

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

# Kubernetes 控制器 StatefulSet

前言：  
课程名称：Kubernetes StatefulSet 控制器

实验环境：  
本章节 Kubernetes 集群环境如下：

角色	IP	主机名	组件	硬件
控制节点	192.168.128.11	k8s-master01	apiserver controller-manager scheduler etcd containerd	CPU：4vCPU 硬盘：100G 内存：4GB 开启虚拟化
工作节点	192.168.128.21	k8s-node01	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化
工作节点	192.168.128.22	k8s-node02	kubelet kube-proxy containerd calico coredns	CPU：6vCPU 硬盘：100G 内存：8GB 开启虚拟化

张岩峰老师微信，加我微信，邀请你加入 VIP 交流答疑群：  
微信号：ZhangYanFeng0429  
二维码：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



## 1、初识 StatefulSet 控制器

### 1.1、什么是 StatefulSet 控制器

#### ● Statefulset 控制器介绍

StatefulSet 是 Kubernetes 中的一种控制器，用于管理有状态应用程序的部署。

相比于 Deployment 控制器，StatefulSet 控制器允许有序且唯一的命名，每个 Pod 可以绑定到不同的持久卷，有序的进行启动和停止，以及有序的进行水平扩展和缩小。这使得 StatefulSet 控制器适合运行有状态应用程序，如数据库、缓存和消息队列等应用。

#### ● 什么是有状态服务？什么是无状态服务？

有状态服务指的是需要保存或维护某种状态的服务。通常涉及到数据的持久性存储，例如数据库、缓存、消息队列、文件系统等。因为这些服务需要存储和维护数据，所以它们需要访问持久存储设备并且要确保高可用性和数据的一致性。

无状态服务则相反，它们不需要维护状态或数据。例如 Web 服务器、负载均衡器等。这些服务需要处理请求并将其发送到其他服务的组件，但不需要维护客户端会话或其他状态。

通过区分有状态服务和无状态服务，可以更好地设计和部署应用程序的不同组件。例如，有状态服务需要使用持久卷和控制器（例如 StatefulSet），而无状态服务则可以使用更简单的部署工具（例如 Deployment）来实现高可用性和水平扩展。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

## ● StatefulSet 组成

- 1、控制器 (Controller): 控制和管理运行 StatefulSet 的 Pod。
- 2、持久存储 (Persistent Storage): StatefulSet 控制器使用持久卷将数据保存在存储节点。
- 3、Headless Service: 用来定义 pod 网路标识，生成可解析的 DNS 记录

## ● 什么是 Headless service?

Headless Service 没有固定的 Cluster IP 地址，而是直接返回 Pod 的 DNS 名称列表。这种服务方式称为“headless”，因为它不会负责监听对应的 DNS 名称，也不存在虚拟 IP 地址，所有的访问请求都会直接转发到 Pod 上。这个功能非常适合于对于需要访问特定的 Pod 的应用程序，例如一些分布式系统。

Headless Service 与普通 Service 的最大区别在于，它返回的不是一个 Cluster IP，而是一个 DNS 名称列表。使用 Headless Service 时，需要设置 Service 的 clusterIP 字段为空，在 Service 的 spec 字段中设置 clusterIP: None。通过这种方式创建的 Headless Service，会在 DNS 服务器中注册一个域名，用来识别 Service 名称。对于每个 Service 名称，Kubernetes 将自动生成一个与它匹配的域名，形式为：“my-service.my-namespace.svc.cluster.local”，其中，“my-service”是 Headless Service 的名称，“my-namespace”是 Headless Service 所处的命名空间。

StatefulSet 会为关联的 Pod 分配一个 dnsName:

`$<Pod Name>.$<service name>.$<namespace name>.svc.cluster.local`

总之，Headless Service 是一种用于访问 Kubernetes 集群内部的 Pod 的服务方式，适用于需要遍历整个 Pod 列表的应用程序，同时避免使用 Cluster IP 的场景。

