作者：李晓辉

联系方式：

1. 微信：Lxh_Chat

2. 邮箱：939958092@qq.com

让用户自己在 Kubernetes 上创建工作负载确实能提高效率。不过呢，集群的资源是有限的，像 CPU、内存和存储这些。如果工作负载太多，超出了集群能提供的资源，那工作负载可能就运行不正常了。好在 Kubernetes 有个厉害的功能，工作负载可以提前预留资源，还能声明资源的上限。这样就能避免资源被过度占用，保证工作负载都能正常运行，工作负载可以指定下列属性：

资源限值

第一招叫“资源限值”，就是给工作负载设个“天花板”，规定它在正常运行时最多能用多少资源。要是工作负载突然出故障或者负载突然变高，这个限值就能防止它把资源全占了，影响到别的工作负载正常运行。

资源请求

第二招是“资源请求”，工作负载可以提前说一声自己最少需要多少资源。Kubernetes 就会按照这些请求去分配资源，要是集群里资源不够，新的工作负载就暂时不能部署。这样一来，就能保证每个工作负载都能拿到自己需要的资源，安稳运行。

pod上的资源配额

说到创建工作负载，可以给每个 Pod 设置资源配额。就像给 Pod 划个“资源小房间”，让它在里面“住”得刚刚好。

比如说，一个 Pod 既设置了 request（请求配额），又设置了 limits（资源限制）。这就好比给 Pod 定了个“最低生活保障”和“最高消费限额”。

至于单位嘛，内存单位挺有意思的：1Mi 等于 1024KB，这是按二进制来的；1M 等于 1000KB，这是按十进制来的。要是啥单位都不带，那就是字节啦。

CPU 单位也很简单：1 等于 1000m，这个还能带小数点，比如 1.5 就等于 1500m。

```
cat > quota.yml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: pod-limit
  labels:
    name: pod-limit
spec:
  containers:
  - name: app
    image: registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest
    imagePullPolicy: IfNotPresent
    resources:
      requests:
        memory: "64Mi"
        cpu: "100m"
      limits:
        memory: "128Mi"
        cpu: "100m"
EOF
```

创建并确认配额

```
[student@workstation ~]$ oc describe -f quota.yml | grep -A 6 Limits
    Limits:
      cpu:     100m
      memory:  128Mi
    Requests:
      cpu:        100m
      memory:     64Mi
    Environment:  <none>
```

虽然给单个 Pod 设置了资源配额，看起来好像挺合理的，但要是有人“机灵”得很，多创建几个 Pod，那资源不就偷偷被占多了嘛。就像本来给一个 Pod 分了 100m 的 CPU，那要是搞两个 Pod，不就变成 200m 了，三个就是 300m，这要是不限制，资源就被“薅羊毛”了。

所以，为了避免这种情况，就得给整个项目（project）设置个上限，就像给整个“资源池”拉起一道防线，不管对方怎么创建 Pod，总的资源都不会超过这个上限。这样一来，资源分配就更公平、更可控啦，也能防止有人“钻空子”。

项目级别设置资源限额

先创建一个project

```
oc new-project lixiaohui
```

给lixiaohui整个project设置配额

```
cat > project-quota.yml <<EOF
apiVersion: v1
kind: ResourceQuota
metadata:
  name: lixiaohuiquota
  namespace: lixiaohui
spec:
  hard:
    pods: "1"
    requests.cpu: "1"
    requests.memory: "1Gi"
    limits.cpu: "2"
    limits.memory: "2Gi"
EOF
```

创建并查看

这样比较直观的显示了我们目前可以申请多少资源

```
[student@workstation ~]$ oc create -f project-quota.yml
resourcequota/lixiaohuiquota created
[student@workstation ~]$ oc get -f project-quota.yml
NAME             AGE   REQUEST                                                 LIMIT
lixiaohuiquota   8s    pods: 0/1, requests.cpu: 0/1, requests.memory: 0/1Gi   limits.cpu: 0
```

我们试着创建一个超过项目配额的资源请求，看看是否失败

我们申请的内存不超标，但是CPU不管是request还是limit，都是超标的

```
cat > exec-pod.yml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: frontend
  namespace: lixiaohui
spec:
  containers:
  - name: app
    image: registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest
    imagePullPolicy: IfNotPresent
    resources:
      requests:
        memory: "64Mi"
        cpu: "150m"
      limits:
        memory: "128Mi"
        cpu: "150m"
EOF
```

从下面的报错来看，你超过限额就不行，我们通过给project设置限额，杜绝了用多个pod来绕过限额的可能

```
[student@workstation ~]$ oc create -f exec-pod.yml
Error from server (Forbidden): error when creating "exec-pod.yml": pods "frontend" is forbi
```

项目级别设置数量限额

在 Kubernetes 中，可以通过 **ResourceQuota（资源配额）** 来限制一个命名空间（Namespace）内资源的数量和使用情况。这个功能就像是给整个项目设置了一个“资源天花板”，防止资源被过度占用。

为什么需要数量限额?

