What is Node taints and tolerations ?

Kubernetes - Beginners | Intermediate | Advanced

View on GitHub Join Slack

What is Node taints and tolerations ?

- This Kubernetes feature allows users to mark a node (taint the node) so that no pods can be scheduled to it, unless a pod explicitly tolerates the taint.
- When you taint a node, it is automatically excluded from pod scheduling. When the schedule runs the predicate tests on a tainted node, they'll fail unless the pod has toleration for that node.

- Like last monitoring example: Let assume new member joins the development team, writes a Deployment for her application, but forgets to exclude the monitoring nodes from the target nodes? Kubernetes administrators need a way to repel pods from nodes without having to modify every pod definition.

Steps

```
git clone https://github.com/collabnix/dockerlabs
cd dockerlabs/kubernetes/workshop/Scheduler101/
kubectl label nodes node2 role=dev
```

```
kubectl label nodes node3 role=dev

[node1 Scheduler101]$ kubectl taint nodes node2 role=dev:NoSchedule
node/node2 tainted
[node1 Scheduler101]$


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

Viewing Your Pods

```
kubectl get pods --output=wide
```

Get nodes label detail

```
[node1 Scheduler101]$ kubectl get nodes --show-labels|grep mynode |grep role
node2   Ready    <none>   175m   v1.14.9   beta.kubernetes.io/arch=amd64,bet
node3   Ready    <none>   175m   v1.14.9   beta.kubernetes.io/arch=amd64,bet
```

Get pod describe

```
[node1 Scheduler101]$ kubectl describe pods nginx
Name:               nginx
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               node3/192.168.0.16
Start Time:         Mon, 30 Dec 2019 19:13:45 +0000
Labels:             <none>
Annotations:        kubectl.kubernetes.io/last-applied-configuration:
                      {"apiVersion":"v1","kind":"Pod","metadata":{"annotatio
Status:             Running
IP:                 10.36.0.1
```

```
Containers:
  nginx:
    Container ID:   docker://57d032f4358be89e2fcad7536992b175503565af82ce4f6
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:b2d89d0a210398b4d1120b3e3
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Mon, 30 Dec 2019 19:14:45 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-qpgxq
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-qpgxq:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-qpgxq
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  105s  default-scheduler  Successfully assigned default/
  Normal  Pulling    101s  kubelet, node3     Pulling image "nginx"
  Normal  Pulled     57s   kubelet, node3     Successfully pulled image "ngi
  Normal  Created    47s   kubelet, node3     Created container nginx
  Normal  Started    45s   kubelet, node3     Started container nginx
```

- Deployed pod on node3.

Step Cleanup

Finally you can clean up the resources you created in your cluster:

```
kubectl delete -f pod-tain-node.yaml
```

Tolerations

- A toleration is a way of ignoring a taint during scheduling. Tolerations aren't applied to nodes, but rather the pods. So, in the example above, if we apply a toleration to the PodSpec, we could "tolerate" the slow disks on that node and still use it.

Steps

```
git clone https://github.com/collabnix/dockerlabs
cd dockerlabs/kubernetes/workshop/Scheduler101/
kubectl apply -f pod-tolerations-node.yaml
```

Viewing Your Pods

```
kubectl get pods --output=wide
```

Which Node Is This Pod Running On?

```
[node1 Scheduler101]$ kubectl describe pods nginx
Name:               nginx
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               node3/192.168.0.16
Start Time:         Mon, 30 Dec 2019 19:20:35 +0000
```

```
Labels:             env=test
Annotations:        kubectl.kubernetes.io/last-applied-configuration:
                      {"apiVersion":"v1","kind":"Pod","metadata":{"annotatio
Status:             Pending
IP:
Containers:
  nginx:
    Container ID:
    Image:          nginx:1.7.9
    Image ID:
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       ContainerCreating
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-qpgxq
Conditions:
  Type              Status
  Initialized       True
  Ready             False
  ContainersReady   False
  PodScheduled      True
Volumes:
  default-token-qpgxq:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-qpgxq
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
                 role=dev:NoSchedule
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  4s    default-scheduler  Successfully assigned default/
  Normal  Pulling    1s    kubelet, node3     Pulling image "nginx:1.7.9
```

Step Cleanup

Finally you can clean up the resources you created in your cluster:

```
kubectl delete -f pod-tolerations-node.yaml
```

- An important thing to notice, though, is that tolerations may enable a tainted node to accept a pod but it does not guarantee that this pod runs on that specific node.
- In other words, the tainted node will be considered as one of the candidates for running our pod. However, if another node has a higher priority score, it will be chosen instead. For situations like this, you need to combine the toleration with nodeSelector or node affinity parameters.

Next »

---

kubelabs is maintained by collabnix.

This page was generated by GitHub Pages.