### 2.2.7 StatefulSet

StatefulSet（有状态集）常用于部署有状态的且需要有序启动的应用程序。

2.2.7.1 StatefulSet的基本概念

StatefulSet主要用于管理有状态应用程序的工作负载API对象。比如在生产环境中，可以部署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。

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一般的格式为

statefulSetName-{0..N-1}.serviceName.namespace.svc.cluster. local。

说明：

* serviceName为Headless Service的名字。
* 0..N-1为Pod所在的序号，从0开始到N-1。
* statefulSetName为StatefulSet的名字。
* namespace为服务所在的命名空间。
* .cluster.local为Cluster Domain（集群域）。

比如，一个Redis主从架构，Slave连接Master主机配置就可以使用不会更改的Master的Headless Service，例如Redis从节点（Slave）配置文件如下：

port 6379

slaveofredis-sentinel-master-ss-0.redis-sentinel-master-ss.public-service.svc.cluster.local 6379

tcp-backlog 511

timeout 0

tcp-keepalive 0

……

其中，redis-sentinel-master-ss-0.redis-sentinel-master-ss.public-service.svc.cluster.local是Redis Master的Headless Service。

2.2.7.2 StatefulSet组件

定义一个简单的StatefulSet的示例如下：

apiVersion: v1

kind: Service

metadata:

name: nginx

labels:

app: nginx

spec:

ports:

- port: 80

name: web

clusterIP: None

selector:

app: nginx

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: web

spec:

serviceName: "nginx"

replicas: 2

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

name: web

其中，

* kind: Service定义了一个名字为Nginx的Headless Service，创建的Service格式为nginx-0.nginx.default.svc.cluster.local，其他的类似，因为没有指定Namespace（命名空间），所以默认部署在default。
* kind: StatefulSet定义了一个名字为web的StatefulSet，replicas表示部署Pod的副本数，本实例为2。

2.2.7.3 创建StatefulSet

创建StatefulSet：

[root@k8s-master01 2.2.7]# kubectl create -f sts-web.yaml

service/nginx created

statefulset.apps/web created

[root@k8s-master01 2.2.7]# kubectl get sts

NAME DESIRED CURRENT AGE

web 2 2 12s

[root@k8s-master01 2.2.7]# kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 7d2h

nginx ClusterIP None <none> 80/TCP 16s

[root@k8s-master01 2.2.7]# kubectl get po -l app=nginx

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 2m5s

web-1 1/1 Running 0 115s

2.2.7.4 StatefulSet扩容和缩容

和Deployment类似，可以通过更新replicas字段扩容/缩容StatefulSet，也可以使用kubectlscale或者kubectlpatch来扩容/缩容一个StatefulSet。

1. 扩容

将上述创建的sts副本增加到5个（扩容之前必须保证有创建完成的静态PV，动态PV和emptyDir）：

[root@k8s-master01 2.2.7]# kubectl scale sts web --replicas=5

statefulset.apps/web scaled

查看Pod状态：

[root@k8s-master01 2.2.7]# kubectl get po

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 2m58s

web-1 1/1 Running 0 2m48s

web-2 1/1 Running 0 116s

web-3 1/1 Running 0 79s

web-4 1/1 Running 0 53s

也可使用以下命令动态查看：

kubectl get pods -w -l app=nginx

2. 缩容

在一个终端动态查看：

[root@k8s-master01 2.2.7]# kubectl get pods -w -l app=nginx

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 4m37s

web-1 1/1 Running 0 4m27s

web-2 1/1 Running 0 3m35s

web-3 1/1 Running 0 2m58s

web-4 1/1 Running 0 2m32s

在另一个终端将副本数改为3：

[root@k8s-master01 ~]# kubectl patch sts web -p '{"spec":{"replicas":3}}'

