作者：李晓辉

联系方式：

1. 微信：Lxh_Chat

2. 邮箱：939958092@qq.com

LimitRange的概念

Kubernetes 里有个东西叫 `LimitRange` ，这玩意儿可厉害了，简直就是集群资源的“管家”。

为啥这么说呢？在咱们用的 Kubernetes 集群里，要是没有个限制，那可就乱套了。比如在一个多租户的环境里，大家共享一个集群，要是有个应用特别“贪心”，把资源都占了，别的应用不就“饿肚子”了？性能肯定差得不行，甚至可能直接挂掉。`LimitRange` 就是来解决这个问题的，它能给资源设置个“上限”和“下限”，就像给每个应用发了“资源配额”，让大家都公平地分资源。

举个例子，假设一个命名空间里有好几个应用，要是没有 `LimitRange` ，一个特别吃内存的应用可能会把内存都占光，其他应用就只能“干瞪眼”。但有了 `LimitRange` ，就能给每个应用设定最大能用多少内存，这样大家都能正常运行了。

而且， `LimitRange` 还有个超实用的功能，就是能给 Pod 提供默认的资源请求和限制。有时候，用户定义 Pod 的时候，可能没写资源请求和限制，那 Kubernetes 调度器在分配资源的时候就容易“懵圈”，要么给多了，要么给少了，影响整个集群的性能。`LimitRange` 就能搞定这个事儿，它能给 Pod 设置默认的资源请求和限制，比如默认请求 256Mi 内存，限制 512Mi 内存。这样一来，调度器就能根据这些默认值合理分配资源，保证集群稳稳当当的。

在 Kubernetes 中， `LimitRange` 是一种关键的资源类型，用于管理和限制命名空间中的资源使用。通过设置资源的默认请求和限制， `LimitRange` 确保了集群资源的合理分配和高效利用，避免了资源的浪费和过度使用。以下是 `LimitRange` 的一些必要性及其示例说明：

防止资源滥用

在多租户环境中，不同团队或应用共享同一个 Kubernetes 集群。如果没有资源限制，某些应用可能会占用过多资源，导致其他应用的性能下降，甚至无法正常运行。 `LimitRange` 通过设置最大和最小资源限制，防止单个容器或 Pod 占用过多资源，从而确保各个应用能够公平地共享集群资源。

示例：假设一个命名空间中运行了多个应用，如果没有 `LimitRange` 限制，一个内存密集型的应用可能会占用大部分内存资源，导致其他应用因资源不足而崩溃。通过设置 `LimitRange` ，可以限制每个应用的最大内存使用量，保证所有应用都能获得足够的资源运行。

提供合理的默认值

有些用户在定义 Pod 时可能没有指定资源请求和限制。这样一来，Kubernetes 调度器在分配资源时可能会过度或不足分配，影响集群的性能和稳定性。通过设置 `LimitRange`，可以为资源请求和限制提供合理的默认值，即使用户未显式指定，Kubernetes 也能根据这些默认值进行调度。

示例：某个命名空间中的 Pod 没有定义资源请求和限制，导致调度器无法合理分配资源。通过设置 `LimitRange`，可以为 Pod 设置默认的资源请求（如 256Mi 内存）和限制（如 512Mi 内存），确保调度器能够合理分配资源，保证集群的稳定运行。

具体来说，LimitRange可以防止资源被不合理的分配或Pod过多的占用资源，LimitRange也会给Pod提供默认的request和limit

LimitRange 案例

我们来做个测试，创建一个deployment，并观察其默认是否有resources的字段，然后设置limitrange，再推出新的pod，继续观察，不过需要注意的是，LimitRange不影响现有的 pod。

```
cat > deployment.yml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 1
  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
EOF
```

创建并观察pod是否会被自动添加resources资源限制，结果是pod跟随deployment要求，并没有添加限制资源

```
oc create -f deployment.yml
```

看看默认的resources字段，根本就没有这个字段，说明并没有添加限额

```
[student@workstation ~]$ oc get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6645d8bb58-5hbzp   1/1     Running   0          119s
[student@workstation ~]$ oc describe pod nginx-deployment-6645d8bb58-5hbzp
```

我们来创建一个LimitRange

```
cat > limitrange.yml <<-EOF
apiVersion: v1
kind: LimitRange
metadata:
  name: lixiaohui-limit
spec:
  limits:
  - default:
      cpu: 500m             # 默认 CPU 限制为 500 毫核
      memory: 512Mi         # 默认内存限制为 512 Mi
    defaultRequest:
      cpu: 250m             # 默认 CPU 请求为 250 毫核
      memory: 256Mi         # 默认内存请求为 256 Mi
    max:
      cpu: "1"              # 最大 CPU 限制为 1 核
      memory: 1Gi           # 最大内存限制为 1 Gi
    min:
      cpu: 125m             # 最小 CPU 请求为 125 毫核
      memory: 128Mi         # 最小内存请求为 128 Mi
    type: Container         # 适用于容器级别
EOF
```

`min` 和 `max` 的作用

- `min`：指定容器或 Pod 可以请求的最小资源量。这个值确保每个容器或 Pod 至少请求一定量的资源，以避免资源不足的问题。
- `max`：指定容器或 Pod 可以请求的最大资源量。这个值限制了容器或 Pod 不能请求过多的资源，以防止单个容器或 Pod 占用过多的资源。

`default` 和 `defaultRequest` 的作用

- `default`：为没有显式指定资源限制的容器或 Pod 设置默认的资源限制。
- `defaultRequest`：为没有显式指定资源请求的容器或 Pod 设置默认的资源请求。

```
oc create -f limitrange.yml
```

创建后，发现pod依然没有resources，那是因为并不影响已有的pod，所以我们删除这个pod，由deployment自动产生一个新的，再describe看看

```
[student@workstation ~]$ oc delete pod nginx-deployment-6645d8bb58-5hbzp
pod "nginx-deployment-6645d8bb58-5hbzp" deleted

[student@workstation ~]$ oc get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-6645d8bb58-qt42t   1/1     Running   0          5s

[student@workstation ~]$ oc describe pod nginx-deployment-6645d8bb58-qt42t
...
    Limits:
      cpu:     500m
      memory:  512Mi
    Requests:
      cpu:        250m
      memory:     256Mi
```

我们发现虽然deployment本身没有提供限额，但是pod却根据limitrange添加了限额，这样就可以防止资源被过多浪费，不过需要注意的是，limitrange并没有修改deployment，而是直接影响pod，如果需要，我们可以用下面的方式修改deployment的限额

```
[student@workstation ~]$ oc set resources deployment nginx-deployment --requests cpu=100m,m
deployment.apps/nginx-deployment resource requirements updated
```

CPU 或内存密钥的值必须遵循下列规则：

- `max` 值必须大于或等于 `default` 值。
- `default` 值必须大于或等于 `defaultRequest` 值。
- `defaultRequest` 值必须大于或等于 `min` 值。

本文档在线版本：https://www.linuxcenter.cn