拉勾教育
— 互联网人实战大学 —

《 Kubernetes 原理剖析与实战应用》

正范

— 拉勾教育出品 —

21 | 优先级调度：你必须掌握的 Pod 抢占式资源调度

前言

当集群资源比较紧张时，如果此时还要部署一些比较重要的关键业务

如何去提前“抢占”集群资源，使得关键业务在集群中跑起来？

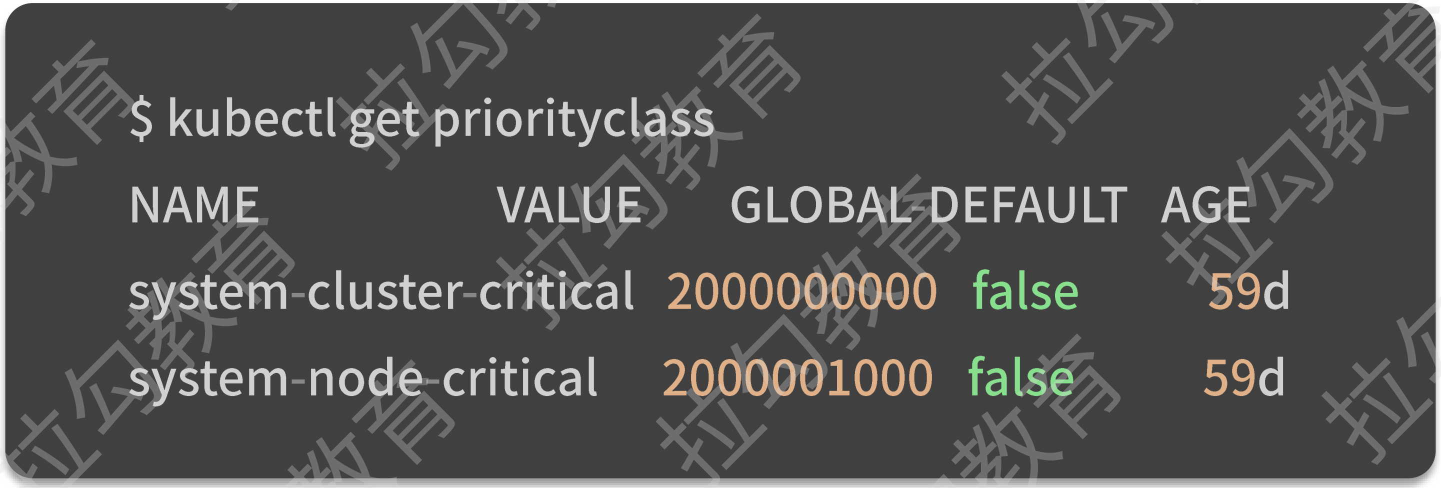
PriorityClass

```yaml
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority
value: 1000000
globalDefault: false
description: "This priority class should be used for XYZ service pods only."
```

PriorityClass

```
// HighestUserDefinablePriority is the highest priority for user defined priority classes.
Priority values larger than 1 billion are reserved for Kubernetes system use.
HighestUserDefinablePriority = int32(1000000000)
// SystemCriticalPriority is the beginning of the range of priority values for critical system
components.
SystemCriticalPriority = 2 * HighestUserDefinablePriority
```

PriorityClass



```
$ kubectl get priorityclass
NAME                    VALUE       GLOBAL-DEFAULT   AGE
system-cluster-critical 2000000000  false            59d
system-node-critical    2000001000  false            59d
```

