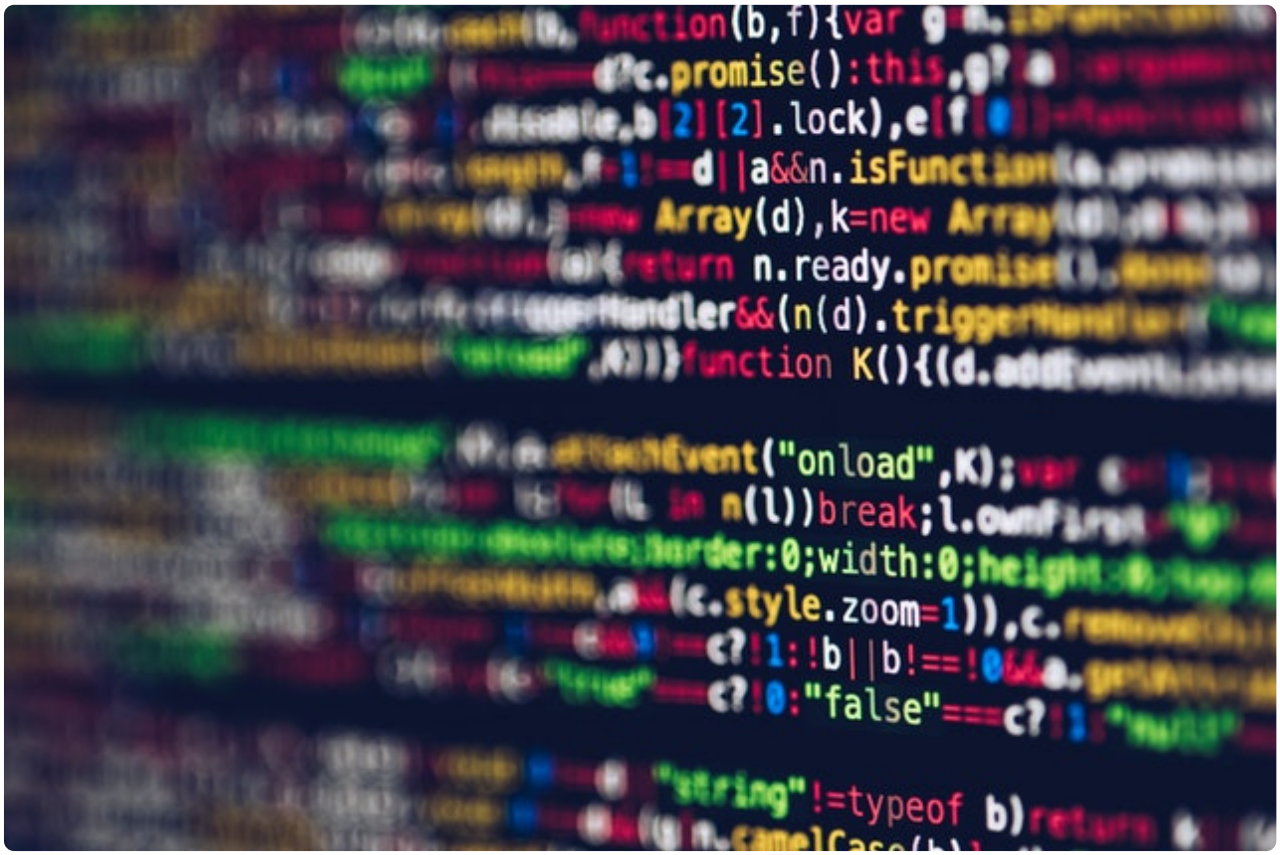
40 Kubernetes 有状态应用管理 StatefulSet

更新时间：2020-10-28 09:47:46



> 先相信你自己，然后别人才会相信你。——屠格涅夫

在前面的文章中我们介绍了 Deployment，对于常规应用，我们都可以通过 Deployment 来编排和管理，但是 Deployment 有一个致命的缺陷就是不能用来管理有状态的应用，因为它认为所代理的后端的 Pod 都是相同的。

