### Taint和Toleration

Taint能够使节点排斥一类特定的Pod，Taint和Toleration相互配合可以用来避免Pod被分配到不合适的节点，比如Master节点不允许部署系统组件之外的其他Pod。每个节点上都可以应用一个或多个Taint，这表示对于那些不能容忍这些Taint的Pod是不会被该节点接受的。如果将Toleration应用于Pod上，则表示这些Pod可以（但不要求）被调度到具有匹配Taint的节点上。

1. 概念

给节点增加一个Taint：

[root@k8s-master01 2.2.8]# kubectl taint nodes k8s-node01 key=value:NoSchedule

node/k8s-node01 tainted

上述命令给k8s-node01增加一个Taint，它的key对应的就是键，value对应就是值，effect对应的就是NoSchedule。这表明只有和这个Taint相匹配的Toleration的Pod才能够被分配到k8s-node01节点上。按如下方式在PodSpec中定义Pod的Toleration，就可以将Pod部署到该节点上。

方式一：

tolerations:

- key: "key"

operator: "Equal"

value: "value"

effect: "NoSchedule"

方式二：

tolerations:

- key: "key"

operator: "Exists"

effect: "NoSchedule"

一个Toleration和一个Taint相匹配是指它们有一样的key和effect，并且如果operator是Exists（此时toleration不指定value）或者operator是Equal，则它们的value应该相等。

注意两种情况：

* 如果一个Toleration的key为空且operator为Exists，表示这个Toleration与任意的key、value和effect都匹配，即这个Toleration能容忍任意的Taint：

tolerations:

- operator: "Exists"

* 如果一个Toleration的effect为空，则key与之相同的相匹配的Taint的effect可以是任意值：

tolerations:

- key: "key"

operator: "Exists"

上述例子使用到effect的一个值NoSchedule，也可以使用PreferNoSchedule，该值定义尽量避免将Pod调度到存在其不能容忍的Taint的节点上，但并不是强制的。effect的值还可以设置为NoExecute。

一个节点可以设置多个Taint，也可以给一个Pod添加多个Toleration。Kubernetes处理多个Taint和Toleration的过程就像一个过滤器：从一个节点的所有Taint开始遍历，过滤掉那些Pod中存在与之相匹配的Toleration的Taint。余下未被过滤的Taint的effect值决定了Pod是否会被分配到该节点，特别是以下情况：

* 如果未被过滤的Taint中存在一个以上effect值为NoSchedule的Taint，则Kubernetes不会将Pod分配到该节点。
* 如果未被过滤的Taint中不存在effect值为NoExecute的Taint，但是存在effect值为PreferNoSchedule的Taint，则Kubernetes会尝试将Pod分配到该节点。
* 如果未被过滤的Taint中存在一个以上effect值为NoExecute的Taint，则Kubernetes不会将Pod分配到该节点（如果Pod还未在节点上运行），或者将Pod从该节点驱逐（如果Pod已经在节点上运行）。

例如，假设给一个节点添加了以下的Taint：

kubectl taint nodes k8s-node01 key1=value1:NoSchedule

kubectl taint nodes k8s-node01 key1=value1:NoExecute

kubectl taint nodes k8s-node01 key2=value2:NoSchedule

然后存在一个Pod，它有两个Toleration：

tolerations:

- key: "key1"

operator: "Equal"

value: "value1"

effect: "NoSchedule"

- key: "key1"

operator: "Equal"

value: "value1"

effect: "NoExecute"

在上述例子中，该Pod不会被分配到上述节点，因为没有匹配第三个Taint。但是如果给节点添加上述3个Taint之前，该Pod已经在上述节点中运行，那么它不会被驱逐，还会继续运行在这个节点上，因为第3个Taint是唯一不能被这个Pod容忍的。

通常情况下，如果给一个节点添加了一个effect值为NoExecute的Taint，则任何不能容忍这个Taint的Pod都会马上被驱逐，任何可以容忍这个Taint的Pod都不会被驱逐。但是，如果Pod存在一个effect值为NoExecute的Toleration指定了可选属性tolerationSeconds的值，则该值表示是在给节点添加了上述Taint之后Pod还能继续在该节点上运行的时间，例如：

tolerations:

- key: "key1"

operator: "Equal"

value: "value1"

effect: "NoExecute"

tolerationSeconds: 3600

表示如果这个Pod正在运行，然后一个匹配的Taint被添加到其所在的节点，那么Pod还将继续在节点上运行3600秒，然后被驱逐。如果在此之前上述Taint被删除了，则Pod不会被驱逐。

