# **<https://docs.openshift.com/container-platform/4.6/openshift_images/managing_images/image-pull-policy.html>**

# **Image pull policy**

* [**Image pull policy overview**](https://docs.openshift.com/container-platform/4.6/openshift_images/managing_images/image-pull-policy.html#images-image-pull-policy-overview_image-pull-policy)

Each container in a pod has a container image. Once you have created an image and pushed it to a registry, you can then refer to it in the pod.

## Image pull policy overview

When OpenShift Container Platform creates containers, it uses the container imagePullPolicy to determine if the image should be pulled prior to starting the container. There are three possible values for imagePullPolicy:

| *Table 1. imagePullPolicy values* | |
| --- | --- |
| **Value** | **Description** |
| Always | Always pull the image. |
| IfNotPresent | Only pull the image if it does not already exist on the node. |
| Never | Never pull the image. |

If a container imagePullPolicy parameter is not specified, OpenShift Container Platform sets it based on the image tag:

1. If the tag is latest, OpenShift Container Platform defaults imagePullPolicy to Always.
2. Otherwise, OpenShift Container Platform defaults imagePullPolicy to IfNotPresent.

[**https://kubernetes.io/docs/concepts/containers/images/**](https://kubernetes.io/docs/concepts/containers/images/)

[**https://www.containiq.com/post/kubernetes-image-pull-policy**](https://www.containiq.com/post/kubernetes-image-pull-policy)

Taint

Taints are **a property of nodes**that push pods away if they don't tolerate this taint. Like Labels, one or more Taints can be **applied to a node. This** means that the node must not accept any pod that does not tolerate all of these taints.

Toleration

Tolerations are **applied to pods, and allow the pods to schedule onto nodes with matching taints.**

Taint and toleration

Taints and tolerations works 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.

Taint is like: **key=value:Effect**

You can assign **3 different values**to “effect:

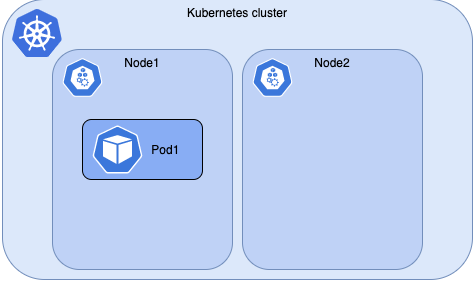
* **NoSchedule**: if this taint is applied to a node which contains already some pod that doesn’t tolerate this taint, they are not excluded from this node. But no more pods are scheduled on this node if it doesn’t match all the taints of this node. This is a strong constraint.
* **PreferNoSchedule:**Like the previous one, this taint may not allow pods to be scheduled on the node. But this time, if the pod tolerates one taint, it can be scheduled. This is a soft constraint.
* **NoExecute**: This taint applies to a node excluding all actual running pods on it and doesn’t allow scheduling if new pods don’t tolerate all taint. This is a strong constraint.

The default value for the operator is “**Equal**. But it can also be “**Exist**.

* Equal : **key=value:Effect**
* Exist**: keyExist:Effect**

Which means that if you change the operator by “exist you don’t have to give a value. If the taint isn’t present on the node, “effect is applied to the pod.

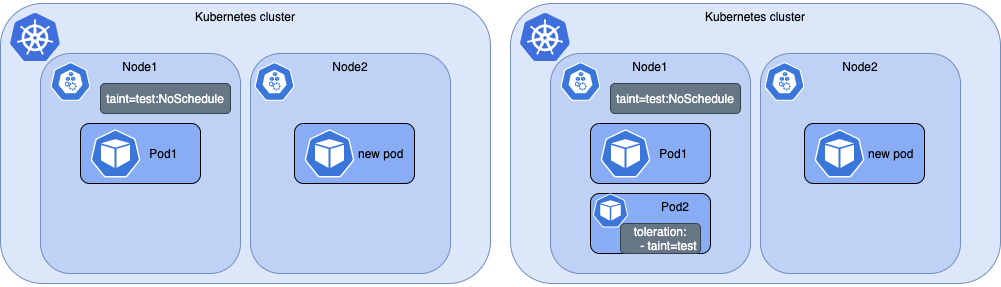
Let's illustrate with schemas. First, when there is no taint, **pod can be affected to any node**(here Pod1 is attached to Node1, but it could also be on Node2).