那么什么叫做有状态的应用呢？简单来说就是后端的多个 Pod 的角色是不同的，比如在分布式应用程序中，很可能会存在主从关系，比如 Zookeeper 集群有一个节点的角色是 Leader，剩下都是 Follower。除此之外，还有一种状态是存储和应用之间是绑定的，比如 Hadoop 中的 HDFS 会启动很多个 DataNode，每个 DataNode 存储的数据都是有区别的，也就是说我们不能使用 Deployment 控制 Pod 的方式来管理 DataNode。这个时候就引入了 StatefulSet。

1. StatefulSet 简介

容器最适合的场景是用来封装和管理无状态应用，比如 Web 服务。对于分布式场景下的有状态应用，使用容器需要很多额外的代价。Kubernetes 也是最先之初也是最先支持无状态应用，对于有状态应用也是在 Deployment 的基础上扩展出来的，也就是 StatefulSet。

StatefulSet 的应用场景，对于需要满足以下一个或多个需求的应用程序都可以尝试引入 StatefulSet：

- 稳定的、唯一的网络标识符；
- 稳定的、持久的存储；

- 有序的、优雅的部署和缩放；
- 有序的、自动的滚动更新。

2. StatefulSet 设计思想

StatefulSet 的设计思想是将有状态应用中的状态做了一层抽象，抽象成两种情况：

- **网络拓扑**：所谓网络拓扑是说应用的多个 Pod 实例之间不是完全的对等的关系，它们可能存在某种依赖关系，比如同一个应用有三个 Pod：A、B、C，启动顺序必须严格的按 A -> B -> C 的顺序启动。对于网络拓扑状态。
- **存储**：存储的意思是应用的多个 Pod 实例对应的存储数据是不同的。如果 Pod 正常运行，那么它对应的存储是一直不变的（存储是可靠的）；但是如果 Pod 被重启，这个时候存储就可能出问题，这就是存储状态需要解决的核心问题。

3. 创建一个 StatefulSet

要创建一个 StatefulSet 资源，需要提供三个组件：

- Headless Service，保证每个 Pod 都有唯一的网络标识；
- PV，可以认为是 Kubernetes 中的存储，声明存储的时候我们只需要声明一个 PVC 即可；
- StatefulSet 对象，也就是 StatefulSet 控制器。

```yaml
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:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: nginx:1.9.1
        ports:
        - containerPort: 80
          name: web
        resources:
          limits:
            cpu: 100m
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "alicloud-disk-efficiency"
      resources:
        requests:
          storage: 1Gi
```

从上面的资源声明文件中，我们可以看到

- 最先声明的 Service 对象是一个 Headless Service 对象（将 spec.clusterIP 设置为 None）；
- 然后是一个 StatefulSet 的对象声明。在 StatefulSet 的对象中有一个字段 spec.serviceName 可以帮助我们区分 Deployment；
- 在 StatefulSet 的声明最后是 PVC 的声明，这里使用了名字为 alicloud-disk-efficiency 的 storageClass，这个依赖于云厂商。这里的 yaml 中的 PVC 声明只是举个例子，由于 PV 需要更多的资源，我们这里就不演示了，在下面正式部署的时候我们先将 PV 注释掉。

部署之后，我们通过 kubectl get 来查看我们部署的 StatefulSet。

```
□ kubectl get statefulset -n imooc
NAME   READY   AGE
web    3/3     7s
```