## 1.2、StatefulSet 控制器应用场景

StatefulSet 控制器适用于运行有状态应用程序的场景，包括：

- 1、数据库：例如 MySQL、PostgreSQL 等数据库系统，这些系统需要一个唯一的网络标识符和持久卷来保存数据。
- 2、缓存：例如 Redis 等缓存系统，这些系统需要一个可靠的网络标识符来支持客户端连接。
- 3、消息队列：例如 RabbitMQ、Kafka 等消息系统，这些系统需要一个固定的网络标识符来保证数据传输的可靠性。
- 4、其他有状态应用程序：例如一些需要固定网络标识符、有持久化状态或需要按序启动和关闭的应用程序。

总之，StatefulSet 控制器适用于有状态应用程序的场景，可以帮助开发者更好地管理和部署这些应用程序。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

## 2、StatefulSet 定义详解

StatefulSet 资源可以通过如下命令查看相关语法：

```
[root@k8s-master01 ~]# kubectl explain StatefulSet
```

### ● StatefulSet 资源说明

属性名称	取值类型	取值说明
apiVersion	string	Api 版本
kind	string	资源类型
metadata	Object	元数据
metadata.name	String	控制器的名称
metadata.namespace	String	控制器所属的命名空间，默认值为 default
metadata.labels[]	List	自定义标签列表
metadata.annotation[]	List	自定义注解列表
spec	Object	规范 DaemonSet 所需行为的规范
spec.template	Object	Pod 模板
spec.minReadySeconds	integer	当新的 pod 启动几秒种后，再 kill 掉旧的 pod
spec.ordinals	Object	如果设置了 ordinals，则 StatefulSet 控制器将按照这个列表中的序号依次创建 Pod。例如，如果设置了 ordinals 为[2, 0, 1]，则创建的 Pod 名称将从 my-pod-2、my-pod-0 到 my-pod-1。 在创建 Pod 时，StatefulSet 控制器会使用这些规则来为每个 Pod 分配一个唯一的网络标识符和持久化存储卷。这使得 Pod 可以保持其标识符和状态，即使在重启、重新调度或扩展 Pod 数量时也是如此。
spec.persistentVolumeClaimRetentionPolicy	Object	persistentVolumeClaimRetentionPolicy 是一个用于定义 StatefulSet 控制器如何管理与状态性 Pod 相关联的持久卷声明 (PVC) 的字段。 这个字段有两个选项： ● Retain: StatefulSet 控制器将不会删除相关的 PVC，手动删除 PVC 或改变 PVC 的拥有者才会导致 PVC 的删除。当需要重新启动一个 Pod 时，控制器会重复使用相同的 PVC。 ● Delete: 如果一个 Pod 消失了，控制器将删除相关的 PVC。
spec.podManagementPolicy	string	用于定义 StatefulSet 控制器如何管理 Pod 的创建、删除和更新的字段。 它有两个选项： ● OrderedReady: StatefulSet 控制器将确保每个 Pod 的启动顺序和就绪状态。首先它会启动 Pod-0，确保其就绪状态后，才会继续启动 Pod-1。在这种情况下，删除 Pod 也需要按照相反的顺序（默认配置） ● Parallel: StatefulSet 控制器将并行启动所有 Pod。在这种情况下，删除 Pod 时，没有顺序要求。

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

spec.replicas	integer	指定副本数量
spec.revisionHistoryLimit	integer	定义 StatefulSet 控制器在回滚时要保留的历史修订版本的数量。它设置历史修订版本的最大数量，如果达到这个限制，在创建新版本之前，最早的版本将被删除。 默认情况下，revisionHistoryLimit 设置为 10。
spec.serviceName	string	指定 headless service 的名称。必须指定所需的 Service 名称，控制器才可以正确地为 Pod 创建服务。
spec.updateStrategy	Object	用于定义 StatefulSet 控制器如何进行 Rolling Update 的字段。  它有两个选项：  ● RollingUpdate: StatefulSet 将按顺序逐个更新 Pod。在更新时，每个 Pod 根据定义的时间间隔等待一段时间，以便在 Pod 更新之前等待几秒钟。这可以避免发生数据丢失或对有状态应用造成损害。  RollingUpdate 有两个关键参数：partition 和 updatePeriodSeconds。partition 定义更新期间最大不可用 Pod 数量；updatePeriodSeconds 定义每个 pod 在更新前必须等待的时间  ● OnDelete: 只有在手动删除 Pod 时才会更新，此时新的 Pod 会自动创建。  示例： <pre>updateStrategy:   type: RollingUpdate   rollingUpdate:     partition: 1     updatePeriodSeconds: 60</pre> partition 最大可不用 Pod 数量为 1, 使更新操作按顺序逐个进行。updatePeriodSeconds 在每个 Pod 更新之前等待 60 秒的时间，这可以减少应用中断并保证数据一致性。
spec.updateStrategy.type	string	
spec.updateStrategy.rollingUpdate	Object	
spec.updateStrategy.rollingUpdate.maxUnavailable	string	
spec.updateStrategy.rollingUpdate.partition	integer	
spec.volumeClaimTemplates	[]Object	定义了 StatefulSet Pod 存储块的模板，是一个 PVC 模板的集合。 示例： <pre>volumeClaimTemplates: - metadata:   name: www   spec:     accessModes: [ "ReadWriteOnce" ]     resources:       requests:         storage: 1Gi</pre> 定义了 volumeClaimTemplates 字段，其中包含对 www PVC 模板的定义。当 Pod 启动时，该 PVC 模板创建了一个存储卷并将其挂载

