[5.1 创建Pod](https://note.youdao.com/ynoteshare1/iframe.html#7775-1575372433333)

[5.2 设置pod的资源请求与限制](https://note.youdao.com/ynoteshare1/iframe.html#4137-1575371777721)

[5.3 一个pod里可以有多个容器](https://note.youdao.com/ynoteshare1/iframe.html#4043-1575372623380)

[5.4 登陆容器操作](https://note.youdao.com/ynoteshare1/iframe.html#3780-1575371777724)

[5.5 apply、edit与patch的使用](https://note.youdao.com/ynoteshare1/iframe.html#9442-1575372706027)

[5.6 Init Containers](https://note.youdao.com/ynoteshare1/iframe.html#9896-1575380930842)

[5.7 static Pods](https://note.youdao.com/ynoteshare1/iframe.html#6925-1575381445073)

**5. pod**

**5.1 创建Pod**

# 编写yaml

cat << EOF > pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: memory-demo

spec:

containers:

- name: demo

image: polinux/stress

command: ["stress"]

args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]

EOF

# 执行yaml创建pod

kubectl create -f pod.yaml

##### 查看pod的状态

NAME: Pod名字

READY： Pod 是否为READY

STATUS： Pod状态

RESTARTS：容器的重启次数

AGE：生命周期

注意：

在kubernetes中，镜像的下载策略为：

　Always：每次都下载镜像（默认）

　Never：只使用本地镜像，从不下载

　IfNotPresent：只有当本地没有的时候才下载镜像

### 编辑已经创建的pod，修改imagePullPolicy: Always 为 imagePullPolicy: IfNotPresent

kubectl -n mem-example edit pod demo

### 在yaml文件里指定image下载策略

cat << EOF > pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: memory-demo

spec:

containers:

- name: demo

image: polinux/stress

imagePullPolicy: IfNotPresent

command: ["stress"]

args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]

EOF

**5.2 设置pod的资源请求与限制**

# 编写yaml

cat << EOF > stress-pod.yaml

apiVersion: v1

kind: Namespace

metadata:

name: mem-example

---

apiVersion: v1

kind: Pod

metadata:

name: memory-demo

namespace: mem-example

spec:

containers:

- name: memory-demo-ctr

image: polinux/stress

imagePullPolicy: IfNotPresent

resources:

limits:

memory: "200Mi"

requests:

memory: "100Mi"

command: ["stress"]

args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]

EOF

# 当程序运行内存超过指定范围

kubectl create namespace mem-example

cat << EOF > stress-pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: memory-demo-2

spec:

containers:

- name: memory-demo-2-ctr

image: polinux/stress

imagePullPolicy: IfNotPresent

resources:

requests:

memory: "50Mi"

limits:

memory: "100Mi"

command: ["stress"]

args: ["--vm", "1", "--vm-bytes", "250M", "--vm-hang", "1"]

EOF

kubectl create -f stress-pod.yaml -n mem-example

### 查看会发现memory-demo-2这个pod状态变为OOMKilled，因为它是内存不足所以显示Container被杀死

[root@master01 ~]# kubectl get pod -n mem-example

NAME READY STATUS RESTARTS AGE

memory-demo 1/1 Running 0 7m13s

memory-demo-2 0/1 OOMKilled 2 20s

**5.3 一个pod里可以有多个容器**

# 编写yaml

cat << EOF > mu-pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: memory-demo

spec:

containers:

- name: memory-demo-ctr-1

image: polinux/stress

imagePullPolicy: IfNotPresent

resources:

limits:

memory: "200Mi"

requests:

memory: "100Mi"

command: ["stress"]

args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]

- name: memory-demo-ctr-2

image: polinux/stress

imagePullPolicy: IfNotPresent

resources:

limits:

memory: "200Mi"

requests:

memory: "100Mi"

command: ["stress"]

args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]

EOF

**5.4 登陆容器操作**

# pod封装一个容器

kubectl -n mem-example exec -it memory-demo bash

kubectl -n mem-example exec -it memory-demo -- ls /root

# pod封装多个容器

kubectl -n mem-example exec -it -c memory-demo-ctr-2 memory-demo bash

kubectl -n mem-example exec -it -c memory-demo-ctr-2 memory-demo -- ls /root

**5.5 apply、edit与patch的使用**

# 编写yaml

cat << EOF > pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: memory-demo

labels:

app: test

spec:

containers:

- name: demo

image: polinux/stress

command: ["stress"]

args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]

EOF

# 执行yaml

kubectl apply -f pod.yaml

# 查看pod的label

kubectl get pod --show-labels

# apply更新本地的yaml，通过apply更新配置

更改app: test 为 app: stress

kubectl apply -f pod.yaml

# edit（直接改）

kubectl edit pod memory-demo

# patch

kubectl get pod memory-demo -o json

kubectl patch pod memory-demo -p '{"metadata": {"labels": {"app": "damon"}}}'

**5.6 Init Containers**

<https://kubernetes.io/zh/docs/concepts/workloads/pods/init-containers/>

# 编写yaml

cat << EOF > initcontainers.yaml

apiVersion: v1 kind: Pod metadata: name: myapp-pod labels: app: myapp spec: containers: - name: myapp-container image: busybox:1.28 command: ['sh', '-c', 'echo The app is running! && sleep 3600'] initContainers: - name: init-myservice image: busybox:1.28 command: ['sh', '-c', 'echo init containers test && sleep 10']

EOF

# 执行yaml文件

kubectl apply -f initcontainers.yaml

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal Scheduled 5m34s default-scheduler Successfully assigned default/myapp-pod to node01

Normal Pulling 5m33s kubelet, node01 Pulling image "busybox:1.28"

Normal Pulled 5m17s kubelet, node01 Successfully pulled image "busybox:1.28"

Normal Created 3m35s (x5 over 5m17s) kubelet, node01 Created container init-myservice

Normal Started 3m35s (x5 over 5m17s) kubelet, node01 Started container init-myservice

Normal Pulled 3m35s (x4 over 5m16s) kubelet, node01 Container image "busybox:1.28" already present on machine

Warning BackOff 32s (x23 over 5m15s) kubelet, node01 Back-off restarting failed container

实际上是因为cmd命令写错了。多了done循环

**5.7 static Pods**

<https://v1-14.docs.kubernetes.io/zh/docs/tasks/administer-cluster/static-pod/>

# 编写yaml

cat <<EOF >/etc/kubernetes/manifests/static-web.yaml

apiVersion: v1

kind: Pod

metadata:

name: static-web

labels:

role: myrole

spec:

containers:

- name: web

image: nginx

imagePullPolicy: IfNotPresent

ports:

- name: web

containerPort: 80

protocol: TCP

EOF

### 在master节点查看，只能查看不能对其管理有任何操作

[root@master01 ~]# kubectl get pod

NAME READY STATUS RESTARTS AGE

static-web-node01 1/1 Running 0 6s

验证操作：

kubectl exec -it static-web-node01 -- touch /root/test

kubectl delete pod static-web-node01

kubectl exec -it static-web-node01 -- ls /root/test

结果：发现test文件依然存在

配置文件就是放在特定目录下的标准的 JSON 或 YAML 格式的 pod 定义文件。用 **kubelet --pod-manifest-path=<the directory>** 来启动 kubelet 进程或者在 [KubeletConfiguration 文件](https://v1-14.docs.kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file)中添加 **staticPodPath: <the directory>** 字段，kubelet 将会周期扫描 **<the directory>** 这个目录，根据这个目录下出现或消失的 YAML/JSON 文件来创建或删除静态 pod。

修改路径：

1。

systemctl status kubelet

vim /etc/systemd/system/kubelet.service.d/10-kubeadm.conf

Environment="KUBELET\_CONFIG\_ARGS=--config=/var/lib/kubelet/config.yaml"

改为Environment="KUBELET\_CONFIG\_ARGS=--config=/var/lib/kubelet/config.yaml **--pod-manifest-path=**/root/storage"

systemctl restart kubelet

systemctl daemon-reload; systemctl restart kubelet.service

2.

vi /var/lib/kubelet/config.yaml

docker ps -a | grep -v Up | xargs docker rm -f

pause的作用