244.1.16   rayeez-cluster-worker  <none>           <none>
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$
```

→ This pod was deployed on Worker node-1 due to the matching taint and toleration. The Exists operator ensures that the node must have the key storage, and the additional toleration ensures that the pod is scheduled in the production environment.

Similarly, Lets deploy another pod on worker node-2 keeping matching tolerations for development environment.

```
apiVersion: v1
kind: Pod
metadata:
  name: dev-pod
spec:
  tolerations:
    - key: "storage"
      operator: "Exists"
      effect: "NoSchedule"
    - key: "env"
      operator: "Equal"
      value: "dev"
      effect: "NoSchedule"
  containers:
    - name: nginx-container
      image: nginx
```

Apply;

```
kubectl apply -f dev-pod.yaml
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP            NODE                   NOMINATED NODE   READINESS GATES
dev-pod    1/1     Running   0          5s      10.244.2.4    rayeez-cluster-worker2 <none>           <none>
prod-pod   1/1     Running   0          8m43s   10.244.1.16   rayeez-cluster-worker  <none>           <none>
```

To remove taints from nodes, Just add a hyphen at end in the command:

```
kubectl taint node rayeez-cluster-worker
storage=ssd:NoSchedule-
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl taint node rayeez-cluster-worker storage=ssd:NoSchedule-
node/rayeez-cluster-worker untainted
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl taint node rayeez-cluster-worker2 storage=hdd:NoSchedule-
node/rayeez-cluster-worker2 untainted
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl taint node rayeez-cluster-worker env=prod:NoSchedule-
node/rayeez-cluster-worker untainted
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl taint node rayeez-cluster-worker2 env=dev:NoSchedule-
node/rayeez-cluster-worker2 untainted
```

Lets Explore various **effect** options:

Effect will ensure that only a pod with matching tolerations are allowed to be deployed on server.

→ Lets deploy 3 replicas without tolerations on the cluster with untainted Nodes.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app1-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx-container
          image: nginx
```

Lets apply;

```
kubectl apply -f deploy.yaml
```

Verify;

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

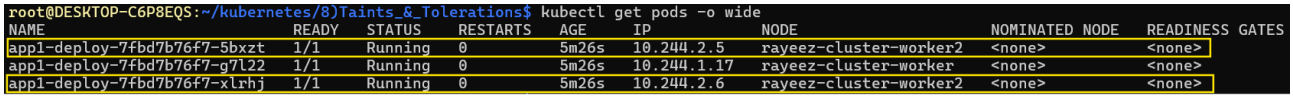
```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP            NODE                      NOMINATED NODE   READINESS GATES
app1-deploy-7fbd7b76f7-5bxzt  1/1     Running   0          78s   10.244.2.5    rayeez-cluster-worker2    <none>           <none>
app1-deploy-7fbd7b76f7-g7l22  1/1     Running   0          78s   10.244.1.17   rayeez-cluster-worker     <none>           <none>
app1-deploy-7fbd7b76f7-xlrhj  1/1     Running   0          78s   10.244.2.6    rayeez-cluster-worker2    <none>           <none>
```

2 replicas are deployed on worker node-2 and one replica is deployed on worker node-1

Let apply a taint to worker node-2 with **NoSchedule** effect

```
kubectl taint node rayeez-cluster-worker2 color=green:NoSchedule
```

Pods are still running on worker node-2:

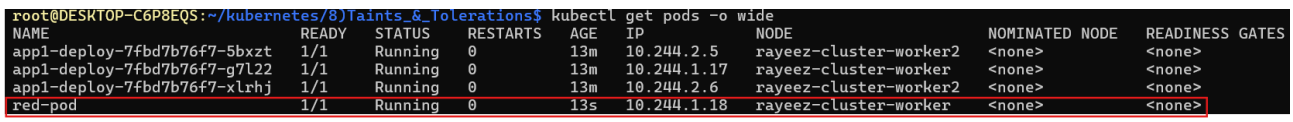


NoSchedule effect: New pods are not allowed to schedule unless they have matching tolerations, while existing pods on the node remain unaffected.

→ Let's try scheduling a pod that does not match the taints on worker node-2.

```
apiVersion: v1
kind: Pod
metadata:
  name: red-pod
spec:
  tolerations:
    - key: "color"
      operator: "Equal"
      value: "red"
      effect: "NoSchedule"
  containers:
    - name: nginx-container
      image: nginx
```

This pod is scheduled on worker node-1(Untainted Node):



Let apply a taint to worker node-1 with **PreferNoSchedule** effect.

→ Let's deploy two pods—one with tolerations matching worker node-1, and another with tolerations that do not match either node.

```
kubectl taint node rayeez-cluster-worker
color=blue:PreferNoSchedule
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl describe node rayeez-cluster-worker | grep -i taints
Taints:             color=blue:PreferNoSchedule
```

```
apiVersion: v1
kind: Pod
metadata:
  name: blue-pod
spec:
  tolerations:
    - key: "color"
      operator: "Equal"
      value: "blue"
      effect: "PreferNoSchedule"
  containers:
    - name: nginx-container
      image: nginx
```

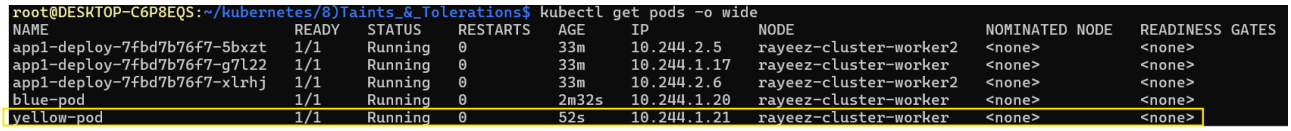
Pod is deployed on worker node-1 because of matching tolerations:

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP            NODE                      NOMINATED NODE   READINESS GATES
app1-deploy-7fbd7b76f7-5bxzt  1/1     Running   0          31m   10.244.2.5    rayeez-cluster-worker2    <none>           <none>
app1-deploy-7fbd7b76f7-g7l22  1/1     Running   0          31m   10.244.1.17   rayeez-cluster-worker     <none>           <none>
app1-deploy-7fbd7b76f7-xlrhj  1/1     Running   0          31m   10.244.2.6    rayeez-cluster-worker2    <none>           <none>
blue-pod                      1/1     Running   0          11s   10.244.1.20   rayeez-cluster-worker     <none>           <none>
```

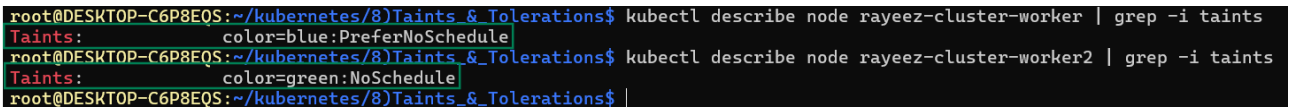
```
apiVersion: v1
kind: Pod
metadata:
  name: yellow-pod
spec:
  tolerations:
    - key: "color"
      operator: "Equal"
      value: "yellow"
      effect: "NoExecute"
  containers:
```

```
    - name: nginx-container
      image: nginx
```

This pod was also deployed on worker node-1, even though its tolerations do not match the worker node-1's taints.



Lets check the taints in both Nodes:

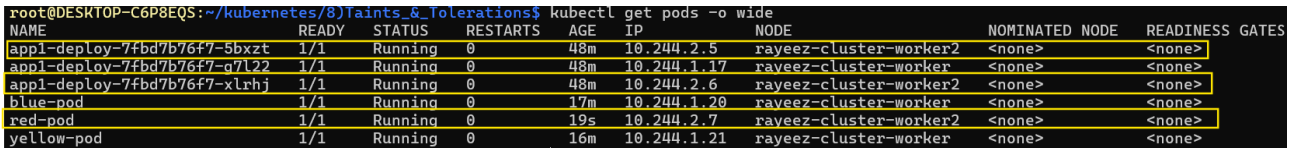


PreferNoSchedule: Kubernetes attempts to avoid scheduling pods, but it does not enforce this behavior as strictly as the **NoSchedule** effect and **NoExecute** effect.

The yellow pod's tolerations do not match the taints on either node. As a result, Worker node-2 did not allow it to schedule due to the **NoSchedule** effect. However, Worker node-1 allowed the pod to run because **PreferNoSchedule** is a soft constraint and does not strictly block scheduling.

Lets remove taint from the worker node-2, list out running pods:

```
kubectl taint node rayeez-cluster-worker2 color=green:NoSchedule-
kubectl get pods -o wide
```



Worker Node 2 has three pods running on it.

Lets apply a taint on worker node-2 with **NoExecute** effect:

```
kubectl taint node rayeez-cluster-worker2 colour=purple:NoExecute
```

→ Instantly, Pods on the worker node2(with NoExecute effect) are termianted and recreated on worker node-1 (With PreferNoSchedule effect).

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE   IP            NODE                     NOMINATED NODE   READINESS GATES
app1-deploy-7fbd7b76f7-4xhw6  1/1     Running   0          10s   10.244.1.22   rayeez-cluster-worker    <none>           <none>
app1-deploy-7fbd7b76f7-g7l22  1/1     Running   0          54m   10.244.1.17   rayeez-cluster-worker    <none>           <none>
app1-deploy-7fbd7b76f7-psmjk  1/1     Running   0          10s   10.244.1.23   rayeez-cluster-worker    <none>           <none>
blue-pod                      1/1     Running   0          23m   10.244.1.20   rayeez-cluster-worker    <none>           <none>
yellow-pod                    1/1     Running   0          21m   10.244.1.21   rayeez-cluster-worker    <none>           <none>
```

NoExecute: Existing pods without a matching toleration will be evicted from the node.

Lets deploy a pod on Worker node-2 with matching Tolerations:

```
apiVersion: v1
kind: Pod
metadata:
  name: purple-pod
spec:
  tolerations:
    - key: "colour"
      operator: "Equal"
      value: "purple"
      effect: "NoExecute"
  containers:
    - name: nginx-container
      image: nginx
```

Apply;

```
kubectl apply -f purple-pod.yaml
```

```
root@DESKTOP-C6P8EQS:~/kubernetes/8)Taints_&_Tolerations$ kubectl get pods -o wide
NAME                          READY   STATUS    RESTARTS   AGE     IP            NODE                     NOMINATED NODE   READINESS GATES
app1-deploy-7fbd7b76f7-4xhw6  1/1     Running   0          9m40s   10.244.1.22   rayeez-cluster-worker    <none>           <none>
app1-deploy-7fbd7b76f7-g7l22  1/1     Running   0          63m     10.244.1.17   rayeez-cluster-worker    <none>           <none>
app1-deploy-7fbd7b76f7-psmjk  1/1     Running   0          9m40s   10.244.1.23   rayeez-cluster-worker    <none>           <none>
blue-pod                      1/1     Running   0          32m     10.244.1.20   rayeez-cluster-worker    <none>           <none>
purple-pod                    1/1     Running   0          8s      10.244.2.8    rayeez-cluster-worker2   <none>           <none>
yellow-pod                    1/1     Running   0          31m     10.244.1.21   rayeez-cluster-worker    <none>           <none>
```