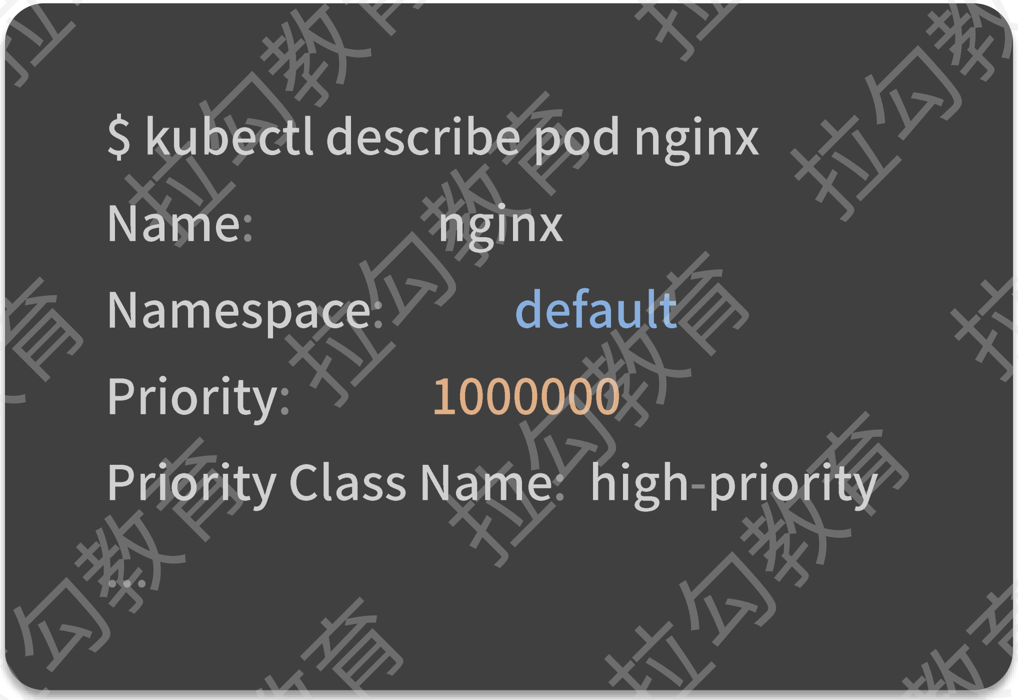
PriorityClass

```
apiVersion: apps/v1
kind: Deployment
metadata:
  ...
  name: coredns
  namespace: kube-system
  ...
spec:
  ...
  template:
    ...
    spec:
      ...
      priorityClassName: system-cluster-critical
      ...
status:
  ...
```

PriorityClass

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
  priorityClassName: high-priority
```

PriorityClass



```
$ kubectl describe pod nginx
Name:              nginx
Namespace:         default
Priority:          1000000
Priority Class Name:  high-priority
...
```

PriorityClass

globalDefault

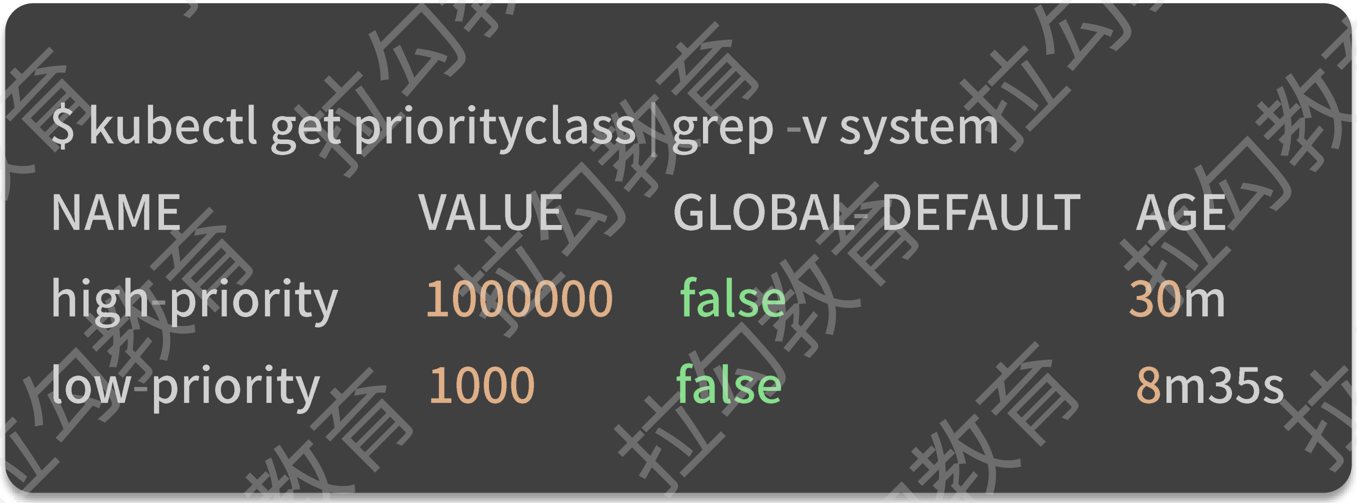
用来表明是否将该 PriorityClass 的数值作为默认值

并将其应用在所有未设置 priorityClassName 的 Pod 上

PriorityClass