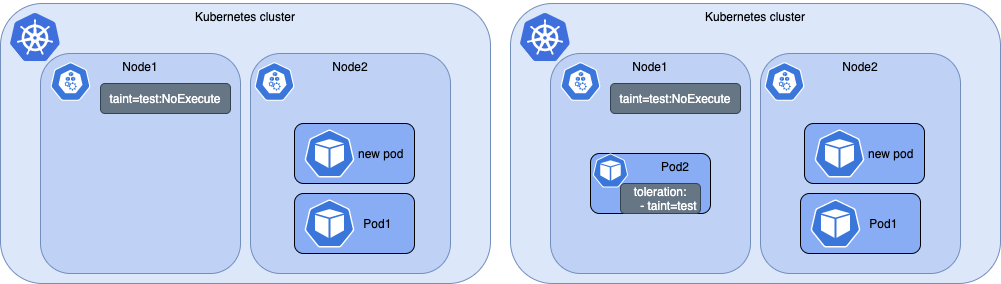
<schema-without-taint>

Now, add a taint on the node: **taint=test:NoSchedule**

This means if the pod doesn't match this taint, it can’t be scheduled on this node. You can see on the schema below, “new pod doesn’t have toleration, so it has been affected on Node2. But Pod2, which has toleration that match the taint, can be affected on Node1.

  
<schema-with-noschedule-taint>

Now, if we **add a taint on the node:  
taint=test:NoExecuteIt will exclude all pods which have toleration that doesn’t match the taint.**



<schema-with-noexecute-taint>

**Deploy node with a taint**

To add a taint to an existing node, you can run the following command:

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

Example: $ kubectl taint nodes node-main taint=test:PrefereNoSchedule

To show taint of your node you can run this command:

$ kubectl describe node

Name: node-main

Roles: <none>

Labels: [...]

Annotations: [...]

CreationTimestamp: Sat, 17 Apr 2021 04:38:19 +0200

Taints: taint=test:PreferNoSchedule

[...]

**Deploy a pod with a specific node**

Now that your nodes are tainted, here is an example to add tolerations to your pod:

tolerations:

key: "key1"

operator: "Equal"

value: "value1"

Example :

|  |  |
| --- | --- |
|  | apiVersion: v1 |
|  | kind: Pod |
|  | metadata: |
|  | name: nginx-toleration |
|  | labels: |
|  | env: test |
|  | spec: |
|  | containers: |
|  | - name: nginx |
|  | image: nginx |
|  | imagePullPolicy: IfNotPresent |
|  | tolerations: |
|  | - key: "taint" |
|  | value: "test" |
|  | effect: "PrefereNoSchedule" |

# **Working with Multiple Taints and Tolerations**

Kubernetes users can set multiple taints on nodes. The process of matching tolerations with these taints then works as a filter. The system will ignore those taints for which the tolerations exist and make un-ignored taints have the indicated effect on the Pod.

Let’s illustrate how this works by applying several taints to our node:

kubectl taint nodes host1 key1=value1:NoSchedule  
kubectl taint nodes host1 key1=value1:NoExecute  
kubectl taint nodes host1 key2=value2:NoSchedule

Next, let’s specify two tolerations in the Pod:

apiVersion: v1  
kind: Pod  
metadata:  
 name: pod-2  
 labels:  
 security: s1  
spec:  
 containers:  
 - name: bear  
 image: supergiantkir/animals:bear  
 tolerations:  
 - key: "key1"  
 operator: "Equal"  
 value: "value1"  
 effect: "NoSchedule"  
 - key: "key1"  
 operator: "Equal"  
 value: "value1"  
 effect: "NoExecute"

This Pod tolerates the first and the second taint but does not tolerate the third taint with the effect NoSchedule . Even though the Pod has two matching tolerations, it won’t be scheduled onto the host01 . The Pod, however, will continue running on this node if it was scheduled before the taint is added because it tolerates the NoExecute effect.

# **NoExecute Effect**

The taint with the NoExecute effect results in the eviction of all Pods without a matching toleration from the node. When using the toleration for the NoExecute effect you can also specify an optional tolerationSeconds field. Its value defines how long the Pod that tolerates the taint can run on that node before eviction and after the taint is added. Let’s look at the manifest below:

apiVersion: v1  
kind: Pod  
metadata:  
 name: pod-2  
 labels:  
 security: s1  
spec:  
 containers:  
 - name: bear  
 image: supergiantkir/animals:bear  
 tolerations:  
 - key: "key1"  
 operator: "Equal"  
 value: "value1"  
 effect: "NoExecute"  
 tolerationSeconds: 3600

If this Pod is running and a matching taint is added to the node, it will stay bound to the node for 3600 seconds. If the taint is removed by that time, the Pod won’t be evicted.

In general, the following rules apply for the NoExecute effect:

* Pods with no tolerations for the taint(s) are evicted immediately.
* Pods with the toleration for the taint but that do not specify tolerationSeconds in their toleration stay bound to the node forever.
* Pods that tolerate the taint with a specified tolerationSeconds remain bound for the specified amount of time.

<https://kubernetes.io/docs/concepts/scheduling-eviction/taint-and-toleration/>

<https://www.densify.com/kubernetes-autoscaling/kubernetes-taints>

<https://www.geeksforgeeks.org/kubernetes-taint-and-toleration/>

<https://www.padok.fr/en/blog/add-taint-nodes-tolerations>

<https://medium.com/kubernetes-tutorials/making-sense-of-taints-and-tolerations-in-kubernetes-446e75010f4e>

<https://www.containiq.com/post/kubernetes-taints-and-tolerations>