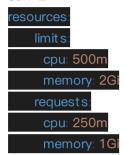
k8s - 资源

cdh-2



资源请求(requests)—— 容器被分配到的资源,最低保障。 资源限制(limits)—— 容器最多能使用到的资源上限。

cpu: 500m --- 表示占用0.5个CPU共享资源。

memory: 500Mi --- 1Mi=1024*1024; 1M=1000*1000.

spec.container[].resources.requests.cpu spec.container[].resources.requests.memory spec.container[].resources.limits.cpu spec.container[].resources.limits.memory

cgroups

cpu:

kubectI exec _ti cdh_2 cat /sys/fs/cgroup/cpu/cpu.cfs_period_us kubectI exec _ti cdh_2 cat /sys/fs/cgroup/cpu/cpu.cfs_quota_us (cfs_quota_us/cfs_period_us, 在长度cfs_period_us的一段时间内,只能被分配到 cfs_quota的CPU时间)

容器内运行的进程会被自动添加到/sys/fs/cgroup/cpu/tasks中momon/:

kubectl exec -ti cdh-2 cat /sys/fs/cgroup/memory/memory.max_usage_in_bytes

Cpu 100%:

for i in `seq 1 \$(cat /proc/cpuinfo |grep "physical id" |wc -I)`; do dd if=/dev/zero of=/dev/null & done

echo '1111' > file1 echo '222222222' > file2 while true;do cat file1 >> file2;cat file2 >> file1;done /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf/var/lib/kubelet/config.yaml

动态资源边界

大部分作业使用到的资源远小于它所请求的资源限额。

requests值则是给调度器是判断用的,较大的limits设定的是cgroups里的配置。

(确保节点资源请求值低于100%, 否则会因"Node didn't have enough resource"而无法启动pod)

Eviction策略

各个node上的kubelet负责定期采集资源占用数据,并与预设的threshold值进行比对,如果超过threshold值,kubelet尝试杀掉一些pod以回收资源(具体的优先级按照QoS),对Node进行保护。

eviction配置: /var/lib/kubelet/config.yaml

QoS模型(优先级从高到低):

- 1、Guaranteed: requests和limits两者相等;
- 2、Burstable: limits超过requests;
- 3、BestEffort: requests和limits均未设置;

当宿主机的不可压缩资源紧张(memory、nodefs)时,会进入MemoryPressure或 DiskPressure状态,避免新的pod被调度到当前宿主机上,同时,kubelet对pod进行 eviction(即资源回收)。

生产环境下,可以将容器绑定到某个CPU核上,不共享CPU的计算能力,这样可以避免CPU 频繁进行上下文切换,提升性能。

如何操作呢?将requests.cpu和limits.cpu均设置为整数值。

Q: 创建calico网络报错: "Calico node XXX is already using the IPv4 address 10.xx.xx.xx" A: Calico自动检测方法默认是first-found,所以可能或获取到外网ip,而虚拟机的外网ip是相同的,所以需要强制指定返回内网ip:

Valid IP address on interface eth0, eth1, eth2 etc.

IP_AUTODETECTION_METHOD=interface=eth.*

IP6_AUTODETECTION_METHOD=interface=eth.*

通过指定网卡来获取内网ip