```yaml
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
 name: low-priority
value: 1000
globalDefault: false
```

PriorityClass

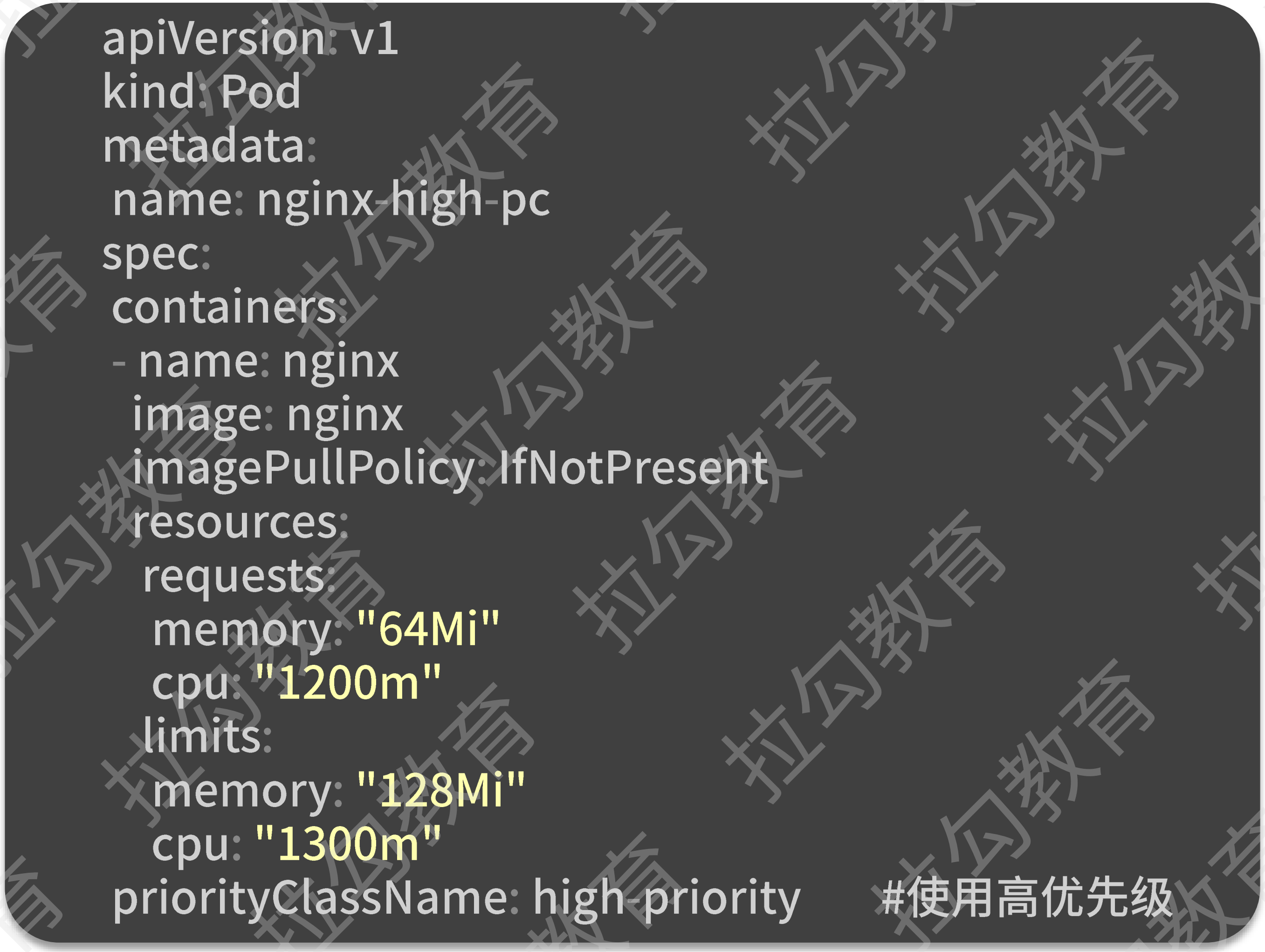