假设你有一个项目，里面有多个团队在创建 Pod。如果不对数量进行限制，每个团队可能会无限制地创建 Pod，导致整个集群资源被耗尽。比如，一个团队可能为了测试，一口气创建了 100 个 Pod，而其他团队可能只需要 10 个 Pod 就够用了。这样一来，资源分配就会变得很不公平，甚至可能导致集群崩溃。

如何设置数量限额?

数量限额可以通过设置 **ResourceQuota** 来实现。你可以指定一个命名空间内可以创建的 Pod 数量上限。比如，你可以设置一个项目最多只能创建 50 个 Pod。这样一来，不管团队怎么折腾，Pod 的总数都不会超过 50 个。

我们看下面这个例子，在`lixiaohui`的命名空间中只允许创建一个pod

```
cat > number-quota.yml <<-EOF
apiVersion: v1
kind: ResourceQuota
metadata:
  name: number-quota
  namespace: lixiaohui
spec:
  hard:
    count/pods: "1"
EOF
```

创建并验证

```
[student@workstation ~]$ oc create -f number-quota.yml
resourcequota/number-quota created
[student@workstation ~]$ oc get -f number-quota.yml
NAME           AGE   REQUEST           LIMIT
number-quota   3s    count/pods: 0/1
```

我们来创建一个3副本的deployment

```
cat > deployment.yml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: lixiaohui
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8080
        resources:
          requests:
            memory: "64Mi"
            cpu: "150m"
          limits:
            memory: "128Mi"
            cpu: "150m"
EOF
```

创建并验证发现，只有1个在线

```
[student@workstation ~]$ oc create -f deployment.yml
deployment.apps/nginx-deployment created
[student@workstation ~]$ oc get -f deployment.yml
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   1/3     1            1           4s
```

查询为什么只有一个在线

清晰的看到，是因为超过了pod数量才失败

```
[student@workstation ~]$ oc get replicasets.apps
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-577877fb79   3         1         1       48s
[student@workstation ~]$ oc describe replicasets.apps nginx-deployment-577877fb79
...
Events:
  Type     Reason            Age               From                   Message
  ----     ------            ----              ----                   -------
  Normal   SuccessfulCreate  63s               replicaset-controller  Created pod: nginx-d
  Warning  FailedCreate      63s               replicaset-controller  Error creating: pods
```

跨project的集群级别配额

有时候，一个集群里有好多项目（也就是命名空间），每个项目都有自己的资源配额。但有时候，集群管理员可能需要从更高的层面来控制资源，比如限制某个开发团队管理的所有项目总共能用多少资源。这时候，就得用到集群级别的资源配额啦。

想象一下，有一群开发人员，他们管理着好几个项目（也就是好几个命名空间）。每个项目都有自己的资源配额，比如每个项目最多能用 10GB 的内存。但是，集群管理员可能担心这些项目加起来会用掉太多资源，比如整个集群的内存。这时候，就需要一个更高层次的控制手段。

集群资源配额的作用

集群资源配额就像是给整个集群加了一道“总闸门”。它不仅可以限制单个项目（命名空间）的资源使用，还可以限制一组项目（比如某个开发团队管理的所有项目）的总资源使用量。这样一来，即使每个项目都有自己的配额，整个团队的资源使用也不会超过管理员设定的上限。

如何实现?

