K8s StatefulSets 部署有状态应用

AUTHOR: 彭玲 TIME: 2022/2/24

StatefulSets 工作负载资源

StatefulSet 是用来管理 有状态应用 的工作负载 API 对象。

StatefulSet 用来管理某 Pod 集合的部署和扩缩，并保证这些 pod 的顺序和唯一性。StatefulSet 为这些 Pod 提供持久存储和持久标识符。StatefulSets 对于需要满足以下一个或多个需求的 应用程序 很有价值：

- 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 。

StatefulSet 示例

对于一个拥有 `n` 个副本的 StatefulSet，Pod 被部署时是按照 `{0..n-1}` 的序号顺序创建的。

前置条件

确认集群中默认的 StorageClass 是否 **由 PersistentVolume Provisioner 提供 or** `管理员` **预先提供**:

```
anxin@node38:~$ kubectl get sc
NAME              PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION   AGE
localhostpath       kubernetes.io/no-provisioner  Retain        Immediate
   false             47d
nfs-storage (default)  fuseim.pri/ifs           Retain        Immediate
 false             27h
```

yaml 配置清单

下面例子中，创建了一个`nginx`无头服务，用来发布 StatefulSet 对象`web`中 Pod 的 IP 地址。

```
anxin@node38:~/pengling/k8s/controller-demo$ vi statefulset-web.yaml

apiVersion: v1
kind: Service
metadata:
  name: nginx
  namespace: julin
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
  namespace: julin
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
        volumeMounts: # 保证了 /usr/share/nginx/html 文件夹由一个 PV 支持
```

```
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata: # 未指定 julin 命名空间
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi
```

创建

下面打开两个终端窗口。

- 在第一个终端中，查看 StatefulSet 的 Pods 的创建情况。

```
anxin@node38:~$ kubectl get pods -w -l app=nginx -n julin

NAME    READY    STATUS                  RESTARTS   AGE
web-0   0/1     Pending                 0          0s # web-0
web-0   0/1     Pending                 0          0s
web-0   0/1     Pending                 0          1s
web-0   0/1     ContainerCreating       0          2s
web-0   1/1     Running              0          10s
web-1   0/1     Pending             0          0s # web-1
web-1   0/1     Pending             0          0s
web-1   0/1     Pending             0          2s
web-1   0/1     ContainerCreating       0          2s
web-1   1/1     Running             0          12s
```

- 在另一个终端中，创建定义在 `web.yaml` 中的 Headless Service 和 StatefulSet。

```
anxin@node38:~/pengling/k8s/controller-demo$ kubectl apply -f statefulset-
web.yaml
service/nginx created
statefulset.apps/web created
```

验证

获取 `nginx` Service 和 `web` StatefulSet 来验证是否成功的创建了它们。

```
anxin@node38:~$ kubectl get service nginx -n julin
NAME    TYPE        CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
nginx   ClusterIP    None        <none>       80/TCP    3m42s
```

获取 `web` StatefulSet，以验证两者均已成功创建：

```
anxin@node38:~$ kubectl get statefulset web -n julin
NAME    READY    AGE
web    2/2     3m46s
```

注意：

> 在 `web-0` Pod 处于 `Running` 和 `Ready` 状态后 `web-1` Pod 才会被启动。

例如，该示例中应该创建 2 个 Pod，但下面的结果显示：`web-0` 为 `Pending` 异常状态，所以 `web-1` 未被创建。

```
anxin@node38:~$ kubectl get po -l app=nginx
NAME        READY     STATUS     RESTARTS    AGE
web-0       0/1       Pending    0           23m
```

在 `web-0` Pod 正常启动后，`web-1` Pod 会被启动：

```
anxin@node38:~$ kubectl get po -l app=nginx -n julin
NAME     READY     STATUS     RESTARTS    AGE
web-0    1/1       Running    0           4m43s
web-1    1/1       Running    0           4m31s
```

StatefulSet 中的 Pod

检查 Pod 的顺序索引

```
anxin@node38:~$ kubectl get pods -l app=nginx -n julin
NAME     READY     STATUS     RESTARTS    AGE
web-0    1/1       Running    0           12m
web-1    1/1       Running    0           11m
```

使用稳定的网络身份标识

每个 Pod 都拥有一个基于其顺序索引的 `稳定的` 主机名。使用 `kubectl exec` 在每个 Pod 中执行 `hostname` 命令。

```
anxin@node38:~$ for i in 0 1; do kubectl exec "web-$i" -n julin -- sh -c
'hostname'; done
web-0
web-1
```

使用 `kubectl run` 运行一个提供 `nslookup` 命令的容器。通过对 Pod 的主机名执行 `nslookup`，你可以检查他们在集群内部的 DNS 地址。

```
kubectl run -i --tty --image busybox:1.28 dns-test --restart=Never --rm
```

这将启动一个新的 shell。在新 shell 中，运行：

```
nslookup web-0.nginx # <pod_name>.<StatefulSet.spec.serviceName>.<namepace>
```