删除一个Taint：

kubectl taint nodes k8s-node01 key1:NoExecute-

查看Taint：

[root@k8s-master01 2.2.8]# kubectl describe node k8s-node01 | grep Taint

Taints: key=value:NoSchedule

2. 用例

通过Taint和Toleration可以灵活地让Pod避开某些节点或者将Pod从某些节点被驱逐。下面是几种情况。

（1）专用节点

如果想将某些节点专门分配给特定的一组用户使用，可以给这些节点添加一个Taint（kubectl taint nodes nodename dedicated=groupName:NoSchedule），然后给这组用户的Pod添加一个相对应的Toleration。拥有上述Toleration的Pod就能够被分配到上述专用节点，同时也能够被分配到集群中的其他节点。如果只希望这些Pod只能分配到上述专用节点中，那么还需要给这些专用节点另外添加一个和上述Taint类似的Label（例如：dedicated=groupName），然后给Pod增加节点亲和性要求或者使用NodeSelector，就能将Pod只分配到添加了dedicated=groupName标签的节点上。

（2）特殊硬件的节点

在部分节点上配备了特殊硬件（比如GPU）的集群中，我们只允许特定的Pod才能部署在这些节点上。这时可以使用Taint进行控制，添加Taint如kubectl taint nodes nodename special=true:NoSchedule或者kubectl taint nodes nodename special=true:PreferNoSchedule，然后给需要部署在这些节点上的Pod添加相匹配的Toleration即可。

（3）基于Taint的驱逐

属于alpha特性，在每个Pod中配置在节点出现问题时的驱逐行为。

3. 基于Taint的驱逐

之前提到过Taint的effect值NoExecute，它会影响已经在节点上运行的Pod。如果Pod不能忍受effect值为NoExecute的Taint，那么Pod将会被马上驱逐。如果能够忍受effect值为NoExecute的Taint，但是在Toleration定义中没有指定tolerationSeconds，则Pod还会一直在这个节点上运行。

在Kubernetes 1.6版以后已经支持（alpha）当某种条件为真时，Node Controller会自动给节点添加一个Taint，用以表示节点的问题。当前内置的Taint包括：

* node.kubernetes.io/not-ready：节点未准备好，相当于节点状态Ready的值为False。
* node.kubernetes.io/unreachable：Node Controller访问不到节点，相当于节点状态Ready的值为Unknown。
* node.kubernetes.io/out-of-disk：节点磁盘耗尽。
* node.kubernetes.io/memory-pressure：节点存在内存压力。
* node.kubernetes.io/disk-pressure：节点存在磁盘压力。
* node.kubernetes.io/network-unavailable：节点网络不可达。
* node.kubernetes.io/unschedulable：节点不可调度。
* node.cloudprovider.kubernetes.io/uninitialized：如果Kubelet启动时指定了一个外部的cloudprovider，它将给当前节点添加一个Taint将其标记为不可用。在cloud-controller-manager的一个controller初始化这个节点后，Kubelet将删除这个Taint。

使用这个alpha功能特性，结合tolerationSeconds，Pod就可以指定当节点出现一个或全部上述问题时，Pod还能在这个节点上运行多长时间。

比如，一个使用了很多本地状态的应用程序在网络断开时，仍然希望停留在当前节点上运行一段时间，愿意等待网络恢复以避免被驱逐。在这种情况下，Pod的Toleration可以这样配置：

tolerations:

- key: "node.alpha.kubernetes.io/unreachable"

operator: "Exists"

effect: "NoExecute"

tolerationSeconds: 6000

注 意

Kubernetes会自动给Pod添加一个key为node.kubernetes.io/not-ready的Toleration并配置tolerationSeconds=300，同样也会给Pod添加一个key为node.kubernetes.io/unreachable的Toleration并配置tolerationSeconds=300，除非用户自定义了上述key，否则会采用这个默认设置。

这种自动添加Toleration的机制保证了在其中一种问题被检测到时，Pod默认能够继续停留在当前节点运行5分钟。这两个默认Toleration是由DefaultTolerationSeconds admission controller添加的。

DaemonSet中的Pod被创建时，针对以下Taint自动添加的NoExecute的Toleration将不会指定tolerationSeconds：

* node.alpha.kubernetes.io/unreachable
* node.kubernetes.io/not-ready

这保证了出现上述问题时DaemonSet中的Pod永远不会被驱逐。