## 有状态应用管理StatefulSet

StatefulSet（有状态集，缩写为sts）常用于部署有状态的且需要有序启动的应用程序，比如在进行SpringCloud项目容器化时，Eureka的部署是比较适合用StatefulSet部署方式的，可以给每个Eureka实例创建一个唯一且固定的标识符，并且每个Eureka实例无需配置多余的Service，其余Spring Boot应用可以直接通过Eureka的Headless Service即可进行注册。

Eureka的statefulset的资源名称是eureka，eureka-0 eureka-1 eureka-2

Service：headless service，没有ClusterIP eureka-svc

Eureka-0.eureka-svc.NAMESPACE\_NAME eureka-1.eureka-svc …

### 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。

和Deployment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，StatefulSet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，在重新调度时也会保留，一般格式为StatefulSetName-Number。比如定义一个名字是Redis-Sentinel的StatefulSet，指定创建三个Pod，那么创建出来的Pod名字就为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一般的格式为：

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

说明：

* serviceName为Headless Service的名字，创建StatefulSet时，必须指定Headless Service名称；
* 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）。

假如公司某个项目需要在Kubernetes中部署一个主从模式的Redis，此时使用StatefulSet部署就极为合适，因为StatefulSet启动时，只有当前一个容器完全启动时，后一个容器才会被调度，并且每个容器的标识符是固定的，那么就可以通过标识符来断定当前Pod的角色。

比如用一个名为redis-ms的StatefulSet部署主从架构的Redis，第一个容器启动时，它的标识符为redis-ms-0，并且Pod内主机名也为redis-ms-0，此时就可以根据主机名来判断，当主机名为redis-ms-0的容器作为Redis的主节点，其余从节点，那么Slave连接Master主机配置就可以使用不会更改的Master的Headless Service，此时Redis从节点（Slave）配置文件如下：

port 6379

slaveof redis-ms-0.redis-ms.public-service.svc.cluster.local 6379

tcp-backlog 511

timeout 0

tcp-keepalive 0

……

其中redis-ms-0.redis-ms.public-service.svc.cluster.local是Redis Master的Headless Service，**在同一命名空间下只需要写redis-ms-0.redis-ms即可，后面的public-service.svc.cluster.local可以省略**。

### StatefulSet注意事项

一般StatefulSet用于有以下一个或者多个需求的应用程序：

* 需要稳定的独一无二的网络标识符。
* 需要持久化数据。
* 需要有序的、优雅的部署和扩展。
* 需要有序的自动滚动更新。

如果应用程序不需要任何稳定的标识符或者有序的部署、删除或者扩展，应该使用无状态的控制器部署应用程序，比如Deployment或者ReplicaSet。

StatefulSet是Kubernetes 1.9版本之前的beta资源，在1.5版本之前的任何Kubernetes版本都没有。

Pod所用的存储必须由PersistentVolume Provisioner（持久化卷配置器）根据请求配置StorageClass，或者由管理员预先配置，当然也可以不配置存储。

为了确保数据安全，删除和缩放StatefulSet不会删除与StatefulSet关联的卷，可以手动选择性地删除PVC和PV（关于PV和PVC请参考2.2.12节）。

StatefulSet目前使用Headless Service（无头服务）负责Pod的网络身份和通信，需要提前创建此服务。

删除一个StatefulSet时，不保证对Pod的终止，要在StatefulSet中实现Pod的有序和正常终止，可以在删除之前将StatefulSet的副本缩减为0。

### 定义一个StatefulSet资源文件

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

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx

ports:

- containerPort: 80

name: web

注意

此示例没有添加存储配置，后面的章节会单独讲解存储相关的知识点

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。

在StatefulSet中必须设置Pod选择器（.spec.selector）用来匹配其标签（.spec.template.metadata.labels）。在1.8版本之前，如果未配置该字段（.spec.selector），将被设置为默认值，在1.8版本之后，如果未指定匹配Pod Selector，则会导致StatefulSet创建错误。

当StatefulSet控制器创建Pod时，它会添加一个标签statefulset.kubernetes.io/pod-name，该标签的值为Pod的名称，用于匹配Service。

### 创建一个StatefulSet

[root@k8s-master01 ~]# # 创建一个sts

[root@k8s-master01 ~]# kubectl create -f nginx-sts.yaml # -n namespace\_name

service/nginx created

statefulset.apps/web created

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

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

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 14d

nginx ClusterIP None <none> 80/TCP <invalid>

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

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 24s

web-1 1/1 Running 0 19s

[root@k8s-master01 ~]# # 更改副本数为3

[root@k8s-master01 ~]# kubectl scale --replicas=3 sts web

[root@k8s-master01 ~]# # 查看Pod

[root@k8s-master01 ~]#

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

NAME READY STATUS RESTARTS AGE

web-0 1/1 Running 0 58s

web-1 1/1 Running 0 53s

web-2 0/1 ContainerCreating 0 13s

[root@k8s-master01 ~]# # 解析无头service

cat<<EOF | kubectl apply -f -

apiVersion: v1

kind: Pod

metadata:

name: busybox

namespace: default

spec:

containers:

- name: busybox

image: busybox:1.28

command:

- sleep

- "3600"

imagePullPolicy: IfNotPresent

restartPolicy: Always

EOF

[root@k8s-master01 ~]# kubectl exec -ti busybox -- sh

/ # ls

bin dev etc home proc root sys tmp usr var

/ # nslookup web-0.nginx

Server: 10.96.0.10

Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name: web-0.nginx

Address 1: 10.244.32.152 web-0.nginx.default.svc.cluster.local

/ # exit

### StatefulSet更新策略

[root@k8s-master01 ~]# kubectl get sts web -o yaml

apiVersion: apps/v1

kind: StatefulSet

metadata:

creationTimestamp: "2020-09-19T07:46:49Z"

generation: 5

name: web

namespace: default

spec:

podManagementPolicy: OrderedReady

replicas: 2

revisionHistoryLimit: 10

selector:

matchLabels:

app: nginx

serviceName: nginx

template:

metadata:

creationTimestamp: null

labels:

app: nginx

spec:

containers:

- image: nginx:1.15.2

imagePullPolicy: IfNotPresent

name: nginx

ports:

- containerPort: 80

name: web

protocol: TCP

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

dnsPolicy: ClusterFirst

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

**updateStrategy:**

**rollingUpdate:**

**partition: 0**

**type: RollingUpdate**