# 1.1有状态应用管理StatefulSet

# 1.1.1基本概念

statefulset ，缩写是sts，主要用于管理有状态的应用程序的工作负责API对象，比如在生产环境中可部署ElasticSearch集群、MongoDB集群或者需要持久化的RabbitMQ集群、Redis集群、Kafka集群和ZooKeeper集群等

和Deloyment类似，一个StatefulSet也同样管理着基于相同容器规范的Pod。不同的是，Stateful为每个Pod维护了一个粘性标识。这些Pod是根据相同的规范创建的，但是不可互换，每个Pod都有一个持久的标识符，在重新调度的时也会保留，一般格式为StatefulSetName-Number。比如一个名字是Redis-Sentinel的StatefulSet，制定创建3个Pod,那么创建出来的名字就是Redis-Sentinel-0，Redis-Sentinel-1，Redis-Sentinel-2。而StatefulSet创建的Pod一般使用Headless Service（无头服务）进行通信，和普通的Service的区别在于Headless Service没有ClusterIP，它使用的是EndPoint进行互相通信，Headless一般格式为：

StatefulSetName-{0…N-1}.serviceName.namespace.svc.cluster.local

说明：

* serviceName 为 Headless Service的名字，创建StatefulSetName必须指定 Headless Service名称（也就是指定为哪个service创建pod）
* .cluster.local为Cluster Domain（集群域）

# 1.1.2创建一个StatefulSet

#kubectl create –f …..yaml

#kubectl get sts –owide

# 1.1.3 StatefulSet扩容缩容：看过程

按顺序起pod，只有-0完全启动之后才会启动-1，依次类推如果0挂掉了，只有等0正常ready才会启动下一个

删除的话从N-1开始删除。假如总共有3个，删除3的过程中，1挂了，只有等1正常Ready了才会继续删除2；如果要扩容5个，在过程中当扩到3时删除了1，只有当1重新创建ready后才会继续往下调度创建新的3,4,5

扩容：#kubectl scale –replicas=3 sts web

#kubectl get pod –show-lables

#kubectl get pod –l app=nginx -w ##watch，动态地看过程

#docker save buybox:1.28 –o busybox.tar ##镜像导出

#docker load –I nginx.tar ##镜像导入

# 1.1.4 StatefulSet 更新策略

* updateStrategy：type
* rollingUpdate:
  + - #kubectl edit sts web
    - #kubectl get pod 或者 #kubectl get pod –l app=nginx -w 显示已经倒序更新，从n-1开始更新至0，如果中途有低序号的pod有问题，会等ready后继续更新
    - #kubectl rollout status sts web
    - #kubect describe po web-2
* OnDelete:

#kubectl delete pod web-02

##kubectl get po web-2 –oyaml |grep image ##删除了2，只有2做了更改；如果要改变，必须先删除

# 1.1.5 StatefulSet 灰度发布

* updateStrategy：

rollingUpdate：

partition：3 ###更新>3的pod

例如现在有5个pod pod-0, pod-1, pod-2, pod-3, pod-4, 会更新pod-4,可以切部分流量过来看效果，做灰度发布

#kubectl edit sts web

#kubectl get pod –oyaml |grep image

# 1.1.6 StatefulSe级联删除和非级联删除

StatefulSe级联删除：删除sts时同时删除Pod，default

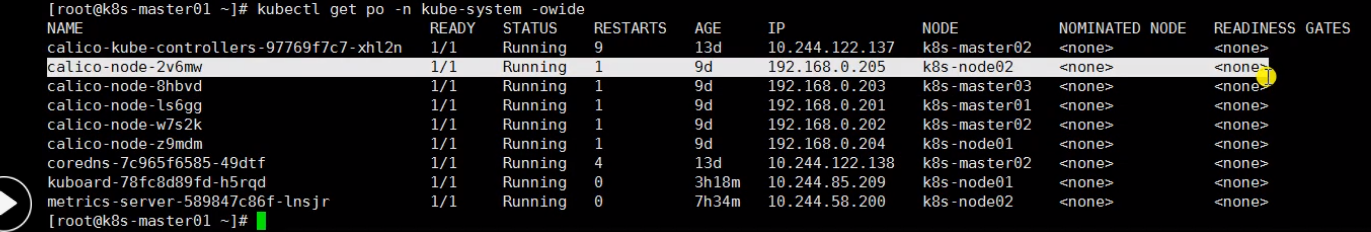
StatefulSe非级联删除：删除sts时不删除Pod，几乎用不到

#kubectl delete sts web –cascade=false删除sts时不删除Pod,此时Pod变成了孤儿Pod，删除Pod不会被重建

# 1.2.1 守护进程服务DeamonSet

DeamonSet： 守护进程集，缩写为ds，在所有节点或者是匹配的节点上都部署一个Pod。通过标签匹配。

例如calico Pod在每个节点上都有一个用来提供网络服务，就是用DeamonSet部署的



运行集群存储的deamon，例如ceph或者glusterd，节点的CNI网络插件，calico

节点日志的收集：fluentd或者是filebeat，

节点的监控： node exporter

服务暴露：ingress-nginx

# 1.2.2 DeamonSet的使用

* **cp nginx-dpl.yaml nginx-ds.yaml**
  + 1 Kind: DeamonSet
  + 2 副本数概念没有用，创建1个
  + 3上面的rollingupdate删除
* **#kubectl label node kus-node01 k8s-node02 ds=true ###label可以给任意资源打标签**
* **yaml写打标签**
  + vim nginx-ds.yaml
  + nodeSelector(与container是平级)：
  + ds: “true”
* 替换yaml文件看结果
  + #kubectl replace –f nginx-ds.yaml
  + **# kubectl get pod** ###不符合标签的pod会被删掉；给节点加label立马会创建相应pod
  + #kubectl rollout history ds nginx

# 1.2.3 DeamonSet的回滚和更新：建议使用OnDelete

#kubectl set image ds nginx nginx=nginx:1.15.3 –record

vim

updateStrategy：

rollingUpdate:

maxUnavaliable: 1 ##建议设置成1，因为DeamonSet涉及到的pod太多，且建议不使用rollingUpdate，而用type：OnDelete，就是删一个更新一个

#kubectl set image ds nginx nginx=nginx:1.15.2 –record

#kubectl delete po nginx-xxxx

#kubectl get po –owide

#kubectl rollout history ds nainx

# HPA

## 1.1HPA概念

Horizontal Pod Autoscaler: Pod的水平自动伸缩器

观察Pod的CPU、内存使用率自动扩展或缩容Pod的数量

不适用与无所缩放的对象，比如DeamonSet，一般用于Deployment

CPU：动态地、内存：一般不用，因为内存扩了没办法

自定义指标的扩缩容，给开发说暴露出来接口，比如流量到1000，扩容(实战篇)

## 1.2必须定义 Request参数，必须安装 metrics-server

#Kubectl get po –n Kube-system ###check是否存在pod metrics-server，前题有pod ：metrics-server-xxxx

#Kubectl top po –n Kube-system ###存在才有top命令

根据vim xxx.yaml里面定义的request的值去判断（这个参数必须有）

## 1.3部署或者使用HPA，一般出现在数据库上

#kubectl autoscale deploy demo-nginx –cpu-percent=20 –min=2 –max=5 ###cpu降低后缩减至最小值2个，最多扩5个，用节点亲和力，使得扩的pod不在同一个node上

#kubectl get hpa ####tartet 10%/20% ：现在/触发值

#kubectl describe hpa

触发：写死循环

#while true； do wget –q –o <https://svcip> > /dev/null ;done ###-q –o 是只请求不下载页面

#kubectl get pod

#kubectl logs –f pod-name

#Kubectl top po