✅ Mastering Taints & Tolerations in Kubernetes

🎯 **Objective**:
Control pod placement in a Kubernetes cluster using **taints** and **tolerations** to ensure only specific pods run on specific nodes.

🧱 Step 1: Taint Worker Nodes

📘 Definitions

- **Taint**: A taint is applied to a node and prevents pods from being scheduled on it unless the pod has a matching toleration. It helps repel pods from certain nodes.

- **Toleration**: A toleration is applied to a pod and allows (but does not require) the pod to schedule onto nodes with matching taints.

Together, taints and tolerations work to control which pods can run on which nodes.

Prevent general pods from being scheduled on specific nodes:

```
kubectl taint nodes worker01 gpu=true:NoSchedule
kubectl taint nodes worker02 gpu=false:NoSchedule
```

```
controlplane ~ ➜ kubectl taint nodes node01 gpu=true:NoSchedule
node/node01 tainted

controlplane ~ ➜ kubectl taint nodes node02 gpu=false:NoSchedule
node/node02 tainted

controlplane ~ ➜ kubectl describe node node01 | grep Taint
Taints:             gpu=true:NoSchedule

controlplane ~ ➜ kubectl describe node node02 | grep Taint
Taints:             gpu=false:NoSchedule

controlplane ~ ➜ sudo vi nginx-tainted.yaml

controlplane ~ ➜ kubectl apply -f nginx-tainted.yaml
pod/nginx-tainted created

controlplane ~ ➜ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
nginx-tainted   0/1     Pending   0          16s
```

🔍 **Verify Taints:**

```
kubectl describe node worker01 | grep Taint
```

```
controlplane ~ ✖ kubectl describe pod nginx-tainted
Name:             nginx-tainted
Namespace:        default
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  nginx:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6z9tw (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
```

🐳 Step 2: Create Pod Without Toleration

➡️ Create a pod without toleration, which **should remain Pending**:

`nginx-tainted.yaml`:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-tainted
spec:
  containers:
  - name: nginx
    image: nginx
```

```
kubectl apply -f nginx-tainted.yaml
kubectl describe pod nginx-tainted
```

🛑 Pod stays in `Pending` state due to taint!

🧩 Step 3: Add Toleration to Pod

➡️ Allow scheduling on `worker01` by adding a toleration:

```
spec:
  tolerations:
  - key: "gpu"
    operator: "Equal"
    value: "true"
    effect: "NoSchedule"
```

🛠️ Update `nginx-tainted.yaml` and reapply:

```
kubectl delete pod nginx-tainted
kubectl apply -f nginx-tainted.yaml
kubectl get pod nginx-tainted -o wide
```

```
controlplane ~ ➜  kubectl apply -f nginx-tainted.yaml
pod/nginx-tainted created

controlplane ~ ➜  kubectl get pods -o wide
NAME            READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   REA
DINESS GATES
nginx-tainted   1/1     Running   0          11s   172.17.1.2   node01   <none>           <no
ne>

controlplane ~ ➜  kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   47m   v1.33.0
node01         Ready    <none>          46m   v1.33.0
node02         Ready    <none>          46m   v1.33.0
```

✅ Pod now runs on `worker01`.

⚙️ Step 4: Remove Control Plane Taint

➡️ Allow pod scheduling on control plane:

```
kubectl taint nodes controlplane node-role.kubernetes.io/control-plane-
```

🗃️ Step 5: Deploy Redis Pod to Control Plane

➡️ Create `redis-on-control.yaml`:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis-on-control
spec:
  containers:
  - name: redis
    image: redis
```

```
controlplane ~ ➜ kubectl apply -f redis-on-control.yaml
pod/redis-on-control unchanged

controlplane ~ ➜ kubectl get pod redis-on-control -o wide
NAME               READY   STATUS    RESTARTS   AGE     IP           NODE           NOMINATE
 NODE   READINESS GATES
redis-on-control   1/1     Running   0          6m38s   172.17.0.5   controlplane   <none>
        <none>

controlplane ~ ➜ ^C

controlplane ~ ✖ kubectl taint nodes controlplane node-role.kubernetes.io/control-plane-
error: taint "node-role.kubernetes.io/control-plane" not found

controlplane ~ ✖ kubectl taint nodes controlplane node-role.kubernetes.io/control-plane=:NoS
hedule
node/controlplane tainted
```

✅ Pod scheduled successfully on `controlplane`.

🔁 Step 6: Reapply Control Plane Taint

➡️ Prevent further scheduling on control plane again:

```
kubectl taint nodes controlplane
node-role.kubernetes.io/control-plane=:NoSchedule --overwrite
```

🎉 Task Completed Successfully!