# K8s资源

《Kubernetes 加固指南》中文版

<https://jimmysong.io/kubernetes-hardening-guidance/>

<https://jimmysong.io/kubernetes-handbook/concepts/scheduling.html>

# **Kubernetes集群中，Node异常时Pod状态分析**

<https://www.huaweicloud.com/articles/13573392.html>

# **[k8s pod节点调度及k8s资源优化](https://www.cnblogs.com/xiaoyaojinzhazhadehangcheng/p/11065510.html)**

<https://www.cnblogs.com/xiaoyaojinzhazhadehangcheng/p/11065510.html>

<https://www.cnblogs.com/baozexu/p/13705680.html>

一、k8s pod 在节点间调度控制

k8s起pod时，会通过调度器scheduler选择某个节点完成调度，选择在某个节点上完成pod创建。  
当需要在指定pod运行在某个节点上时，可以通过以下几种方式：

1. 在部署pod的yaml中，指定nodeName

指定了nodeName的Pod会直接跳过Scheduler的调度逻辑，直接写入PodList列表，该匹配规则是强制匹配。  
eg:  
apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
name: tomcat-deploy  
spec:  
replicas: 1  
template:  
metadata:  
labels:  
app: tomcat-app  
spec:  
 nodeName: k8s.node1 #指定调度节点为k8s.node1  
containers:  
- name: tomcat  
image: tomcat:8.0  
ports:  
- containerPort: 8080

2. 在部署pod的yaml中，指定NodeSelector  
通过kubernetes的label-selector机制进行节点选择，由scheduler调度策略MatchNodeSelector进行label匹配，调度pod到目标节点，该匹配规则是强制约束。启用节点选择器的步骤为：  
起pod步骤：

i. 先给node打上对应的label  
#标记规则：kubectl label nodes <node-name> <label-key>=<label-value>  
kubectl label nodes k8s.node1 cloudnil.com/role=dev  
  
#确认标记  
root@k8s.master1:~# kubectl get nodes k8s.node1 --show-labels  
NAME STATUS AGE LABELS  
k8s.node1 Ready 29d beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,cloudnil.com/role=dev,kubernetes.io/hostname=k8s.node1  
  
ii. 起pod的时候，在yaml中指定nodeSelector

apiVersion: extensions/v1beta1  
kind: Deployment  
metadata:  
name: tomcat-deploy  
spec:  
replicas: 1  
template:  
metadata:  
labels:  
app: tomcat-app  
spec:  
nodeSelector:  
cloudnil.com/role: dev #指定调度节点为带有label标记为：cloudnil.com/role=dev的node节点  
containers:  
- name: tomcat  
image: tomcat:8.0  
ports:  
- containerPort: 8080

3. 使用污点(Taint)  
taint是k8s 1.6版本开始提供的功能，在k8s同样用于节点调度的，还有一个容忍(Tolerations)功能  
污点的组成为  
key=value:effect

effect支持的类型包括一下三类：  
NoSchedule：表示k8s将不会将Pod调度到具有该污点的Node上  
PreferNoSchedule：表示k8s将尽量避免将Pod调度到具有该污点的Node上  
NoExecute：表示k8s将不会将Pod调度到具有该污点的Node上，同时会将Node上已经存在的Pod驱逐出去  
当希望起pod的时候，某个node不参与调度，可以给这个node加上一个NoSchedule的污点，eg：

# 设置污点  
kubectl taint nodes kube-node1 key1=value1:NoSchedule  
  
# 去除污点  
kubectl taint nodes kube-node1 key1:NoSchedule-  
此污点可以通过kubectl describe node <xxx-node> 查看

4. 容忍(tolerance)  
当参与taint给node打算NoSchedule的污点之后，pod无法调度到该节点上运行，但有时我们希望某些pod仍然能运行到该node上时，可以通过在部署pod的yaml中添加tolerance，使pod能跳过污点，调度到对应节点上：

eg:  
tolerations:  
- key: "key1"  
operator: "Equal"  
value: "value1"  
effect: "NoSchedule"  
tolerationSeconds: 3600  
- key: "key1"  
operator: "Equal"  
value: "value1"  
effect: "NoExecute"  
- key: "key2"  
operator: "Exists"  
effect: "NoSchedule"

其中：  
operator为Equal时，key, vaule, effect要与Node上设置的taint保持一致  
operator为Exist时，将会忽略value的值  
tolerationSeconds指当pod需要被驱逐时，仍能在node上保留运行的时间  
当tolerance不指定effect时，将容忍污点的所有作用  
tolerations:  
- key: "key"  
operator: "Exists"