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

		到容器中。在这种情况下，我们使用了 accessModes 允许一个 Pod 以读写模式挂载它，并为每个 Pod 配置 1Gi 存储资源。
--	--	---

### 3、StatefulSet 案例：部署 web 站点

小目标：使用 StatefulSet 控制器部署一个 web 站点

#### (1) 安装 NFS Server 并创建数据存放目录

1、安装 nfs

```
[root@k8s-master01 ~]# yum install nfs-utils -y
```

2、创建 NFS 需要的共享目录

```
[root@k8s-master01 ~]# mkdir /nfs/data -p
```

3、配置 nfs 共享服务器，共享/data/volumes 目录

```
[root@k8s-master01 ~]# vi /etc/exports
/nfs/data 192.168.128.0/24(rw,no_root_squash)
# no_root_squash: 用户具有根目录的完全管理访问权限
```

4、启动 nfs 服务

```
[root@k8s-master01 ~]# systemctl restart nfs && systemctl enable nfs &&
systemctl status nfs
```

#### (2) 安装 nfs-provisioner

```
[root@k8s-master01 ~]# vi nfs.yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: default
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: nfs-client-provisioner-runner
rules:
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
resources: ["persistentvolumes"]
verbs: ["get", "list", "watch", "create", "delete"]
- apiGroups: [""]
  resources: ["persistentvolumeclaims"]
  verbs: ["get", "list", "watch", "update"]
- apiGroups: ["storage.k8s.io"]
  resources: ["storageclasses"]
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources: ["events"]
  verbs: ["create", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: run-nfs-client-provisioner
subjects:
- kind: ServiceAccount
  name: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: default
roleRef:
  kind: ClusterRole
  name: nfs-client-provisioner-runner
  apiGroup: rbac.authorization.k8s.io
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: default
rules:
- apiGroups: [""]
  resources: ["endpoints"]
  verbs: ["get", "list", "watch", "create", "update", "patch"]
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: leader-locking-nfs-client-provisioner
  # replace with namespace where provisioner is deployed
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
namespace: default
subjects:
- kind: ServiceAccount
  name: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: default
roleRef:
  kind: Role
  name: leader-locking-nfs-client-provisioner
  apiGroup: rbac.authorization.k8s.io
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nfs-client-provisioner
  labels:
    app: nfs-client-provisioner
  # replace with namespace where provisioner is deployed
  namespace: default
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: nfs-client-provisioner
  template:
    metadata:
      labels:
        app: nfs-client-provisioner
    spec:
      serviceAccountName: nfs-client-provisioner
      containers:
        - name: nfs-client-provisioner
          image:
registry.cn-hangzhou.aliyuncs.com/lfy_k8s_images/nfs-subdir-external-provisioner:v4.0.2
          # resources:
          #   limits:
          #     cpu: 10m
          #   requests:
          #     cpu: 10m
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```

    volumeMounts:
      - name: nfs-client-root
        mountPath: /persistentvolumes
    env:
      - name: PROVISIONER_NAME #供应商名称，创建 SC 的时候需要指定
        value: kubernetes.test/nfs
      - name: NFS_SERVER #指定 nfs 服务器地址
        value: 192.168.128.11
      - name: NFS_PATH #指定 nfs 服务器共享的目录
        value: /nfs/data
  volumes:
    - name: nfs-client-root
      nfs: #使用 nfs 方式进行挂载，挂载到容器内
        server: 192.168.128.11
        path: /nfs/data
---
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: nfs-storage
provisioner: kubernetes.test/nfs
parameters:
  archiveOnDelete: "true"

[root@k8s-master01 ~]# kubectl apply -f nfs.yaml

[root@k8s-master01 ~]# kubectl get sc nfs-storage

[root@k8s-master01 ~]# kubectl get sc nfs-storage
NAME          PROVISIONER      RECLAIMPOLICY   VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
nfs-storage   kubernetes.test/nfs  Delete          Immediate           false                 3m43s
[root@k8s-master01 ~]#
```

