存储

介绍

介绍

概念

访问模式与卷模式

Kubernetes 中的访问模式

Kubernetes 中的卷模式

存储功能：快照与扩容

总结

核心概念

Persistent Volume (PV)

Persistent Volume Claim (PVC)

通用临时卷（Generic Ephemeral Volumes）

emptyDir

hostPath

ConfigMap

Secret

StorageClass

Container Storage Interface (CSI)

Persistent V

动态 Persistent

Persistent Volu

功能指南

创建 CephFS 文件存储类型存储类

部署卷插件

创建存储类

创建 CephRBD 块存储类型存储类

部署卷插件

创建存储类

创建 TopoLV

背景信息

部署卷插件

创建存储类

后续操作

创建 NFS 共享存储类

前提条件

部署 NFS 共享存储插件

创建 NFS 共享存储类

相关操作

部署卷快照组件

操作步骤

创建 PV

前提条件

创建 PVCs

先决条件

使用卷快照

先决条件

示例 VolumeSnapshot 自定义资源（CR）

通过 Web 控制台创建卷快照

通过 CLI 创建卷快照

从卷快照创建持久卷声明

额外资源

实用指南

设置 NFS 共享存储类的子目录命...

功能概述

通用临时卷

示例临时卷

使用 emptyI

示例 emptyDir

使用案例
前提条件
操作步骤

主要特性
何时使用通用临时卷
它们与 emptyDir 的区别

可选的 Medium
主要特性
常见用例

如何注释第三方存储能力

第一步：打开存储类配置
第二步：填写存储类信息
第三步：使用 ConfigMap 注释存储能力
第四步：了解支持的存储能力字段
第五步：完成存储类配置
可选：使用注释的存储类创建 PVC

故障排除

从 **PVC** 扩容失败中恢复

操作步骤
其他提示

介绍

Kubernetes 提供灵活且可扩展的存储机制，用于在容器化环境中管理数据持久性。通过抽象存储资源（如 Volumes、PersistentVolumes 和 PersistentVolumeClaims），Kubernetes 将应用与底层存储解耦，支持动态供给、自动挂载以及跨节点的数据持久化。

其核心功能包括支持多种后端存储系统（例如，本地磁盘、NFS、云存储服务）、动态供给、访问模式控制（如读写权限）和生命周期管理，满足有状态应用的存储需求。对于需要高可用性、数据持久性和多租户隔离的企业级工作负载，Kubernetes 存储是重要的基础能力。

Kubernetes 的存储功能旨在帮助开发者、运维工程师和平台团队高效且安全地管理容器化工作负载中的数据。

概念

访问模式与卷模式

Kubernetes 中的访问模式

Kubernetes 中的卷模式

存储功能：快照与扩容

总结

核心概念

Persistent Volume (PV)

Persistent Volume Claim (PVC)

通用临时卷（Generic Ephemeral Volumes，

emptyDir

hostPath

ConfigMap

Secret

StorageClass

Container Storage Interface (CSI)

Persistent V

动态 Persistent

Persistent Volu

访问模式与卷模式

在 Kubernetes 中，PersistentVolumeClaims（PVC）与 StorageClass 协同工作，用于管理工作负载如何配置和访问存储。在这一领域中，两个关键概念是 访问模式（**Access Modes**） 和卷模式（**Volume Modes**）。本文将探讨这些概念，并说明不同存储系统对它们的支持情况。

目录

Kubernetes 中的访问模式

访问模式定义了一个卷可以被 Pods 挂载和使用的方式。主要的访问模式包括：

- **ReadWriteOnce (RWO)**：卷可以被单个节点以读写方式挂载。
- **ReadOnlyMany (ROX)**：卷可以被多个节点以只读方式挂载。
- **ReadWriteMany (RWX)**：卷可以被多个节点以读写方式挂载。

各存储类对访问模式的支持情况

存储类	支持 **RWO**	支持 **ROX**	支持 **RWX**
CephFS 文件存储	是	否	是
CephRBD 块存储	是	否	否
TopoLVM	是	否	否
NFS 共享存储	是	否	是

