实践篇-Qos

一、什么是Qos

> QoS类是Kubernetes用来决定Pod的调度和驱逐的策略
> 本文介绍怎样配置Pod让其获得特定的服务质量（QoS）类

二、QoS级别

2.1、QoS级别

- Guaranteed：POD中所有容器都必须统一设置了limits，并且设置参数都一致，如果有一个容器要设置requests，那么所有容器都要设置，并设置参数同limits一致
- Burstable：POD中只要有一个容器，这个容器requests和limits的设置同其他容器设置的不一致
- BestEffort：POD中的所有容器都没有指定CPU和内存的requests和limits

删除策略：先删除服务质量为BestEffort，然后在删除Burstable，Guaranteed最后被删除

三、举例说明

3.0、创建命名空间

创建一个命名空间，以便将本练习所创建的资源与集群的其余资源相隔离

```
[root@k8s-master01 ~]# kubectl create namespace qos-example
namespace/qos-example created
```

3.1、创建一个 QoS 类为 Guaranteed 的 Pod

对于 QoS 类为 Guaranteed 的 Pod：

- Pod 中的每个容器，包含初始化容器，必须指定内存请求和内存限制，并且两者要相等。
- Pod 中的每个容器，包含初始化容器，必须指定 CPU 请求和 CPU 限制，并且两者要相等

```
# 1、创建qos-pod.yaml
cat > qos-pod.yaml << EFO
apiVersion: v1
kind: Pod
metadata:
  name: qos-demo
  namespace: qos-example
spec:
  containers:
  - name: qos-demo-ctr
    image: nginx:1.15.2
    resources:
      limits:
```

```
        memory: "200Mi"
        cpu: "700m"
      requests:
        memory: "200Mi"
        cpu: "700m"
EFO

# 2、创建 Pod
[root@k8s-master01 ~]# kubectl create -f qos-pod.yaml -n qos-example
pod/qos-demo created

# 3、查看 Pod 详情
# 结果表明 Kubernetes 为 Pod 配置的 QoS 类为 Guaranteed。 结果也确认了 Pod 容器设置了与内存限制匹配的内存请求，设置了与 CPU 限制匹配的 CPU 请求。
[root@k8s-master01 ~]# kubectl get pod qos-demo --namespace=qos-example --output=yaml
spec:
  containers:
  - image: nginx:1.15.2
    imagePullPolicy: IfNotPresent
    name: qos-demo-ctr
    resources:
      limits:
        cpu: 700m
        memory: 200Mi
      requests:
        cpu: 700m
        memory: 200Mi
  ............
status:
  qosClass: Guaranteed
# 4、删除Pod
[root@k8s-master01 ~]# kubectl delete pod qos-demo --namespace=qos-example
pod "qos-demo" deleted
```

> **说明：** 如果容器指定了自己的内存限制，但没有指定内存请求，Kubernetes 会自动为它指定与内存限制匹配的内存请求。 同样，如果容器指定了自己的 CPU 限制，但没有指定 CPU 请求，Kubernetes 会自动为它指定与 CPU 限制匹配的 CPU 请求。

3.2、创建一个 QoS 类为 Burstable 的 Pod

如果满足下面条件，将会指定 Pod 的 QoS 类为 Burstable：

- Pod 不符合 Guaranteed QoS 类的标准。
- Pod 中至少一个容器具有内存或 CPU 请求。

下面是包含一个容器的 Pod 配置文件。 容器设置了内存限制 200 MiB 和内存请求 100 MiB。

```
apiVersion: v1
kind: Pod
metadata:
```

```
  name: qos-demo-2
  namespace: qos-example
spec:
  containers:
  - name: qos-demo-2-ctr
    image: nginx
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"

# 结果表明 Kubernetes 为 Pod 配置的 QoS 类为 Guaranteed。 结果也确认了 Pod 容器设置了与内存限制匹配的内存请求，设置了与 CPU 限制匹配的 CPU 请求。
spec:
  containers:
    ...
    resources:
      limits:
        cpu: 700m
        memory: 200Mi
      requests:
        cpu: 700m
        memory: 200Mi
  ......
status:
  qosClass: Guaranteed
```

> **说明：** 如果容器指定了自己的内存限制，但没有指定内存请求，Kubernetes 会自动为它指定与内存限制匹配的内存请求。 同样，如果容器指定了自己的 CPU 限制，但没有指定 CPU 请求，Kubernetes 会自动为它指定与 CPU 限制匹配的 CPU 请求。

3.3、创建一个 QoS 类为 BestEffort 的 Pod

- 对于 QoS 类为 BestEffort 的 Pod，Pod 中的容器必须没有设置内存和 CPU 限制或请求

下面是包含一个容器的 Pod 配置文件。 容器没有设置内存和 CPU 限制或请求。

```
apiVersion: v1
kind: Pod
metadata:
  name: qos-demo-3
  namespace: qos-example
spec:
  containers:
  - name: qos-demo-3-ctr
    image: nginx

# 结果表明 Kubernetes 为 Pod 配置的 QoS 类为 BestEffort。
spec:
```

```
  containers:
    ...
    resources: {}
  ...
status:
  qosClass: BestEffort
```

3.4、创建包含两个容器的 Pod

下面是包含两个容器的 Pod 配置文件。 一个容器指定了内存请求 200 MiB。 另外一个容器没有指定任何请求和限制。

```
apiVersion: v1
kind: Pod
metadata:
  name: qos-demo-4
  namespace: qos-example
spec:
  containers:

  - name: qos-demo-4-ctr-1
    image: nginx
    resources:
      requests:
        memory: "200Mi"

  - name: qos-demo-4-ctr-2
    image: redis

# 结果表明
spec:
  containers:
    ...
    name: qos-demo-4-ctr-1
    resources:
      requests:
        memory: 200Mi
    ...
    name: qos-demo-4-ctr-2
    resources: {}
    ...
status:
  qosClass: Burstable
```

注意此 Pod 满足 Burstable QoS 类的标准。 也就是说它不满足 Guaranteed QoS 类标准，因为它的一个容器设有内存请求。