Headless Service 的 CNAME 指向 SRV 记录（记录每个 Running 和 Ready 状态的 Pod）。SRV 记录指向一个包含 Pod IP 地址的记录表项。完整过程如下：

```
anxin@node38:~$ kubectl run -it --image busybox:1.28 dns-test --
restart=Never --rm
If you don't see a command prompt, try pressing enter.
/ #
/ # nslookup web-0.nginx.julin
Server:    10.96.0.10
```

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

Name:      web-0.nginx.julin
Address 1:   10.244.1.52 web-0.nginx.julin.svc.cluster.local
/ #
/ # nslookup web-1.nginx.julin
Server:    10.96.0.10
Address 1:   10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      web-1.nginx.julin
Address 1:   10.244.2.44 web-1.nginx.julin.svc.cluster.local
```

写入稳定的存储

动态提供 PV

获取 `web-0` 和 `web-1` 的 PersistentVolumeClaims。

```
anxin@node38:~$ kubectl get pvc -l app=nginx -n julin
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES
  STORAGECLASS   AGE
www-web-0   Bound    pvc-7f3d201c-847f-4272-ae52-3c4132bf00a8   1Gi      RWO
      nfs-storage    48m
www-web-1   Bound    pvc-ccf60a35-1a29-476c-a722-bdc16f4fb249   1Gi      RWO
      nfs-storage    47m
```

StatefulSet 控制器创建了 2 个 PVC，绑定到 2 个 PersistentVolumes (动态提供 PV，所有的 PV 都是自动创建和绑定的)。

```
anxin@node38:~$ kubectl get pv
NAME                                       RECLAIM     POLICY          STATUS
pvc-7f3d201c-847f-4272-ae52-3c4132bf00a8     Bound     julin/www-web-0   nfs-
storage
pvc-ccf60a35-1a29-476c-a722-bdc16f4fb249     Bound     julin/www-web-1   nfs-
storage
```

Nginx 服务器文件

NGINX web 服务器默认会加载位于 `/usr/share/nginx/html/index.html` 的 index 文件。

> 所有 web 服务器始终使用它们的`主机名`提供服务。

即使`web-0`和`web-1`被重新调度了，但它们仍然继续监听各自的`主机名`，因为 PV 被重新挂载到了各自的`volumeMount`上。不管`web-0`和`web-1`被调度到了哪个节点上，它们的 PV 将会被挂载到合适的`挂载点`上。

```
# 1. 将 Pod 的主机名写入它们的`index.html`文件
anxin@node38:~$ for i in 0 1; do kubectl exec "web-$i" -n julin -- sh -c
'echo "$(hostname)" > /usr/share/nginx/html/index.html'; done
# 2. 验证 NGINX web 服务器使用该主机名提供服务
anxin@node38:~$ for i in 0 1; do kubectl exec -it "web-$i" -n julin -- curl
http://localhost/; done
web-0
web-1
```

StatefulSet 扩容/缩容

扩容/缩容 StatefulSet 指增加或减少它的副本数。这通过更新 `replicas` 字段完成。你可以使用 `kubectl scale` 或者 `kubectl patch` 来扩容/缩容一个 StatefulSet。

扩容

使用 `kubectl scale` 扩展副本数为 `5`。

```
anxin@node38:~$ kubectl scale sts web --replicas=5 -n julin
statefulset.apps/web scaled
```

缩容

使用 `kubectl patch` 将 StatefulSet 缩容回 3 个副本。

```
anxin@node38:~$ kubectl patch sts web -p '{"spec":{"replicas":3}}' -n julin
statefulset.apps/web patched
```

清理工作

1. 顺序终止 Pod

控制器会按照与 Pod 序号索引相反的顺序每次删除一个 Pod。在删除下一个 Pod 前会等待上一个被完全关闭。

2. 删除 PVC 和 PV

当删除 StatefulSet 的 Pod 时，挂载到 StatefulSet 的 Pod 的 PersistentVolumes 不会被删除。

这种 `删除行为` 由 StatefulSet 缩容引起时，也是一样的。

Pod 管理策略

对于某些分布式系统来说，StatefulSet 的顺序性保证是不必要和/或者不应该的。这些系统仅仅要求唯一性和身份标志。

为了解决这个问题，在 Kubernetes 1.7 中 我们针对 StatefulSet API 对象引入了 `.spec.podManagementPolicy`。

`.spec.podManagementPolicy` 选项仅影响 `扩缩操作` 的行为，更新不受影响。

OrderedReady Pod 管理

`OrderedReady` pod 管理策略是 StatefulSets 的**默认选项**。它告诉 StatefulSet 控制器遵循上文展示的 `顺序性` 保证。

Parallel Pod 管理

`Parallel` pod 管理策略告诉 StatefulSet 控制器 `并行` 终止所有 Pod， 在启动或终止另一个 Pod 前，不必等待这些 Pod 变成 Running 和 Ready 或者完全终止状态。例如，下面的例子：

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  ...
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
  namespace: julin
spec:
  serviceName: "nginx"
  podManagementPolicy: "Parallel" # Pod 管理策略 (The default policy is `OrderedReady`)
  ...
```