如上所示，**CephFS** 和 **NFS** 等基于文件的存储系统支持多节点的共享读写操作，适用于多副本或并发访问的场景。而像 **CephRBD** 和 **TopoLVM** 这样的块存储系统则更适合单节点独占访问。

Kubernetes 中的卷模式

卷模式定义了数据如何暴露给 Pods：

- **Filesystem**（文件系统）：将卷作为文件系统挂载进 Pods。
- **Block**（块设备）：将卷作为原始块设备呈现。

各存储类对卷模式的支持情况

存储类	类型	支持的卷模式
CephFS 文件存储	文件存储	文件系统
CephRBD 块存储	块存储	文件系统、块设备
TopoLVM	块存储	文件系统、块设备
NFS 共享存储	文件存储	文件系统

如 **CephRBD** 和 **TopoLVM** 等块存储系统同时支持文件系统模式与原始块模式，为不同应用需求提供了更大的灵活性。而 **CephFS** 与 **NFS** 等文件存储系统则仅支持文件系统模式。

存储功能：快照与扩容

Kubernetes 还支持一些高级功能，如卷快照与 PVC 动态扩容，具体支持情况如下：

存储类	支持卷快照	支持扩容
CephFS 文件存储	支持	支持
CephRBD 块存储	支持	支持
TopoLVM	支持	支持
NFS 共享存储	不支持	不支持

仅当使用 StorageClass 动态创建 PVC 时，平台才支持卷快照功能。这一功能适合用于数据备份和环境克隆等场景。

总结

在 Kubernetes 中配置存储时，了解 PVC 及其背后 StorageClass 的 访问模式 与 卷模式 是选择合适解决方案的关键。像 CephFS 与 NFS 这样的文件存储适用于共享访问场景，而 CephRBD 与 TopoLVM 等块存储则更适合高性能、单节点部署。此外，对快照与扩容等功能的支持也能显著提升存储的灵活性和数据管理能力。

核心概念

Kubernetes 存储围绕三个关键概念展开：**PersistentVolume (PV)**、**PersistentVolumeClaim (PVC)** 和 **StorageClass**。它们定义了集群内存储的请求、分配和配置方式。在底层，**CSI**（Container Storage Interface）驱动程序通常负责实际的存储供应和挂载。下面我们简要介绍每个组件，并重点说明 CSI 驱动的作用。

目录

Persistent Volume (PV)

PersistentVolume (PV) 是集群中已被预配的存储（可以由管理员静态配置，也可以通过 StorageClass 动态配置）。它代表底层存储，例如云服务商的磁盘或网络附加文件系统，并作为集群中的一种资源存在，类似于节点。

Persistent Volume Claim (PVC)

PersistentVolumeClaim (PVC) 是对存储的请求。用户定义所需的存储容量和访问模式（例如读写）。如果有合适的 PV 可用，或者可以通过 StorageClass 动态供应，PVC 就会与该 PV “绑定”。绑定后，Pod 可以引用 PVC 来持久化或共享数据。

通用临时卷（Generic Ephemeral Volumes）

Kubernetes 引入的通用临时卷功能允许您在 Pod 生命周期内使用由 CSI 驱动的 `temporary` 卷，这类似于 `emptyDir`，但功能更强大，支持挂载任何类型的 CSI 卷（支持快照、扩展等功能）。

更多用法请参考 Generic ephemeral volumes

emptyDir

1. emptyDir 是一种临时存储卷，类型为空目录。
2. 它在 Pod 被调度到节点时创建，存储位于该节点的本地文件系统（默认是节点磁盘）。
3. 当 Pod 被删除时，emptyDir 中的数据也会被清除。

更多用法请参考 Using an emptyDir

hostPath

在 Kubernetes 中，hostPath 卷是一种特殊类型的卷，它将宿主节点文件系统中的文件或目录直接映射到 Pod 的容器中。

- 它允许 Pod 访问宿主节点上的文件或目录。
- 适用于：
  - 访问宿主级资源（例如 Docker 套接字）

- 调试
- 使用节点上已有的数据

ConfigMap

Kubernetes 中的 ConfigMap 是一种 API 对象，用于以键值对形式存储非敏感的配置信息。它允许您将配置与应用代码解耦，使应用更具可移植性且易于管理。