然后我们再通过 kubectl describe 来看一下 StatefulSet 的明细信息。

```
□ kubectl describe statefulset web -n imooc
Name:               web
Namespace:          imooc
CreationTimestamp:  Sun, 24 May 2020 22:15:29 +0800
Selector:           app=nginx
Labels:             <none>
Annotations:        kubectl.kubernetes.io/last-applied-configuration:
                      {"apiVersion":"apps/v1","kind":"StatefulSet","metadata":{"annotations":{},"name":"web","namespace":"imooc"},"spec":{"replicas":3,"selector..
.
Replicas:           3 desired | 3 total
Update Strategy:    RollingUpdate
  Partition:        824642172956
Pods Status:        3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=nginx
  Containers:
   nginx:
    Image:      nginx:1.9.1
    Port:       80/TCP
    Host Port:  0/TCP
    Limits:
      cpu:     100m
      memory:  200Mi
    Requests:
      cpu:        100m
      memory:     200Mi
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Volume Claims:    <none>
Events:
  Type    Reason            Age   From                    Message
  ----    ------            ----  ----                    -------
  Normal  SuccessfulCreate  46m   statefulset-controller  create Pod web-0 in StatefulSet web successful
  Normal  SuccessfulCreate  46m   statefulset-controller  create Pod web-1 in StatefulSet web successful
  Normal  SuccessfulCreate  46m   statefulset-controller  create Pod web-2 in StatefulSet web successful
```

从 StatefulSet 的 Events 中我们可以看到这个 StatefulSet 创建了三个 Pod，分别为：web-0，web-1，web-2，也就是所有的 Pod 都是 StatefulSet 的名字外加一个序号生成。

下面演示一下 Pod 重启的变化。

```
□ statefulset kubectl get pods -n imooc -o wide | grep web
web-0                   1/1     Running   0          50m   10.1.2.34    cn-beijing.172.16.60.187   <none>           <none>
web-1                   1/1     Running   0          50m   10.1.2.35    cn-beijing.172.16.60.187   <none>           <none>
web-2                   1/1     Running   0          50m   10.1.1.155   cn-beijing.172.16.60.188   <none>           <none>
```

我们通过 kubectl delete 删除 web-1 这个 Pod，然后等 1 分钟看一下自动拉起的 Pod 名称。

```
kubectl get pods -n imooc -o wide | grep web
web-0                   1/1     Running   0          53m   10.1.2.34    cn-beijing.172.16.60.187   <none>           <none>
web-1                   1/1     Running   0          3s    10.1.2.162   cn-beijing.172.16.60.186   <none>           <none>
web-2                   1/1     Running   0          53m   10.1.1.155   cn-beijing.172.16.60.188   <none>           <none>
```

我们可以看到尽管新的 Pod 的 IP 和运行的 Node 已经发生变化，但是 Pod 的名字还是一样，有的人可能会问这样能保证网络状态不变了。别着急，我们来看一下上面 StatefulSet 绑定的 Headless Service。

```
/ $ nslookup nginx

Name:      nginx
Address 1: 10.1.2.34 web-0.nginx.imooc.svc.cluster.local
Address 2: 10.1.1.155 web-2.nginx.imooc.svc.cluster.local
Address 3: 10.1.2.162 web-1.nginx.imooc.svc.cluster.local
```

没错，Headless Service 返回的直接是三个 Pod 的 DNS 条目，后面的名称形式为 web-0.nginx.imooc.svc.cluster.local 使用了 Pod 的名字，由于 Pod 每次重启名称都不变也就是 DNS 条目这里一直可以找到。

除此之外，StatefulSet 还会为每个 Pod 自动生成一个特定的 label。我们可以通过 `kubectl describe` 来查看，其中 `statefulset.kubernetes.io/pod-name=web-0` 就是自动生成的 label。

```
□ kubectl describe pods web-0 -n imooc
Name:         web-0
Namespace:    imooc
Priority:     0
Node:         cn-beijing.172.16.60.187/172.16.60.187
Start Time:   Sun, 24 May 2020 22:15:29 +0800
Labels:       app=nginx
              controller-revision-hash=web-5d5bd68b9b
              statefulset.kubernetes.io/pod-name=web-0
Annotations:  <none>
Status:       Running
IP:           10.1.2.34
IPs:          <none>
Controlled By:  StatefulSet/web
Containers:
  nginx:
    Container ID:   docker://70764fed63d5305e7a0c85f430f774171f2d0e350c5945aa71477e2cde8a45e2
    Image:          nginx:1.9.1
    Image ID:       docker-pullable://nginx@sha256:2f68b99bc0d6d25d0c56876b924ec20418544ff28e1fb89a4c27679a40da811b
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Sun, 24 May 2020 22:15:30 +0800
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     100m
      memory:  200Mi
    Requests:
      cpu:        100m
      memory:     200Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-84db9 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-84db9:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-84db9
    Optional:    false
QoS Class:       Guaranteed
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:          <none>
```

