Taints are for nodes & Taint ⇒ nodes

Tolerations are for pods Tolerations ⇒ Pods

[Taints & toleration is a concept where the node has to accept certain pods based on pod's toleration.
⇒ In K8S taints and tolerations work together to control which pods can be scheduled on specific nodes.

⇒ Taints are applied to nodes to indicate that they should not accept certain pods unless those pods have a matching toleration.

→ Tolerations, on the other hand, are added to pods to allow them to be scheduled on nodes with specific taints.

Taints:
⇒ Taints are applied to nodes to restrict scheduling. They're essentially "locks" that repel pods that don't have the appropriate "key" (toleration)

⇒ Taints have 3 parts: a key, an optional value, and an effect.
Kubectl taint nodes <node> key=value: NoSchedule Block new pods
Kubectl taint nodes <node> key=value: NoExecute (Effect) Evict existing pods
⇒ The effect determines how the taint affects scheduling:

* NoSchedule:
The K8S Scheduler will not schedule pods that don't have a toleration for the taint.
If the toleration in the pod doesn't match with the taint in the node, then scheduler will not schedule the pod.
* PreferNoSchedule:
The Scheduler will try to avoid scheduling pods without a toleration, but it might do so if there are no other suitable nodes available.
[The system will try to avoid placing the pod but it is not guaranteed] if the toleration in the pod doesn't match the taint in the node, then there can be chances that scheduler may or may not place the pod

NoExecute

If a pod without a matching toleration is already running on the node, it will be evicted.

Tolerations:

=> Tolerations are applied to pods to allow them to be scheduled on nodes with matching taints

=> they act as the "key" that unlocks the taint on the node.

=> Tolerations also have 3 parts:
a key, an optional value, and an effect

=> The effect of a toleration must match the effect of the taint it tolerates

In essence, taints and tolerations work like this

1) A node is tainted with a specific taint (key, value, effect)

2) Pods that want to be scheduled on that node must have a toleration that matches the taint's key, value and effect.

3) If a pod doesnot have the correct toleration, it will be repelled from the tainted node by the scheduler.

4) If a pod does have the correct toleration, it can be scheduled on the tainted node.

Tainting the node:

* kubectl taint nodes node-name key=value : taint-effect

NoSchedule | preferNoSchedule | NoExecute

kubectl taint nodes node1 app=blue : NoSchedule

(key, operator, value, effect)

If the taint in the node & toleration in the pods matches then the pod will be scheduled to node 1, if not then the pod will not get scheduled.

Adding Toleration to a pod

To add toleration to a pod, first pull the pod-definition file, under the spec section, add a section called "toleration" move the same values used while creating the taint

accept = taints & tolerations

Restrict = node affinity

```
apiVersion:
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: nginx-container
    image: nginx

  tolerations:
  - key: "app"
    operator: "Equal"
    value: "blue"
    effect: "NoSchedule"
```

NOTE :- (**)

Taints & Tolerations are only meant to restrict nodes from accepting certain pods

(*) ⇒ Taints and Tolerations doesn't tell the pod to go to a particular node, Instead, it tells the node to only accept pods with certain tolerations

⇒ If your requirement is to restrict a pod to a certain nodes, it is achieved through another concept called node affinity

INTERSTING FACT

The Scheduler doesn't schedule any pods on master node, why is that?

When K8S cluster is first setup, a taint is set on the master node automatically that prevents any pods from being scheduled on this node [we can view & change this behavior]

To view :-

```
Kubectl describe node kubemaster | grep Taint
```

(nodename)

Taints: node-role.kubernetes.io/master: NoSchedule

Add Taint

```
Kubectl taint nodes <node> <key>:<effect>
```

(<key>=<value>:<effect>)

Eg:-

```
kubectl taint nodes node1 app=prod
```

Remove Taint :- To untaint a node, append - after the effect

```
Kubectl taint nodes <node> <key>:<effect>-
```

Eg:-

```
kubectl taint nodes node1 app=prod-
```