Secret

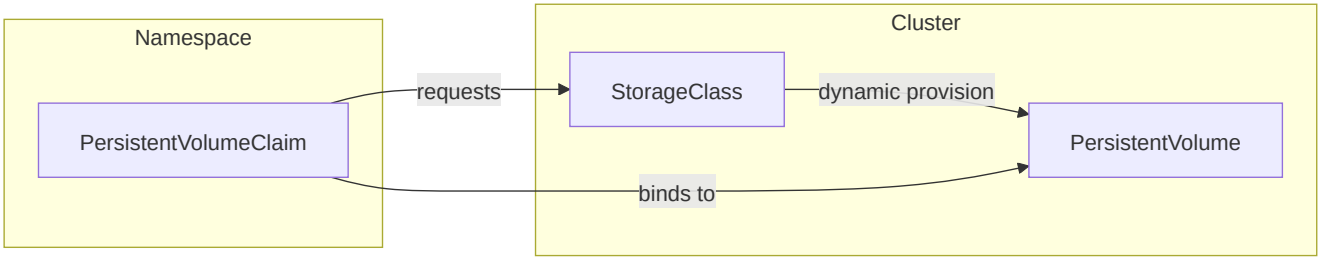
在 Kubernetes 中，Secret 是一种 API 对象，用于存储敏感数据，例如：

- 密码
- OAuth 令牌
- SSH 密钥
- TLS 证书
- 数据库凭据

Secret 通过避免将敏感数据直接存储在 Pod 规格或容器镜像中来保护这些数据。

StorageClass

StorageClass 描述了卷应如何动态供应。它映射到特定的供应器（通常是 CSI 驱动），并可以包含存储层级、性能特征或其他后端配置参数。通过创建多个 StorageClass，您可以为开发者提供多种类型的存储选择。



图示：PVC、PV 和 StorageClass 之间的关系。

Container Storage Interface (CSI)

Container Storage Interface (CSI) 是 Kubernetes 用于集成存储驱动的标准 API。它允许第三方存储提供商构建外部插件，这意味着您可以安装或更新存储驱动而无需修改 Kubernetes 本身。

一个 CSI 驱动 通常包含两个组件：

1. 控制器组件：运行在集群中（通常以 Deployment 形式），负责高级操作，如 创建 或 删除卷。对于网络存储，它还可能负责将卷附加或分离到节点。
2. 节点组件：运行在每个节点上（通常以 DaemonSet 形式），负责在该节点上 挂载 和 卸载卷。它与 kubelet 通信，确保卷对 Pod 可用。

当用户创建引用使用 CSI 驱动的 StorageClass 的 PVC 时，CSI 驱动会监听该请求并相应地供应存储（如果需要动态供应）。存储创建完成后，驱动通知 Kubernetes，Kubernetes 创建对应的 PV 并将其绑定到 PVC。每当 Pod 使用该 PVC 时，驱动的节点组件负责挂载卷，使存储在容器内可用。

通过利用 **PV**、**PVC**、**StorageClass** 和 **CSI**，Kubernetes 实现了强大且声明式的存储管理方式。管理员可以定义一个或多个 StorageClass 来代表不同的存储后端或性能层级，而开发者只需通过 PVC 请求存储，无需关心底层基础设施。

Persistent Volume

PersistentVolume（PV）表示 Kubernetes 集群中与后端存储卷的映射关系，作为 Kubernetes API 资源存在。它是由管理员统一创建和配置的集群资源，负责抽象实际的存储资源，构成集群的存储基础设施。

PersistentVolume 拥有独立于 Pod 的生命周期，能够实现 Pod 数据的持久化存储。

管理员可以手动创建静态 PersistentVolume，或者基于存储类生成动态 PersistentVolume。当开发人员需要为应用获取存储资源时，可以通过 PersistentVolumeClaim（PVC）进行申请，PVC 会匹配并绑定合适的 PersistentVolume。