4. Pod 标识

StatefulSet 的一个核心要点就是控制的 Pod 具有唯一的标识，包括顺序标识、稳定的网络标识和稳定的存储。Pod 启动之后标识就是固定的，包括之后重启都不会发生变化。

顺序标识

对于具有 N 个 Pod 副本的 StatefulSet，StatefulSet 会为每个 Pod 分配一个固定的名字，形如 <statefulset-name>-x，其中 x 介于 0 和 N-1 之间。正如我们上面看到的 Pod 的名称 `web-0`、`web-1` 、`web-2` 。

网络标识

StatefulSet 通过 Headless Service 控制 Pod 的网络标识，网络标识的格式为 `$(服务名称).$(命名空间).svc.cluster.local`，其中 `cluster.local` 是集群域。一旦 Pod 创建成功，就会得到一个匹配的 DNS 子域，格式为 `$(pod名称).$(所属服务的 DNS 域名)`，其中所属服务由 StatefulSet 的 spec 中的 serviceName 域来设定。由于 Pod 的名称的固定的，那么也就意味着每个 Pod 对应的 DNS 子域也是固定的。

稳定存储

Kubernetes 为每个 VolumeClaimTemplate 域创建一个 PV。在上面的示例中，每个 Pod 得到基于 StorageClass `alicloud-disk-efficiency` 提供的 5Gib 的 PV。如果没有声明 StorageClass，就会使用默认的 StorageClass。

当 Pod 被调度以及重新调度（比如 Pod 重启或者 Node 节点挂掉）到节点上时，它的 `volumeMounts` 会挂载与其 PersistentVolumeClaims 相关联的 PV。需要注意的是，当 Pod 或者 StatefulSet 被删除时，Pod 之前使用的 PV 并不会被自动删除，我们需要手动删除。

标签

通过上面的演示，我们可以看到 StatefulSet 创建 Pod 的时候会自动添加 Label `statefulset.kubernetes.io/pod-name`。通过这个标签，我们可以为 StatefulSet 中的特定 Pod 绑定一个 Service。

5. 部署和扩缩容

StatefulSet 的部署和扩缩容遵循如下约定：

- 对于包含 N 个 Pod 副本的 StatefulSet，当部署时，Pod 按 0，1，…，N-1 的顺序依次被创建的。
- 当删除 StatefulSet 时，Pod 按 N-1，…，1，0 的顺序被逆序终止的。
- 当缩放操作应用到某个 Pod 之前，它前面的所有 Pod 必须是 Running 和 Ready 状态，所谓前面的 Pod 的意思是序号小于当前 Pod 的序号。
- 在 Pod 终止之前，所有序号大于该 Pod 的 Pod 都必须完全关闭。

还是以我们上面的 nginx 的 StatefulSet 为例，在 StatefulSet 被部署时，Pod 按 web-0、web-1、web-2 的顺序被启动。在 web-0 进入 Running 或者 Ready 状态前不会启动 web-1 和 web-2。以此类推。如果 web-1 已经处于 Running 或者 Ready 状态，而 web-2 尚未部署，此时 web-0 运行失败，那么 web-2将不会被部署，要等到 web-0 恢复才会继续部署。

如果我们减小 StatefulSet 的副本数 replicas = 1，将首先终止 web-2，在 web-2 没有被完全停止和删除前，web-1 不会被终止。

6. 总结

本文介绍了 Kubernetes 针对有状态应用提出的解决方案 StatefulSet，StatefulSet 可以满足某些有状态应用的编排和部署。但是有些分布式应用复杂度极高，所以在 Kubernetes Operator 机制出来之后，很多分布式应用都开始提供自己的 Operator。

}