5. cordon/uncordon/drain  
这几个命令用于k8s节点维护，用法：  
kubectl cordon <xxx-node>  
此时通过

kubectl get node -o wide  
可以看到node的状态为scheduleDisable，即该节点为不可调度状态。

可以通过uncordon来恢复node状态：  
kubectl uncordon <xxx-node>  
当cordon node之后，原来运行在node上的pod，当超过保活周期之后，将会重新调度在其他node上运行。如果想让pod马上实现迁移，可以使用drain完成。

drain:标记节点为不可调度,并驱逐节点上的pod，用法：  
kubectl drain <xxx-node>  
注意：drain不只是当node被cordon时才适用，其实当node正常运行时，也可以用drain将运行在该node的pod驱赶出去

二、k8s(node节点资源)优化设置

在k8s节点中通过ExecStart=/usr/bin/kubelet设置系统保留资源来提高node节点的稳定行。

涉及到的参数：1.--system-reserved  
                  2.--kube-reserved

vim /etc/systemd/system/kubelet.service.d/10-kubeadm.conf  
添加参数：  
1：设置预留系统服务的资源  
--system-reserved=cpu=200m,memory=1G

2:设置预留给k8s组件的资源(主要组件)  
--kube-reserved=cpu=200m,memory=1G  
系统内存--sytem-reserved --kube-reserved就是可以分配给pod的内存

3.驱逐条件  
--eviction-hard=memory.available<500Mi,nodefs.available<1Gi,imagefsavailable<100Gi

4.最小驱逐  
--eviction-minimum-reclaim="memory.available<0Mi,nodefs.available<500Mi,imagefsavailable<2Gi"

5.节点状态更新时间  
--node-status-update-frequency=10s

6.驱逐等待时间  
--eviction-pressure-transition-period=20s

## 选择节点（nodeSelector）

<https://www.kubernetes.org.cn/2614.html>

## 亲和性（Affinity）与非亲和性（anti-affinity）

前面提及的nodeSelector，其仅以一种非常简单的方式、即label强制限制pod调度到指定节点。而亲和性（Affinity）与非亲和性（anti-affinity）则更加灵活的指定pod调度到预期节点上，相比nodeSelector，Affinity与anti-affinity优势体现在：

* 表述语法更加多样化，不再仅受限于强制约束与匹配。
* 调度规则不再是强制约束（hard），取而代之的是软限（soft）或偏好（preference）。
* 指定pod可以和哪些pod部署在同一个/不同拓扑结构下。

亲和性主要分为3种类型：node affinity与inter-pod affinity/anti-affinity，下文会进行详细说明。

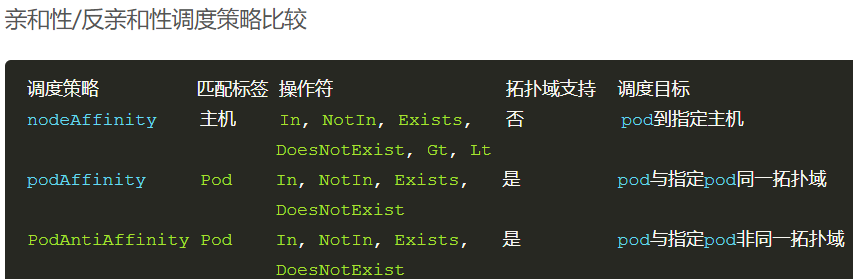
## 节点亲和性（Node affinity）

Node affinity在Kubernetes 1.2做为alpha引入，其涵盖了nodeSelector功能，主要分为requiredDuringSchedulingIgnoredDuringExecution与preferredDuringSchedulingIgnoredDuringExecution 2种类型。前者可认为一种强制限制，如果 Node 的标签发生了变化导致其没有符合 Pod 的调度要求节点，那么pod调度就会失败。而后者可认为理解为软限或偏好，同样如果 Node 的标签发生了变化导致其不再符合 pod 的调度要求，pod 依然会调度运行。

## pod亲和性（Inter-pod affinity）与反亲和性（anti-affinity）

inter-pod affinity与anti-affinity由Kubernetes 1.4引入，当前处于beta阶段，其中podAffinity用于调度pod可以和哪些pod部署在同一拓扑结构之下。而podAntiAffinity相反，其用于规定pod不可以和哪些pod部署在同一拓扑结构下。通过pod affinity与anti-affinity来解决pod和pod之间的关系。  
与Node affinity类似，pod affinity与anti-affinity同样分为requiredDuringSchedulingIgnoredDuringExecution and preferredDuringSchedulingIgnoredDuringExecution等2种类型，

前者被认为是强制约束，而后者后者可认为理解软限（soft）或偏好（preference）。



## Marking pod as critical

<https://kubernetes.io/docs/tasks/administer-cluster/guaranteed-scheduling-critical-addon-pods/>

Todo

# **Pod 拓扑分布约束**

<https://kubernetes.io/zh/docs/concepts/workloads/pods/pod-topology-spread-constraints/>

<https://kubernetes.io/blog/2020/05/introducing-podtopologyspread/>

Todo