目录

动态 Persistent Volume 与静态 Persistent Volume

平台支持管理员管理两种类型的 PersistentVolume，分别是动态和静态 Persistent Volume。

- 动态 **Persistent Volume**：基于存储类实现。存储类由管理员创建，定义了一种描述存储资源类别的 Kubernetes 资源。当开发人员创建关联存储类的 PersistentVolumeClaim 后，平台会根据 PersistentVolumeClaim 和存储类中配置的参数动态创建合适的 PersistentVolume，并绑定到该 PersistentVolumeClaim，实现存储资源的动态分配。

- 静态 **Persistent Volume**：由管理员手动创建的 Persistent Volume。目前支持创建 **HostPath** 或 **NFS** 共享存储 类型的静态 Persistent Volume。当开发人员创建不使用存储类的 PersistentVolumeClaim 时，平台会根据 PersistentVolumeClaim 中配置的参数匹配并绑定合适的静态 PersistentVolume。

  - **HostPath**：使用节点主机上的文件目录（不支持本地存储）作为后端存储，例如：`/etc/kubernetes`。通常仅适用于单计算节点集群的测试场景。

  - **NFS** 共享存储：指网络文件系统，是 Persistent Volume 常见的后端存储类型。用户和程序可以像访问本地文件一样访问远程系统上的文件。

Persistent Volume 的生命周期

1. **Provisioning**（配置）：管理员手动创建静态 Persistent Volume。创建后，Persistent Volume 进入 **Available** 状态；或者平台根据关联存储类的 PersistentVolumeClaim 动态创建合适的 Persistent Volume。

2. **Binding**（绑定）：静态 Persistent Volume 被匹配并绑定到 PersistentVolumeClaim 后，进入 **Bound** 状态；动态 Persistent Volume 根据匹配的 PersistentVolumeClaim 动态创建成功后，也进入 **Bound** 状态。

3. **Using**（使用）：开发人员将 PersistentVolumeClaim 关联到计算组件的容器实例，使用 Persistent Volume 映射的后端存储资源。

4. **Releasing**（释放）：开发人员删除 PersistentVolumeClaim 后，Persistent Volume 被释放。

5. **Reclaiming**（回收）：Persistent Volume 释放后，根据 Persistent Volume 或存储类的回收策略参数对其进行回收操作。

功能指南

创建 CephFS 文件存储类型存储类

CephFS 文件存储是内置的 Ceph 文件存储系统，为平台提供基于 CSI（Container Storage Interface）的存储接入方式，提供安全、可靠和可扩展的共享文件存储服务，适用于文件共享和数据备份等场景。在继续之前，您必须先创建 CephFS 文件存储类。

在持久卷声明（PVC，PersistentVolumeClaim）中绑定存储类后，平台将根据持久卷声明在节点上动态创建持久卷以供业务应用使用。

目录

部署卷插件

单击 部署 后，在 分布式存储 页面中 创建存储服务 或 接入存储服务。

创建存储类

1. 进入 平台管理。

2. 在左侧导航栏中，单击 存储管理 > 存储类。

3. 单击 创建存储类。

说明：下述内容以表单方式为例，您也可选择使用 YAML 创建。

4. 选择 **CephFS** 文件存储，单击 下一步。

5. 参考以下说明配置相关参数。

参数	说明
回收策略	持久卷的回收策略。 - 删除：当持久卷声明被删除时，绑定的持久卷也会被删除。 - 保留：即使持久卷声明被删除，绑定的持久卷仍然保留。
访问模式	当前存储支持的所有访问模式。后续声明持久卷时，只能选择其中一种模式。 - 单节点读写 (RWO)：可以被一个节点以读写方式挂载。 - 多节点读写 (RWX)：可以被多个节点以读写方式挂载。
分配项目	请分配可使用此类型存储的项目。 如果目前没有项目需要使用此存储类型，您可以选择不分配，稍后再进行更新。