```
$ kubectl get priorityclass | grep -v system
NAME           VALUE      GLOBAL-DEFAULT   AGE
high-priority  1000000    false            30m
low-priority   1000       false            8m35s
```

PriorityClass

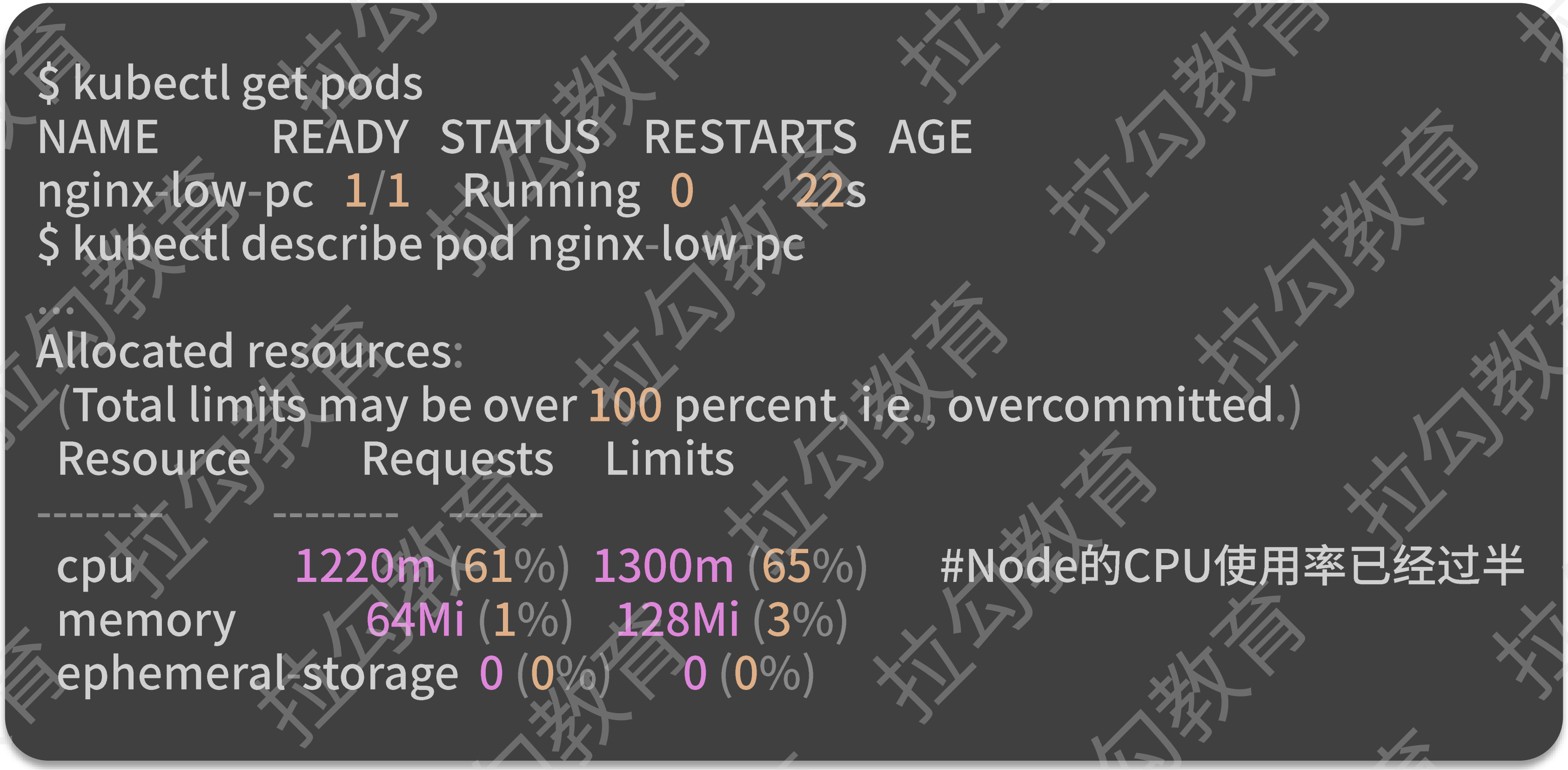
```
apiVersion: v1
kind: Pod
metadata:
 name: nginx-low-pc
spec:
 containers:
 - name: nginx
   image: nginx
   imagePullPolicy: IfNotPresent
   resources:
    requests:
     memory: "64Mi"
     cpu: "1200m"     #CPU需求设置较大
    limits:
     memory: "128Mi"
     cpu: "1300m"
 priorityClassName: low-priority   #使用低优先级
```

PriorityClass



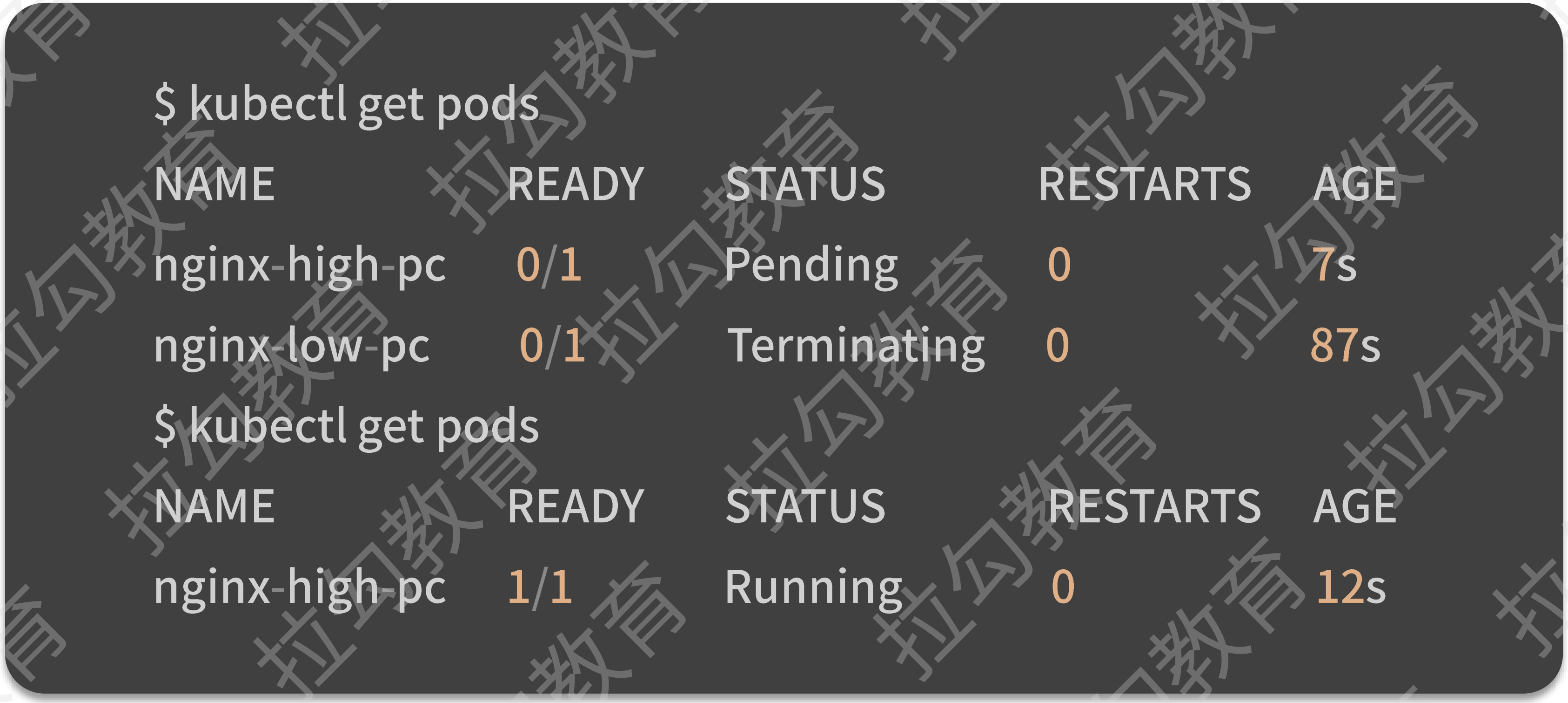
```
apiVersion: v1
kind: Pod
metadata:
 name: nginx-high-pc
spec:
 containers:
 - name: nginx
   image: nginx
