1.1 QoS

1.1.1 配置 Pod 的资源配额

步骤 1　创建一个文件夹，用于存放 qos 实验相关文件。

```
[root@k8s-master labfile]# mkdir /labfile/qosfile
[root@k8s-master labfile]# cd /labfile/qosfile
```

步骤 2　查看 node 资源。

```
[root@k8s-master qosfile]# kubectl get node k8s-node1 -o yaml
```

回显中可以找到如下一段参数

```
allocatable:
    cpu: "4"
    ephemeral-storage: "42495643169"
    hugepages-1Gi: "0"
    hugepages-2Mi: "0"
    memory: 7906792Ki
    pods: "110"
  capacity:
    cpu: "4"
    ephemeral-storage: 46110724Ki
    hugepages-1Gi: "0"
    hugepages-2Mi: "0"
    memory: 8009192Ki
    pods: "110"
```

步骤 3　创建 namespace

```
[root@k8s-master qosfile]# kubectl create namespace qos-namespace
```

步骤 4　配置一个 BestEffort 级别的 pod

```
[root@k8s-master qosfile]# vim best-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: best-pod
  namespace: qos-namespace
spec:
  containers:
  - name: best-container
```

```
    image: nginx
```

步骤 5　创建 pod

```
[root@k8s-master qosfile]# kubectl apply -f best-pod.yaml
```

查看 pod 类型

```
[root@k8s-master qosfile]# kubectl describe pods best-pod -n qos-namespace
QoS Class:       BestEffort
```

步骤 6　创建 Burstable 的 Pod 的 yaml

```
[root@k8s-master qosfile]# vim bur-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: bur-pod
  namespace: qos-namespace
spec:
  containers:
  - name: bur-container
    image: nginx
    resources:
      limits:
        memory: "200Mi"
      requests:
        memory: "100Mi"
```

步骤 7　创建 pod

```
[root@k8s-master qosfile]# kubectl apply -f bur-pod.yaml
```

查看 pod 的 qos 类型

```
[root@k8s-master qosfile]# kubectl describe pod bur-pod -n qos-namespace
QoS Class:       Burstable
```

步骤 8　创建 Guaranteed 类型的 pod 的 yaml

```
[root@k8s-master qosfile]# vim gua-pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: gua-pod
  namespace: qos-namespace
spec:
  containers:
  - name: gua-container
```

```
    image: nginx
    resources:
      limits:
        memory: "200Mi"
        cpu: "700m"
      requests:
        memory: "200Mi"
        cpu: "700m"
```

步骤 9　创建 pod

```
[root@k8s-master qosfile]# kubectl apply -f gua-pod.yaml
```

查看 pod 的 qos 类型

```
[root@k8s-master qosfile]# kubectl describe pod gua-pod -n qos-namespace
QoS Class:       Guaranteed
```

1.1.2 测试内存限制

步骤 1　创建一个 yaml 文件，该文件定义了一个 stress 应用的 Pod，并且 Pod 内程序运行会占用超过 Pod 限额的内存。

```
[root@k8s-master qosfile]# vim outmem.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo
  namespace: qos-namespace
spec:
  containers:
  - name: memory-demo-ctr
    image: polinux/stress
    resources:
      limits:
        memory: "100Mi"
      requests:
        memory: "50Mi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

步骤 2　运行 Pod

```
[root@k8s-master qosfile]# kubectl apply -f outmem.yaml
pod/memory-demo created
```

步骤 3　查看 Pod 状态，可以看到状态是 OOMKilled。

```
[root@k8s-master qosfile]# kubectl get pod -n qos-namespace
```

```
NAME          READY   STATUS      RESTARTS   AGE
memory-demo   0/1     OOMKilled   4          116s
```

查看更详细的 Pod 信息可以看到，Pod 出现问题的原因是 OOMKilled

```
[root@k8s-master qosfile]# kubectl describe pod memory-demo -n qos-namespace
```

```
Name:             memory-demo
Namespace:        qos-namespace
……
    State:        Waiting
      Reason:     CrashLoopBackOff
    Last State:   Terminated
      Reason:     OOMKilled
      Exit Code:  1
      Started:    Fri, 09 Aug 2019 03:18:57 -0400
      Finished:   Fri, 09 Aug 2019 03:18:57 -0400
    Ready:        False
    Restart Count:  4
……
```

1.1.3 测试节点资源限额

步骤 1　创建一个 yaml，该文件定义了一个 pod，但这个 Pod 使用的资源超出节点拥有的资源。

```
[root@k8s-master qosfile]# vim outnode.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo2
  namespace: qos-namespace
spec:
  containers:
  - name: memory-demo2-ctr
    image: polinux/stress
    resources:
      limits:
        memory: "1000Gi"
      requests:
        memory: "1000Gi"
    command: ["stress"]
    args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

步骤 2　创建该 Pod

```
[root@k8s-master qosfile]# kubectl apply -f outnode.yaml
```

```
pod/memory-demo2 created
```

步骤 3　查看 Pod 状态，Pod 一直无法创建成功。

```
[root@k8s-master qosfile]# kubectl get pod -n qos-namespace
NAME          READY   STATUS    RESTARTS   AGE
memory-demo2   0/1     Pending   0          13m
```

步骤 4　查看 Pod 创建不成功的原因，可以看到是由于没有节点有足够的资源运行 Pod。

```
[root@k8s-master qosfile]# kubectl describe pod memory-demo2 -n qos-namespace
Name:             memory-demo2
Namespace:         qos-namespace
……
Events:
  Type     Reason            Age                From               Message
  ----     ------            ----               ----               -------
  Warning  FailedScheduling  65s (x19 over 25m)  default-scheduler  0/1 nodes
are available: 3 Insufficient memory.
```