提示：以下参数需在分布式存储中设置，并将在此处直接应用。

- 存储集群：当前集群中的内置 Ceph 存储集群。
- 存储池：存储集群中用于数据存储的逻辑分区。

6. 单击 创建。

创建 CephRBD 块存储类型存储类

CephRBD 块存储为平台内置的 Ceph 块存储，可为平台提供基于 CSI（Container Storage Interface）的存储接入方式，可用于提供高 IOPS，低延迟的存储服务，适用于数据库、虚拟化等场景。在此之前，您需先创建 CephRBD 块存储类。

持久卷声明（PVC，PersistentVolumeClaim）中绑定存储类后，平台将根据持久卷声明在节点上动态创建持久卷供业务应用使用。

目录

部署卷插件

单击 部署 后，在 分布式存储 页面中 创建存储服务 或 接入存储服务。

创建存储类

1. 进入 平台管理。

2. 在左侧导航栏中，单击 存储管理 > 存储类。

3. 单击 创建存储类。

说明：下述内容以表单方式为例，您也可选择 YAML 创建完成操作。

4. 选择 **CephRBD** 块存储，单击 下一步。

5. 按要求配置参数。

参数	说明
文件系统	默认为 **EXT4**，即 Linux 下的一种日志文件系统，可提供 extent 文件存储方式，支持处理大型文件。文件系统容量可达 1 EiB，支持的文件大小可达 16 TiB。
回收策略	持久卷的回收策略。 - 删除：删除持久卷声明的同时，也会删除绑定的持久卷。 - 保留：即使删除持久卷声明，其绑定的持久卷仍会被保留。
访问模式	仅支持单节点读写 (RWO)：可以被一个节点以读写方式挂载。
分配项目	请分配可使用此类型存储的项目。 如果暂时没有项目需要使用此类型存储，您也可先不分配项目，后续再更新项目。

提示：以下参数需在分布式存储中设置，此处将直接应用。

- 存储集群：当前集群中的内置 Ceph 存储集群。
- 存储池：存储集群中用于存储数据的逻辑分区。

6. 单击 创建。

创建 TopoLVM 本地存储类

TopoLVM 是基于 LVM 的本地存储解决方案，提供简单、易于维护且高性能的本地存储服务，适用于数据库和中间件等场景。在使用之前，需要先创建一个 TopoLVM 存储类。

一旦持久卷声明（PVC）与存储类绑定，平台会根据持久卷声明在节点上动态创建持久卷供业务应用使用。

目录

背景信息

使用优势

- 相比远程存储（例如 **NFS** 共享存储）：TopoLVM 类型的存储位于节点本地，提供更好的 IOPS 和吞吐性能，具有更低的延迟。

- 相比 hostPath（例如 **local-path**）：虽然两者都是节点上的本地存储，但 TopoLVM 允许将容器组灵活调度到资源充足的节点上，避免因资源不足而导致容器组无法启动。
- TopoLVM 默认支持自动扩容。修改持久卷声明中的存储配额后，无需重启容器组，扩容将自动完成。

使用场景

- 仅当需要临时存储时，例如开发和调试。
- 当存在高存储 I/O 要求时，例如实时索引。

约束与限制

请尽量仅在能够实现应用层数据复制和备份的应用中使用本地存储，例如 MySQL。避免因本地存储缺乏数据持久性保证而导致的数据丢失。

了解更多 ↗

部署卷插件

单击部署后，在新打开的页面中配置本地存储 配置本地存储。

创建存储类

1. 转到 平台管理。
2. 在左侧导航栏中，单击 存储管理 > 存储类。
3. 单击 创建存储类。
4. 选择 **TopoLVM**，然后单击 下一步。
5. 根据以下说明配置存储类参数。