   imagePullPolicy: IfNotPresent
   resources:
    requests:
     memory: "64Mi"
     cpu: "1200m"
    limits:
     memory: "128Mi"
     cpu: "1300m"
 priorityClassName: high-priority     #使用高优先级
```

PriorityClass



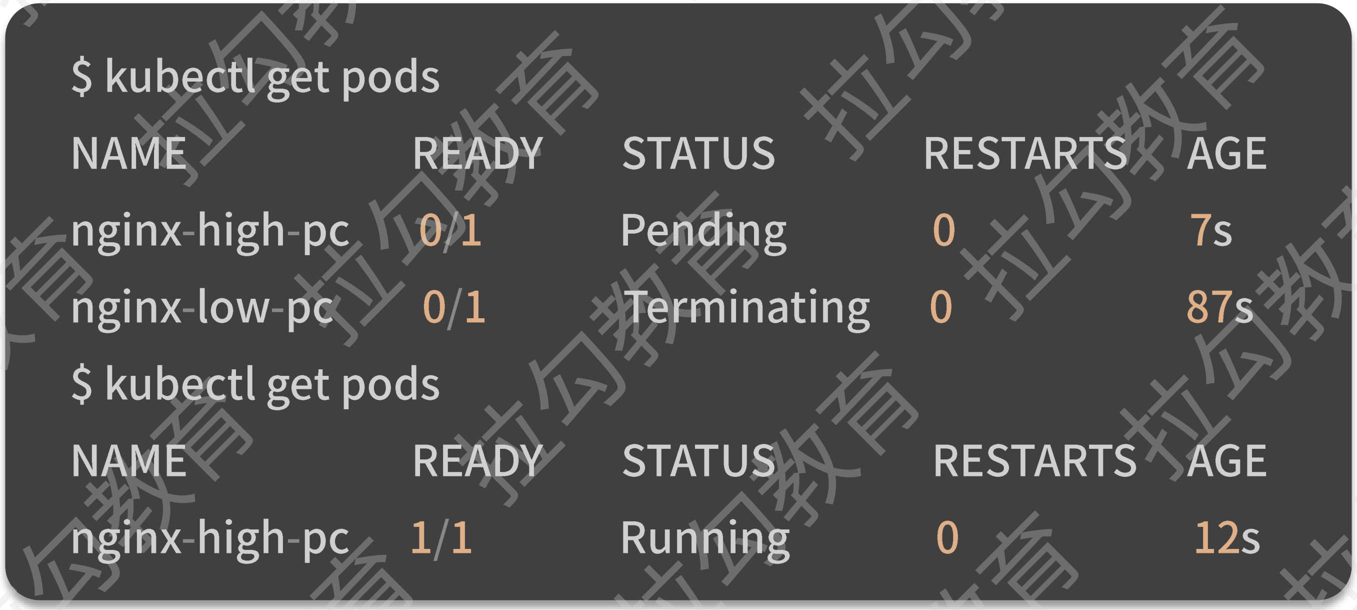
```
$ kubectl get pods
NAME         READY  STATUS   RESTARTS  AGE
nginx-low-pc  1/1    Running  0        22s
$ kubectl describe pod nginx-low-pc
...
Allocated resources:
 (Total limits may be over 100 percent, i.e., overcommitted.)
 Resource        Requests     Limits
--------      --------    ------
 cpu            1220m (61%)  1300m (65%)      #Node的CPU使用率已经过半
 memory            64Mi (1%)    128Mi (3%)
 ephemeral-storage  0 (0%)       0 (0%)
```

PriorityClass



