

Kubernetes 存储卷管理实践

叶家炜

Oct 17, 2026

容器化技术的普及让应用部署变得轻量而敏捷，但存储问题却成了新的瓶颈。无状态应用可以随时启停，而有状态服务如数据库或消息队列却依赖持久化数据。Kubernetes 通过存储卷机制，将数据生命周期与 Pod 生命周期解耦，确保即使 Pod 被重建，数据依然存在。早期版本只提供简单的 EmptyDir 卷，数据随 Pod 消失而丢失；随着 CSI 接口的引入，社区开始支持 Ceph、NFS、云盘等多种后端存储，形成了丰富而成熟的生态。

1 核心概念：存储卷类型与生命周期

理解 Kubernetes 存储，首先要区分临时卷和持久卷。临时卷如 EmptyDir、ConfigMap、Secret，随 Pod 创建而生成，Pod 删除后即被清理，适合存放缓存或配置。持久卷则通过 PV、PVC、StorageClass 三层模型实现解耦。PV 是集群管理员预置的存储资源，描述了容量、访问模式和回收策略；PVC 是用户向集群申请存储的声明，指定所需容量和访问要求；StorageClass 则充当动态供应的模板，定义了存储后端的 provisioner、参数和回收策略。

在实际运行中，PV 与 PVC 的绑定遵循匹配规则：容量、访问模式、StorageClass 名称必须一致。绑定成功后，Pod 通过 PVC 名称挂载卷，数据写入后将持久化到后端存储。卷的生命周期可概括为四个阶段：Binding、Using、Releasing、Reclaiming。Reclaiming 阶段由 reclaimPolicy 决定，可选 Delete 或 Retain，前者自动删除 PV，后者保留 PV 供人工处理。

2 静态供应 vs 动态供应：两种供应模式对比

静态供应要求管理员提前创建 PV，再由用户创建 PVC 进行匹配。这种模式适合已有存储资源或合规性要求严格的场景，但维护成本高。动态供应则借助 StorageClass，由 Kubernetes 自动为 PVC 创建 PV。管理员只需在 StorageClass 中指定 provisioner，如 aws-ebs 或 ceph-rbd，Kubernetes 就会调用后端 API 创建对应存储卷。选型时需考虑资源是否已存在、是否需要自动化，以及对存储类型的要求。

3 实战：从 0 到 1 部署有状态应用

以 MySQL 单实例为例，展示 StatefulSet 与 PVC 的结合使用。首先确保集群版本不低于 v1.21，并已配置好 StorageClass。以下是核心 YAML 片段：

```
1 apiVersion: apps/v1
  kind: StatefulSet
3 metadata:
```

```
    name: mysql
5 spec:
    serviceName: mysql
7 replicas: 1
    selector:
9      matchLabels:
        app: mysql
11 template:
    metadata:
13      labels:
        app: mysql
15    spec:
        containers:
17      - name: mysql
          image: mysql:8.0
19          env:
            - name: MYSQL_ROOT_PASSWORD
21              valueFrom:
                secretKeyRef:
23                  name: mysql-secret
                  key: password
25          volumeMounts:
            - name: data
27              mountPath: /var/lib/mysql
    volumeClaimTemplates:
29      - metadata:
          name: data
31          spec:
            accessModes: ["ReadWriteOnce"]
33            storageClassName: standard
            resources:
35              requests:
                storage: 10Gi
```

这段配置中，StatefulSet 的 volumeClaimTemplates 字段定义了 PVC 模板，Kubernetes 会为每个 Pod 创建独立的 PVC，名称格式为 data-mysql-0。accessModes 设置为 ReadWriteOnce，表示卷只能被单个节点挂载。storageClassName 指向已存在的 StorageClass，触发动态供应。Pod 启动后，MySQL 数据目录 /var/lib/mysql 被映射到后端存储。即使 Pod 被删除重建，新的 Pod 仍会绑定同一 PVC，数据得以保留。

扩展到主从复制时，可将 replicas 设为 2，StatefulSet 会自动创建 mysql-0 和 mysql-1 两个 Pod，每个

Pod 拥有独立的 PVC。初始化脚本可通过 ConfigMap 挂载，敏感信息通过 Secret 注入，避免将密码写死在镜像中。

4 高级主题：快照、扩容与跨可用区

卷快照功能依赖 CSI Snapshotter 组件。安装后，可创建 VolumeSnapshot 对象：

```
1 apiVersion: snapshot.storage.k8s.io/v1
2 kind: VolumeSnapshot
3 metadata:
4   name: mysql-snapshot
5 spec:
6   volumeSnapshotClassName: csi-hostpath-snapclass
7   source:
8     persistentVolumeClaimName: data-mysql-0
```

该对象触发后端存储生成快照，可用于定时备份或克隆新卷。扩容场景下，需在 StorageClass 中设置 `allowVolumeExpansion: true`，然后修改 PVC 的 `resources.requests.storage` 字段。文件系统层需执行 `resize2fs` 或 `xfs_growfs` 完成在线扩容，无需停机。

跨可用区部署时，StorageClass 的 `volumeBindingMode` 应设为 `WaitForFirstConsumer`，延迟 PV 绑定直到 Pod 调度完成。此时结合 Pod 的 `nodeAffinity` 或 `topologySpreadConstraints`，可确保数据与计算亲和性一致，提升高可用性。

5 性能与可靠性调优

I/O 性能受存储类型影响最大。块存储适合数据库，延迟低且支持随机读写；文件存储便于共享；对象存储则适合备份。生产环境常采用 `local-path-provisioner` 配合 SSD 节点构建缓存层，降低延迟。高可用方案可选用 Ceph 或 Longhorn，实现多副本数据冗余。故障转移演练时，可模拟节点宕机，观察 `StatefulSet` 是否自动在健康节点重建 Pod 并重新挂载 PVC。

监控方面，利用 Prometheus 采集 `kube-state-metrics` 中的 `kube_persistentvolumeclaim_status_phase` 和 `kube_persistentvolumeclaim_resource_requests_storage_bytes` 指标，设置告警规则：当 PVC 使用率超过 85% 时触发通知，提前扩容避免业务中断。

6 常见问题排查手册

PVC 长期处于 Pending 状态，通常因 StorageClass 不存在、节点亲和性冲突或资源配额不足。排查时先执行 `kubectl describe pvc` 查看 Events 字段，确认错误原因。数据丢失风险主要来自 `reclaimPolicy=Delete` 的误操作，建议生产环境始终设为 `Retain`。误删 PV 后，若底层存储未被真正释放，可通过恢复 PV 对象并重新绑定 PVC 挽救数据。性能瓶颈定位可结合 `kubectl top pod` 与节点上的 `iostat -x 1` 命令，区分是存储后端延迟还是网络带宽限制。

生产环境应遵循三条原则：设置 `reclaimPolicy=Retain`、敏感数据使用加密卷、定期演练灾难恢复。未来趋势

包括 WASM 存储插件、CSI v2 规范，以及 Serverless 存储如 AWS EFS CSI，将进一步降低运维复杂度。延伸阅读可参考官方文档与社区最佳实践案例，持续关注存储生态演进。