### (3) 编写并更新 Statefulset 资源清单文件

```

[root@k8s-master01 ~]# vi statefulset.yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
    - port: 80
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
name: web
clusterIP: None
selector:
  app: nginx
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
    - metadata:
        name: www
      spec:
        accessModes: ["ReadWriteOnce"]
        storageClassName: "nfs-storage"
        resources:
          requests:
            storage: 5Gi

[root@k8s-master01 ~]# kubectl apply -f statefulset.yaml
service/nginx created
statefulset.apps/web created
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

对上面的 yaml 文件说明：

这是一个 Kubernetes YAML 配置文件，它定义了一个 StatefulSet 和一个 Service。StatefulSet 名为 web，包含了 2 个 nginx 的副本。Service 名为 nginx，通过端口 80 的 web 端口向外暴露服务。

该配置文件使用了一个名为 nfs-storage 的存储类来创建一个 PVC，并将其挂载在 StatefulSet 中每个 nginx 容器的 /usr/share/nginx/html 目录下。这意味着，两个容器将共享一个持久化的卷，其中存储了 nginx 需要的静态资源。

值得注意的是，这个 Service 的 clusterIP 属性设置为 None，这意味着该 Service 是一个 headless Service，它只作为一个 DNS 名称解析服务，不会分配 ClusterIP。

#### (4) 检查创建的资源

##### 1、查看 statefulset 是否创建成功

```
[root@k8s-master01 ~]# kubectl get statefulset
```

NAME	READY	AGE
web	2/2	46s

##### 2、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods -l app=nginx
```

NAME	READY	STATUS	RESTARTS	AGE
web-0	1/1	Running	0	2m13s
web-1	1/1	Running	0	2m4s

# 通过上面可以看到创建的 pod 是有序的

##### 3、查看 headless service

```
[root@k8s-master01 ~]# kubectl get svc -l app=nginx
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx	ClusterIP	None	<none>	80/TCP	3m28s

##### 4、查看 pvc

```
[root@k8s-master01 ~]# kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
www-web-0	Bound	pvc-0dd3bd08-3283-4cb1-bf97-8e1c9c097e27	5Gi	RWO	nfs-storage	3m45s
www-web-1	Bound	pvc-7aa150bd-bb5b-455d-932f-349ea5b8b7ad	5Gi	RWO	nfs-storage	3m36s

```
[root@k8s-master01 ~]#
```

##### 5、查看 pv

```
[root@k8s-master01 ~]# kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pvc-0dd3bd08-3283-4cb1-bf97-8e1c9c097e27	5Gi	RWO	Delete	Bound	default/www-web-0	nfs-storage		4m16s
pvc-7aa150bd-bb5b-455d-932f-349ea5b8b7ad	5Gi	RWO	Delete	Bound	default/www-web-1	nfs-storage		4m6s

```
[root@k8s-master01 ~]#
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

#### (5) 查看 pod 主机名

```
[root@k8s-master01 ~]# for i in 0 1; do kubectl exec web-$i -- sh -c 'hostname';done
web-0
web-1
```