   注意：以下内容作为表单示例呈现；您也可以选择使用 YAML 创建。

参数	描述
名称	存储类的名称，必须在当前集群中唯一。
显示名称	一个可以帮助您识别或筛选的名称，例如存储类的中文描述。
设备类	设备类是 TopoLVM 中对存储设备进行分类的方式，每个设备类对应一组具有相似特性的存储设备。如无特殊要求，请使用 自动分配 设备类。
文件系统	• **XFS** 是一种高性能的日志文件系统，适合处理并行 I/O 工作负载，支持大文件处理和顺畅的数据传输。 • **EXT4** 是 Linux 下的日志文件系统，提供 extent 文件存储，支持大文件处理，最大文件系统容量为 1 EiB，最大文件大小为 16 TiB。
回收策略	持久卷的回收策略。 • 删除：绑定的持久卷将在 PVC 删除的同时被删除。 • 保留：即使删除 PVC，其绑定的持久卷仍会保留。
访问模式	ReadWriteOnce (RWO)：可以由单个节点以读写方式挂载。
PVC 重建	支持 PVC 跨节点重建。当启用此功能后，必须配置 重建等待时间。如果使用此存储类创建的 PVC 所在节点故障，PVC 将在等待时间结束后自动在其他节点重建，以确保业务连续性。 注意： • 重建的 PVC 不包含原始数据。 • 请确保存储节点的数量大于应用实例副本的数量，否则这将影响 PVC 的重建。
分配项目	此类型的持久卷声明仅能在特定项目中创建。 如果当前没有分配项目，该项目可以 稍后更新。

6. 确认配置无误后，单击 创建 按钮。

后续操作

一切准备就绪后，您可以通知开发人员使用 TopoLVM 功能。例如，在容器平台的 存储 > 持久卷声明 页面上创建持久卷声明并将其绑定到 TopoLVM 存储类。

创建 NFS 共享存储类

基于社区 NFS CSI（Container Storage Interface）存储驱动，提供接入多个 NFS 存储系统或账户的能力。

与传统的 NFS 客户端-服务器模型不同，NFS 共享存储使用社区 NFS CSI（Container Storage Interface）存储插件，更加符合 Kubernetes 的设计原则，允许客户端访问多个服务器。

目录

前提条件

- 必须配置 NFS 服务器，并获取其访问方式。目前平台支持三种 NFS 协议版本：`v3`、`v4.0` 和 `v4.1`。您可以在服务器端执行 `nfsstat -s` 来检查版本信息。

部署 NFS 共享存储插件

1. 进入 平台管理。

2. 在左侧导航栏中，单击 存储管理 > 存储类。

3. 单击 创建存储类。

4. 在 **NFS** 共享存储 右侧单击部署，以跳转至 插件 页面。

5. 在 **NFS** 插件右侧单击⋮> 部署。

6. 等待部署状态指示为 部署成功 后完成部署。

创建 NFS 共享存储类

1. 单击 创建存储类。

   说明：以下内容以表单形式呈现，但您也可以选择使用 YAML 完成操作。

2. 选择 **NFS** 共享存储，单击 下一步。

3. 参考以下说明配置相关参数。

参数	说明
名称	存储类的名称。必须在当前集群中唯一。
服务地址	NFS 服务器的访问地址。例如：`192.168.2.11`。
路径	NFS 文件系统在服务器节点中的挂载路径。例如：`/nfs/data`。
NFS 协议版本	当前支持三种版本：`v3`、`v4.0` 和 `v4.1`。
回收策略	持久卷的回收策略。 - 删除：当持久卷声明被删除时，绑定的持久卷也会被删除。 - 保留：即使持久卷声明被删除，绑定的持久卷仍将保留。

参数	说明
访问模式	当前存储支持的所有访问模式。在随后声明持久卷时，只能选择其中一种模式来挂载持久卷。 - 单节点读写 (RWO)：可以被单个节点以读写方式挂载。 - 多节点读写 (RWX)：可以被多个节点以读写方式挂载。 - 多节点只读 (ROX)：可以被多个节点以只读方式挂载。
分配项目	请分配可以使用此类型存储的项目。 如果目前没有项目需要使用此类型存储，您可以暂时不分配任何项目，并在后续更新它们。

