请注意：

k8s 1.20版本已经删除PodPreset功能，低于1.20版本正常使用

准入控制：

就是在创建资源经过身份验证之后，kube-apiserver在数据写入etcd之前做一次拦截，然后对资源进行更改、判断正确性等操作。

LimitRanger：

limits:

- default: #没有配置limit或者request的时候设置的默认值，limits

cpu: 50m

memory: 50Mi

defaultRequest: #Request限制

cpu: 10m

memory: 20Mi

type: Container

max: # Limit配置不能超过max设置的值

cpu:

memory:

min: # Request的值不能低于min的值

cpu:

memory:

- max: #指定Pod里面所有的Container加在一起的CPU和内存不能超过这个值,limit

cpu: 50m

memory: 50Mi

type: Pod

min: request

<https://kubernetes.io/docs/concepts/policy/limit-range/>

example：

vim limitranger.yaml

apiVersion: v1

kind: LimitRange

metadata:

name: limits-test

spec:

limits:

- max:

cpu: "4"

memory: 2Gi

min:

cpu: 200m

memory: 6Mi

maxLimitRequestRatio:

cpu: 3

memory: 2

type: Pod

- default:

cpu: 300m

memory: 200Mi

defaultRequest:

cpu: 200m

memory: 100Mi

max:

cpu: "2"

memory: 1Gi

min:

cpu: 100m

memory: 3Mi

maxLimitRequestRatio:

cpu: 5

memory: 4

type: Container

kubectl create -f limitranger.yaml -n XXXX

<https://kubernetes.io/docs/concepts/policy/resource-quotas/>

resourceQuota Example：

[root@k8s-master01 ~]# cat resourcequota.yaml

apiVersion: v1

kind: ResourceQuota

metadata:

name: resource-test

labels:

app: resourcequota

spec:

hard:

pods: 50

requests.cpu: 0.5

requests.memory: 512Mi

limits.cpu: 5

limits.memory: 16Gi

configmaps: 20

persistentvolumeclaims: 20

replicationcontrollers: 20

secrets: 20

services: 50

kubectl create -f resourcequota.yaml -n NAMESPACE-SELECT

QoS：Quality of Service，服务质量。

Guaranteed：k8s的最高的服务质量，也就是CPU内存的Request和Limit的值一样。

Burstable：CPU内存的request有配置，但是值不一样

BestEffort：没有配置Request和Limit

OOM，先删除服务质量为BestEffort，然后在删除Burstable，Guaranteed最后被删除。

16核—request=limit，1 一个16核的宿主机只能运行16个容器。

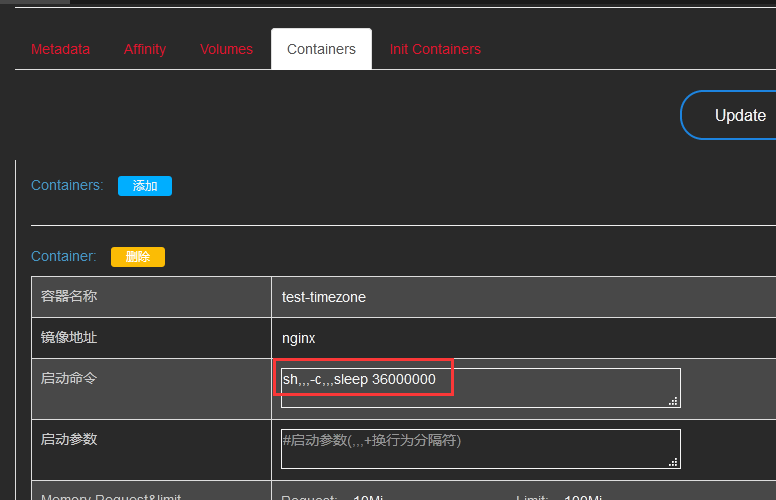
Node上面的CPU，利用率其实很低。100m=100m

3G 3.5G

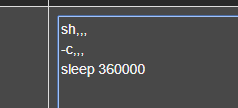
Ratel安装请参考

最新安装文档：<https://github.com/dotbalo/ratel-doc/blob/master/cluster/Install.md>

注意视频的这一块已经更新，此前的有bug，需要将



改成（新版ratel 通过,,, 加上回车进行换行）



请注意：

k8s 1.20版本已经删除PodPreset功能，低于1.20版本正常使用

PodPreset：

Pod的一些预配置。比如添加一些变量、挂载卷、配置容器的时间、字符集等。

开启PodPreset的方法：<https://kubernetes.io/docs/concepts/workloads/pods/podpreset/>

使用PodPreset的文档：<https://kubernetes.io/docs/tasks/inject-data-application/podpreset/>

【课程配置的修改是针对二进制安装方式的，kubeadm的配置文件需要参考kubeadm安装文档最后说明的配置文件的位置】

是有namespace隔离。

同一个Volume不能挂载到不同的路径

apiVersion: settings.k8s.io/v1alpha1

kind: PodPreset

metadata:

name: time

spec:

# selector:

# matchLabels:

# role: frontend

env:

- name: LANG

value: "C.UTF-8"

volumeMounts:

- mountPath: /usr/share/zoneinfo/Asia/Shanghai

name: tz-config

readOnly: true

- mountPath: /etc/localtime

name: tz-config2

readOnly: true

- mountPath: /etc/timezone

name: timezone

readOnly: true

volumes:

- name: tz-config2

hostPath:

path: /usr/share/zoneinfo/Asia/Shanghai

- name: tz-config

hostPath:

path: /usr/share/zoneinfo/Asia/Shanghai

- name: timezone

hostPath:

path: /etc/timezone

忽略预配置：podpreset.admission.kubernetes.io/exclude: "true"

RBAC：

<https://www.cnblogs.com/dukuan/p/11976406.html>

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

labels:

ratel: "true"

username: xxx1

name: ratel-pod-delete-xxx1

namespace: ratel-test1

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: ratel-pod-delete

subjects:

- apiGroup: rbac.authorization.k8s.io

kind: User

name: xxx1

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

labels:

ratel: "true"

username: xxx1

name: ratel-pod-exec-xxx1

namespace: ratel-test1

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: ratel-pod-exec

subjects:

- apiGroup: rbac.authorization.k8s.io

kind: User

name: xxx1

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

labels:

ratel: "true"

username: xxx1

name: ratel-resource-readonly-xxx1

namespace: ratel-test1

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: ratel-resource-readonly

subjects:

- apiGroup: rbac.authorization.k8s.io

kind: User

name: xxx1

使用ClusterRole创建通用权限，使用roleBinding绑定到指定namespace下的User或者是指定的ServiceAccount上。

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

labels:

ratel: "true"

username: java1

name: ratel-pod-exec-sa-java1

namespace: default

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: ratel-pod-exec

subjects:

- kind: ServiceAccount

name: java1

namespace: kube-users