(6) 进入刚才创建的 pod 容器中，安装 dnsutils 软件包，使用 nslookup 检查 statefulset 资源集群内部的 DNS 地址

```
[root@k8s-master01 ~]# kubectl exec -it web-1 -- /bin/bash
root@web-1:/# apt-get update
root@web-1:/# apt-get install dnsutils -y
root@web-1:/# nslookup web-0.nginx.default.svc.cluster.local
Server:      10.10.0.10
Address:     10.10.0.10#53

Name:   web-0.nginx.default.svc.cluster.local
Address: 10.244.85.197

root@web-1:/# nslookup web-1.nginx.default.svc.cluster.local
Server:      10.10.0.10
Address:     10.10.0.10#53

Name:   web-1.nginx.default.svc.cluster.local
Address: 10.244.58.200

root@web-1:/# nslookup nginx.default.svc.cluster.local
Server:      10.10.0.10
Address:     10.10.0.10#53

Name:   nginx.default.svc.cluster.local
Address: 10.244.85.197
Name:   nginx.default.svc.cluster.local
Address: 10.244.58.200

# 从上面的解析记录可以看出，statefulset 创建的 pod 也是有 dns 记录的。通过解析 headless service 地址，能够解析出对应 Pod 地址。
```

#### (7) 扩展：

举例说明 service 和 headless service 区别：

1、通过 deployment 创建 pod，pod 前端创建一个 service

```
[root@k8s-master01 ~]# vi deploy-service.yaml
apiVersion: v1
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  type: ClusterIP
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    run: my-nginx
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx:latest
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 80
```

## 2、更新资源清单文件

```
[root@k8s-master01 ~]# kubectl apply -f deploy-service.yaml
service/my-nginx created
deployment.apps/my-nginx created
```

## 3、查看 service

```
[root@k8s-master01 ~]# kubectl get svc -l run=my-nginx
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
my-nginx	ClusterIP	10.10.56.210	<none>	80/TCP	6s

4、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods -l run=my-nginx
```

NAME	READY	STATUS	RESTARTS	AGE
my-nginx-55598fb54c-2b79v	1/1	Running	0	24s
my-nginx-55598fb54c-vzfb	1/1	Running	0	24s

5、检查 fqdn

```
[root@k8s-master01 ~]# kubectl exec -it my-nginx-55598fb54c-2b79v -- /bin/bash
root@my-nginx-55598fb54c-2b79v:/# apt-get update
root@my-nginx-55598fb54c-2b79v:/# apt-get install dnsutils -y
root@my-nginx-55598fb54c-2b79v:/# nslookup
my-nginx.default.svc.cluster.local

Server:      10.10.0.10
Address:     10.10.0.10#53

Name:   my-nginx.default.svc.cluster.local
Address: 10.10.56.210

root@my-nginx-55598fb54c-2b79v:/# nslookup
my-nginx-55598fb54c-2b79v.my-nginx.default.svc.cluster.local

root@my-nginx-55598fb54c-2b79v:/# nslookup my-nginx-55598fb54c-2b79v.my-nginx.default.svc.cluster.local
Server:      10.10.0.10
Address:     10.10.0.10#53

** server can't find my-nginx-55598fb54c-2b79v.my-nginx.default.svc.cluster.local: NXDOMAIN
root@my-nginx-55598fb54c-2b79v:/#
```

headless service 可以为 Pod 提供 FQDN 解析，而 service 不能。

## 4、StatefulSet 管理 pod

### 4.1、StatefulSet 实现 pod 的动态扩容

下面介绍两种方法来实现 pod 的动态扩容：

方法一：编辑 yaml 文件实现扩容（推荐）

1、修改资源清单文件
<pre>[root@k8s-master01 ~]# vi statefulset.yaml replicas: 3</pre>
2、更新资源清单文件
<pre>[root@k8s-master01 ~]# kubectl apply -f statefulset.yaml</pre>

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

```
service/nginx unchanged
statefulset.apps/web configured
```

3、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods -l app=nginx
```

NAME	READY	STATUS	RESTARTS	AGE
web-0	1/1	Running	0	25m
web-1	1/1	Running	0	25m
web-2	1/1	Running	0	17s

4、查看 pvc

```
[root@k8s-master01 ~]# kubectl get pvc
```

```
[root@k8s-master01 ~]# kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
www-web-0	Bound	pvc-0dd3bd08-3283-4cb1-bf97-8e1c9c097e27	5Gi	RW0	nfs-storage	43m
www-web-1	Bound	pvc-7aa150bd-bb5b-455d-932f-349ea5b8b7ad	5Gi	RW0	nfs-storage	43m
www-web-2	Bound	pvc-a8bf97ec-47d4-4945-a0ac-3cbb3af8b72d	5Gi	RW0	nfs-storage	31s

```
[root@k8s-master01 ~]#
```