```
$ kubectl get pods
NAME            READY    STATUS        RESTARTS    AGE
nginx-high-pc    0/1     Pending        0           7s
nginx-low-pc     0/1     Terminating    0           87s
$ kubectl get pods
NAME            READY    STATUS         RESTARTS    AGE
nginx-high-pc   1/1      Running         0          12s
```

PriorityClass



```
$ kubectl get pods
NAME            READY    STATUS         RESTARTS   AGE
nginx-high-pc    0/1     Pending         0         7s
nginx-low-pc     0/1     Terminating     0         87s
$ kubectl get pods
NAME            READY    STATUS          RESTARTS  AGE
nginx-high-pc   1/1      Running          0        12s
```

如果这时没有任何一个节点能够满足这个 Pod的所有要求

调度器会尝试寻找一个节点，通过移除一个或者多个比该 Pod 的优先级低的 Pod

尝试使目标 Pod 可以被调度

PriorityClass

```yaml
apiVersion: scheduling.k8s.io/v1
kind: PriorityClass
metadata:
  name: high-priority-nonpreempting
value: 1000000
preemptionPolicy: Never
globalDefault: false
description: "This priority class will not cause other pods to be preempted."
```

PriorityClass

```
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration
algorithmSource:
  provider: DefaultProvider

...

disablePreemption: true
```

写在最后

提高集群的资源利用率最常见的做法就是采用优先级的方案

实际使用时，要避免恶意用户创建高优先级的 Pod

集群管理员可以为特定用户创建特定优先级级别

防止他们恶意使用高优先级的 PriorityClass

Next：《22 | 安全机制：Kubernetes 如何保障集群安全？》