statefulset.apps/web patched

此时可以看到第一个终端显示web-4和web-3的Pod正在被删除（或终止）：

[root@k8s-master01 2.2.7]# kubectl get pods -w -l app=nginx

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 4m37s

web-1 1/1 Running 0 4m27s

web-2 1/1 Running 0 3m35s

web-3 1/1 Running 0 2m58s

web-4 1/1 Running 0 2m32s

web-0 1/1 Running 0 5m8s

web-0 1/1 Running 0 5m11s

web-4 1/1 Terminating 0 3m36s

web-4 0/1 Terminating 0 3m38s

web-4 0/1 Terminating 0 3m47s

web-4 0/1 Terminating 0 3m47s

web-3 1/1 Terminating 0 4m13s

web-3 0/1 Terminating 0 4m14s

web-3 0/1 Terminating 0 4m22s

web-3 0/1 Terminating 0 4m22s

2.2.7.5 更新策略

1. On Delete策略

OnDelete更新策略实现了传统（1.7版本之前）的行为，它也是默认的更新策略。当我们选择这个更新策略并修改StatefulSet的.spec.template字段时，StatefulSet控制器不会自动更新Pod，我们必须手动删除Pod才能使控制器创建新的Pod。

2. RollingUpdate策略

RollingUpdate（滚动更新）更新策略会更新一个StatefulSet中所有的Pod，采用与序号索引相反的顺序进行滚动更新。

比如Patch一个名称为web的StatefulSet来执行RollingUpdate更新：

[root@k8s-master01 2.2.7]# kubectl patch statefulset web -p '{"spec":{"updateStrategy":{"type":"RollingUpdate"}}}'

statefulset.apps/web patched

查看更改后的StatefulSet：

[root@k8s-master01 2.2.7]# kubectl get sts web -o yaml | grep -A 1 "updateStrategy"

updateStrategy:

type: RollingUpdate

然后改变容器的镜像进行滚动更新：

[root@k8s-master01 2.2.7]# kubectl patch statefulset web --type='json' -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/image", "value":"dotbalo/canary:v1"}]'

statefulset.apps/web patched

如上所述，StatefulSet里的Pod采用和序号相反的顺序更新。在更新下一个Pod前，StatefulSet控制器会终止每一个Pod并等待它们变成Running和Ready状态。在当前顺序变成Running和Ready状态之前，StatefulSet控制器不会更新下一个Pod，但它仍然会重建任何在更新过程中发生故障的Pod，使用它们当前的版本。已经接收到请求的Pod将会被恢复为更新的版本，没有收到请求的Pod则会被恢复为之前的版本。

在更新过程中可以使用 kubectl rollout status sts/<name> 来查看滚动更新的状态：

[root@k8s-master01 2.2.7]# kubectl rollout status sts/web

Waiting for 1 pods to be ready...

waiting for statefulset rolling update to complete 1 pods at revision web-56b5798f76...

Waiting for 1 pods to be ready...

Waiting for 1 pods to be ready...

waiting for statefulset rolling update to complete 2 pods at revision web-56b5798f76...

Waiting for 1 pods to be ready...

Waiting for 1 pods to be ready...

statefulset rolling update complete 3 pods at revision web-56b5798f76...

查看更新后的镜像：

[root@k8s-master01 2.2.7]# for p in 0 1 2; do kubectl get po web-$p --template '{{range $i, $c := .spec.containers}}{{$c.image}}{{end}}'; echo; done

dotbalo/canary:v1

dotbalo/canary:v1

dotbalo/canary:v1

3. 分段更新

StatefulSet可以使用RollingUpdate更新策略的partition参数来分段更新一个StatefulSet。分段更新将会使StatefulSet中其余的所有Pod（序号小于分区）保持当前版本，只更新序号大于等于分区的Pod，利用此特性可以简单实现金丝雀发布（灰度发布）或者分阶段推出新功能等。注：金丝雀发布是指在黑与白之间能够平滑过渡的一种发布方式。