## 方法二：编辑控制器实现扩容（不推荐）

1、查看 statefulset 状态

```
[root@k8s-master01 ~]# kubectl get statefulset
```

NAME	READY	AGE
web	3/3	27m

2、编辑 statefulset 控制器

```
[root@k8s-master01 ~]# kubectl edit statefulset web
```

```
replicas: 4
```

```
spec:
  podManagementPolicy: OrderedReady
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  serviceName: nginx
```

3、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods -l app=nginx
```

NAME	READY	STATUS	RESTARTS	AGE
web-0	1/1	Running	0	28m
web-1	1/1	Running	0	28m
web-2	1/1	Running	0	3m44s
web-3	1/1	Running	0	6s

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

#### 4、查看 svc

```
[root@k8s-master01 ~]# kubectl get pvc
```

```
[root@k8s-master01 ~]# kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
www-web-0	Bound	pvc-0dd3bd08-3283-4cb1-bf97-8e1c9c097e27	5Gi	RW0	nfs-storage	46m
www-web-1	Bound	pvc-7aa150bd-bb5b-455d-932f-349ea5b8b7ad	5Gi	RW0	nfs-storage	46m
www-web-2	Bound	pvc-a8bf97ec-47d4-4945-a0ac-3cbb3af8b72d	5Gi	RW0	nfs-storage	3m56s
www-web-3	Bound	pvc-6609e494-e234-4a64-8c41-071cc3b1f818	5Gi	RW0	nfs-storage	18s

```
[root@k8s-master01 ~]#
```

### 4.2、StatefulSet 实现 pod 的动态缩容

下面介绍两种方法来实现 pod 的动态缩容：

方法一：编辑 yaml 文件实现缩容（推荐）

#### 1、修改资源清单文件

```
[root@k8s-master01 ~]# vi statefulset.yaml
replicas: 3
```

#### 2、更新资源清单文件

```
[root@k8s-master01 ~]# kubectl apply -f statefulset.yaml
service/nginx unchanged
statefulset.apps/web configured
```

#### 3、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods -l app=nginx
```

NAME	READY	STATUS	RESTARTS	AGE
web-0	1/1	Running	0	37m
web-1	1/1	Running	0	36m
web-2	1/1	Running	0	12m

#### 4、查看 pvc

```
[root@k8s-master01 ~]# kubectl get pvc
```

```
[root@k8s-master01 ~]# kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
www-web-0	Bound	pvc-0dd3bd08-3283-4cb1-bf97-8e1c9c097e27	5Gi	RW0	nfs-storage	56m
www-web-1	Bound	pvc-7aa150bd-bb5b-455d-932f-349ea5b8b7ad	5Gi	RW0	nfs-storage	55m
www-web-2	Bound	pvc-a8bf97ec-47d4-4945-a0ac-3cbb3af8b72d	5Gi	RW0	nfs-storage	13m
www-web-3	Bound	pvc-6609e494-e234-4a64-8c41-071cc3b1f818	5Gi	RW0	nfs-storage	9m39s

```
[root@k8s-master01 ~]#
```

在做缩容的时候，不会删除 PVC。

方法二：编辑控制器实现缩容（不推荐）

#### 1、查看 statefulset 状态

```
[root@k8s-master01 ~]# kubectl get statefulset
```

NAME	READY	AGE
web	3/3	39m

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

## 2、编辑 statefulset 控制器

```
[root@k8s-master01 ~]# kubectl edit statefulset web
replicas: 2
```

```
spec:
  podManagementPolicy: OrderedReady
  replicas: 2
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app: nginx
  serviceName: nginx
  template:
    metadata:
      creationTimestamp: null
    labels:
```