集群资源配额的实现方式和命名空间资源配额有点像，但多了一个“选择器”（Selector）。选择器的作用就是告诉 Kubernetes：“嘿，我要对这几个特定的项目（命名空间）应用这个配额。”比如，你可以指定一个开发团队管理的所有项目，然后给这些项目设置一个总的内存使用上限。

集群管理员可能会对资源应用限制，而不限于单个命名空间。例如，一组开发人员管理着许多命名空间。命名空间配额可以限制每个命名空间的 RAM 使用量。但是，集群管理员无法限制该开发人员组管理的所有工作负载的总 RAM 使用量。

这里我们给具有quota=lixiahui标签的所有namespace设置了配额

```
cat > cluster-quota.yml <<-EOF
apiVersion: quota.openshift.io/v1
kind: ClusterResourceQuota
metadata:
  name: cluster-quota
spec:
  quota:
    hard:
      limits.memory: 400Mi
  selector:
    annotations: {}
    labels:
      matchLabels:
        quota: lixiahui
EOF
```

创建并确认

```
[student@workstation ~]$ oc create -f cluster-quota.yml
clusterresourcequota.quota.openshift.io/cluster-quota created
[student@workstation ~]$ oc get -f cluster-quota.yml
NAME            AGE
cluster-quota   3s
[student@workstation ~]$ oc describe -f cluster-quota.yml
Name:           cluster-quota
Created:        48 seconds ago
Labels:         <none>
Annotations:    <none>
Namespace Selector: []
Label Selector: quota=lixiahui
AnnotationSelector: map[]
Resource        Used    Hard
--------        ----    ----
limits.memory      0       400Mi
```

我们创建一个zhangsan的project，然后给这个project添加上预期的标签，再创建一个名为lixiaohui的pod

```
oc new-project zhangsan
oc label namespace zhangsan quota=lixiahui
oc describe project zhangsan
Name:                   zhangsan
Created:                23 seconds ago
Labels:                 kubernetes.io/metadata.name=zhangsan
                        pod-security.kubernetes.io/audit=restricted
                        pod-security.kubernetes.io/audit-version=v1.24
                        pod-security.kubernetes.io/warn=restricted
                        pod-security.kubernetes.io/warn-version=v1.24
                        quota=lixiahui
```

看上去我们预期的quota=lixiahui标签已经存在，先来看看集群配额是否已经选中zhangsan

没问题，已经选中了zhangsan

```
[student@workstation ~]$ oc describe clusterresourcequotas.quota.openshift.io cluster-quota
Name:           cluster-quota
Created:        3 minutes ago
Labels:         <none>
Annotations:    <none>
Namespace Selector: ["zhangsan"]
Label Selector: quota=lixiahui
AnnotationSelector: map[]
Resource        Used    Hard
--------        ----    ----
limits.cpu      0       400Mi
```

我们创建一个pod看看

```
cat > lixiaohui-pod.yml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: lixiaohui
  namespace: zhangsan
spec:
  containers:
  - name: app
    image: registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest
    imagePullPolicy: IfNotPresent
    resources:
      requests:
        memory: "64Mi"
        cpu: "150m"
      limits:
        memory: "128Mi"
        cpu: "150m"
EOF
```

查看配额，然后创建出来，再看看配额

```
[student@workstation ~]$ oc create -f lixiaohui-pod.yml
pod/lixiaohui created
```

再看的时候，就用了128Mi了

```
[student@workstation ~]$ oc describe clusterresourcequotas.quota.openshift.io cluster-quota
Name:           cluster-quota
Created:        19 minutes ago
Labels:         <none>
Annotations:    kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"quota.opens

Namespace Selector: ["zhangsan"]
Label Selector: quota=lixiahui
AnnotationSelector: map[]
Resource        Used    Hard
--------        ----    ----
limits.memory   128Mi   400Mi
```

我们再创建一个lisi的project，并分配quota=lixiaohui的标签，然后再创建一个pod，如果集群资源配额又上升，就说明不管你在哪儿创建，都会受到这个限制

```
oc new-project lisi
oc label namespace lisi quota=lixiahui
```

我们创建一个pod看看

```
cat > lixiaohui-pod.yml <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: lixiaohui
  namespace: lisi
spec:
  containers:
  - name: app
    image: registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:latest
    imagePullPolicy: IfNotPresent
    resources:
      requests:
        memory: "64Mi"
        cpu: "150m"
      limits:
        memory: "128Mi"
        cpu: "150m"
EOF
```

创建并验证配额是否上升，从刚才的128，上升到256，说明不管你在哪个project，都会受到限制

```
[student@workstation ~]$ oc create -f lixiaohui-pod.yml
[student@workstation ~]$ oc describe clusterresourcequotas.quota.openshift.io cluster-quota
Name:           cluster-quota
Created:        25 minutes ago
Labels:         <none>
Annotations:    kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"quota.opens

Namespace Selector: ["zhangsan" "lisi"]
Label Selector: quota=lixiahui
AnnotationSelector: map[]
Resource        Used    Hard
--------        ----    ----
limits.memory   256Mi   400Mi
```

本文档在线版本：https://www.linuxcenter.cn