比如我们定义一个分区"partition":3，可以使用patch直接对StatefulSet进行设置：

# kubectl patch statefulset web -p '{"spec":{"updateStrategy":{"type":"RollingUpdate","rollingUpdate":{"partition":3}}}}'

statefulset "web" patched

然后再次patch改变容器的镜像：

# kubectl patch statefulset web --type='json' -p='[{"op": "replace", "path": "/spec/template/spec/containers/0/image", "value":"k8s.gcr.io/nginx-slim:0.7"}]'

statefulset "web" patched

删除Pod触发更新：

kubectl delete po web-2

pod "web-2" deleted

此时，因为Pod web-2的序号小于分区3，所以Pod不会被更新，还是会使用以前的容器恢复Pod。

将分区改为2，此时会自动更新web-2（因为之前更改了更新策略），但是不会更新web-0和web-1：

# kubectl patch statefulset web -p '{"spec":{"updateStrategy":{"type":"RollingUpdate","rollingUpdate":{"partition":2}}}}'

statefulset "web" patched

按照上述方式，可以实现分阶段更新，类似于灰度/金丝雀发布。查看最终的结果如下：

[root@k8s-master01 2.2.7]# for p in 0 1 2; do kubectl get po web-$p --template '{{range $i, $c := .spec.containers}}{{$c.image}}{{end}}'; echo; done

dotbalo/canary:v1

dotbalo/canary:v1

dotbalo/canary:v2

2.2.7.6 删除StatefulSet

删除StatefulSet有两种方式，即级联删除和非级联删除。使用非级联方式删除StatefulSet时，StatefulSet的Pod不会被删除；使用级联删除时，StatefulSet和它的Pod都会被删除。

1. 非级联删除

使用kubectldeletestsxxx删除StatefulSet时，只需提供--cascade=false参数，就会采用非级联删除，此时删除StatefulSet不会删除它的Pod：

[root@k8s-master01 2.2.7]# kubectl get po

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 16m

web-1 1/1 Running 0 16m

web-2 1/1 Running 0 11m

[root@k8s-master01 2.2.7]# kubectl delete statefulset web --cascade=false

statefulset.apps "web" deleted

[root@k8s-master01 2.2.7]# kubectl get sts

No resources found.

[root@k8s-master01 2.2.7]# kubectl get po

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 16m

web-1 1/1 Running 0 16m

web-2 1/1 Running 0 11m

由于此时删除了StatefulSet，因此单独删除Pod时，不会被重建：

[root@k8s-master01 2.2.7]# kubectl get po

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 16m

web-1 1/1 Running 0 16m

web-2 1/1 Running 0 11m

[root@k8s-master01 2.2.7]# kubectl delete po web-0

pod "web-0" deleted

[root@k8s-master01 2.2.7]# kubectl get po

NAME READY STATUS RESTARTS AGE

web-1 1/1 Running 0 18m

web-2 1/1 Running 0 12m

当再次创建此StatefulSet时，web-0会被重新创建，web-1由于已经存在而不会被再次创建，因为最初此StatefulSet的replicas是2，所以web-2会被删除，如下（忽略AlreadyExists错误）：

[root@k8s-master01 2.2.7]# kubectl create -f sts-web.yaml

statefulset.apps/web created

Error from server (AlreadyExists): error when creating "sts-web.yaml": services "nginx" already exists

[root@k8s-master01 2.2.7]# kubectl get po

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 32s

web-1 1/1 Running 0 19m

2. 级联删除

省略--cascade=false参数即为级联删除：

[root@k8s-master01 2.2.7]# kubectl delete statefulset web

statefulset.apps "web" deleted

[root@k8s-master01 2.2.7]# kubectl get po

No resources found.

也可以使用-f参数直接删除StatefulSet和Service（此文件将sts和svc写在了一起）：

[root@k8s-master01 2.2.7]# kubectl delete -f sts-web.yaml

service "nginx" deleted

Error from server (NotFound): error when deleting "sts-web.yaml": statefulsets.apps "web" not found

[root@k8s-master01 2.2.7]#