## 3、查看 pod

```
[root@k8s-master01 ~]# kubectl get pods -l app=nginx
```

NAME	READY	STATUS	RESTARTS	AGE
web-0	1/1	Running	0	42m
web-1	1/1	Running	0	42m

## 4、查看 svc

```
[root@k8s-master01 ~]# kubectl get pvc
```

```
[root@k8s-master01 ~]# kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
www-web-0	Bound	pvc-0dd3bd08-3283-4cb1-bf97-8e1c9c097e27	5Gi	RWO	nfs-storage	60m
www-web-1	Bound	pvc-7aa150bd-bb5b-455d-932f-349ea5b8b7ad	5Gi	RWO	nfs-storage	60m
www-web-2	Bound	pvc-a8bf97ec-47d4-4945-a0ac-3cbb3af8b72d	5Gi	RWO	nfs-storage	17m
www-web-3	Bound	pvc-6609e494-e234-4a64-8c41-071cc3b1f818	5Gi	RWO	nfs-storage	14m

```
[root@k8s-master01 ~]#
```

## 4.3、StatefulSet 实现 pod 的更新

### (1) 查看当前版本 statefulset 使用的镜像

```
[root@k8s-master01 ~]# kubectl describe statefulset web | grep Image:
Image:          nginx
```

### (2) 再开一个终端，动态查看 pod 更新策略（这里是默认的更新策略）

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

### (3) 更新资源清单文件

```
[root@k8s-master01 ~]# vi statefulset.yaml
image: tomcat:latest
[root@k8s-master01 ~]# kubectl apply -f statefulset.yaml
service/nginx unchanged
statefulset.apps/web configured
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

#### (4) 查看更新过程

[root@k8s-master01 ~]# kubectl get pods -l app=nginx -w					
[root@k8s-master01 ~]# kubectl get pods -l app=nginx -w					
NAME	READY	STATUS	RESTARTS	AGE	
web-0	1/1	Running	0	57s	旧Pod
web-1	1/1	Running	0	41s	
web-1	1/1	Terminating	0	56s	删除一个旧Pod
web-1	1/1	Terminating	0	56s	
web-1	0/1	Terminating	0	56s	
web-1	0/1	Terminating	0	56s	
web-1	0/1	Pending	0	0s	创建一个新Pod
web-1	0/1	Pending	0	0s	
web-1	0/1	ContainerCreating	0	0s	
web-1	0/1	ContainerCreating	0	1s	
web-1	1/1	Running	0	42s	再删除一个旧Pod
web-0	1/1	Terminating	0	114s	
web-0	1/1	Terminating	0	114s	
web-0	0/1	Terminating	0	114s	
web-0	0/1	Terminating	0	114s	
web-0	0/1	Pending	0	0s	再创建一个新Pod，完成更新
web-0	0/1	Pending	0	0s	
web-0	0/1	ContainerCreating	0	1s	
web-0	0/1	ContainerCreating	0	1s	
web-0	1/1	Running	0	2s	

#### (5) 查看当前版本 statefulset 使用的镜像

[root@k8s-master01 ~]# kubectl describe statefulset web   grep Image:	
Image:	tomcat:latest

提示:

这里对 StatefulSet 资源进行更新使用的是默认的更新策略,如果想要修改更新方式,可以参考上面的 StatefulSet 定义详解。

### 4.4、StatefulSet 实现 pod 的回滚

目标: 研发部突然发消息说刚才上的新版本有问题,让马上回滚。现在我们对 StatefulSet web 进行版本回退。

#### (1) 查看 statefulset web 对象的历史版本

[root@k8s-master01 ~]# kubectl rollout history statefulset web	
statefulset.apps/web	
REVISION	CHANGE-CAUSE
1	<none>
2	<none>

#### (2) 回滚 statefulset web 对象的历史版本为 1

[root@k8s-master01 ~]# kubectl rollout undo statefulset web --to-revision=1	
statefulset.apps/web rolled back	

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**

(3) 查看 statefulset web 对象的历史版本

```
[root@k8s-master01 ~]# kubectl rollout history statefulset web
statefulset.apps/web
REVISION  CHANGE-CAUSE
2          <none>    #<==这里是 v2 版本
3          <none>    #<==这里是 v1 版本，也是我们当前版本
```

(4) 查看当前版本使用的镜像，确认回退成功

```
[root@k8s-master01 ~]# kubectl describe statefulset web | grep Image
Image:          nginx
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，**禁止私自传阅，违者依法追责。**