4. 一旦确认配置信息正确，单击 创建。

相关操作

设置 NFS 共享存储类的子目录命名规则

部署卷快照组件

卷快照是指持久卷的快照，它是在特定时间点对持久卷的副本。如果集群中使用支持快照功能的持久卷，则可以部署卷快照组件来启用此功能。

目前，平台仅支持为使用存储类动态创建的 PVC 创建卷快照。您可以基于这些快照创建新的 PVC 绑定。

提示：使用快照创建 PVC 时支持的访问模式与使用存储类创建 PVC 时支持的访问模式不同，这些在下表中已用加粗标识。

创建卷快照使用的存储类	单节点读写（**RWO**）	多节点只读（**ROX**）	多节点读写（**RWX**）
TopoLVM	支持	不支持	不支持
CephRBD 块存储	支持	不支持	不支持
CephFS 文件存储	支持	支持	支持

目录

操作步骤

1. 进入 平台管理。

2. 在左侧导航栏中，单击 存储管理 > 卷快照。

3. 单击 快速部署，将跳转至集群插件。

4. 单击 快照控制器 右侧的⋮> 部署，稍等片刻直到部署成功。

创建 PV

手动创建类型为 **HostPath** 或 **NFS** 共享存储 的静态持久卷。

- **HostPath**：将容器所在主机的文件目录挂载到容器中的指定路径（对应于 Kubernetes 的 HostPath），允许容器使用主机的文件系统进行持久化存储。如果主机变得不可访问，则 HostPath 可能也无法访问。
- **NFS** 共享存储：NFS 共享存储使用社区版 NFS CSI（Container Storage Interface）存储插件，更符合 Kubernetes 的设计原则，能够为多个服务提供客户端访问能力。在使用之前，请确保当前集群已部署 **NFS** 存储插件。

目录

前提条件

- 确认要创建的持久卷的大小，并确保后端存储系统目前能够提供相应的存储空间。
- 获取后端存储访问地址、待挂载的文件路径、凭证访问（如需要）及其他相关信息。

示例 PersistentVolume

```yaml
# example-pv.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: example-pv
spec:
  capacity:
    storage: 5Gi (1)
  accessModes:
    - ReadWriteOnce (2)
  persistentVolumeReclaimPolicy: Retain (3)
  storageClassName: manual (4)
  hostPath: (5)
    path: "/mnt/data"
```

1. 存储量。
2. 卷的挂载方式。
3. PVC 删除后会发生什么（保留、删除、回收）。
4. StorageClass 的名称（用于动态绑定）。
5. 存储后端类型。

通过 Web 控制台创建 PV

1. 进入 平台管理。
2. 在左侧导航栏中，点击 存储管理 > 持久卷 **(PV)**。
3. 点击 创建持久卷。
4. 参考以下说明并在配置参数后点击 创建。

存储信息

类型	参数	说明
HostPath	路径	后端存储卷所在节点的文件目录路径。例如：`/etc/kubernetes`。
NFS 共享存储	服务器地址	NFS 服务器的访问地址。
	路径	NFS 文件系统在服务器节点的挂载路径，例如 `/nfs/data`。
	NFS 协议版本	当前平台支持的 NFS 协议版本包括 `v3`、`v4.0` 和 `v4.1`。您可以在服务器端执行 `nfsstat -s` 来查看版本信息。

通过 CLI 创建 PV

```
kubectl apply -f example-pv.yaml
```

访问模式

持久卷的访问模式受后端存储相关参数设置的影响。

访问模式	含义
单节点读写 **(RWO)**	可以被单个节点以读写方式挂载。
多节点读写 **(RWX)**	可以被多个节点以读写方式挂载。
多节点只读 **(ROX)**	可以被多个节点以只读方式挂载。

回收策略

回收策略	含义
删除	删除持久卷声明的同时也会删除绑定的持久卷及后端存储卷资源。 注意：NFS 共享存储类型的持久卷的回收策略不支持 删除。
保留	即使删除持久卷声明，绑定的持久卷和存储数据仍将保留。之后需要手动处理存储数据并删除持久卷。

相关操作

您可以在列表页面右侧点击⋮或在详情页面右上角点击 操作，根据需要更新或删除持久卷。

删除持久卷适用于以下两种场景：

- 删除未被绑定的持久卷：此卷未被写入数据且不再需要写入，删除后可释放相应的存储空间。
- 删除 保留 的持久卷：持久卷声明已被删除，但由于其回收策略为保留，未同时删除。如果持久卷中的数据已备份至其他存储或不再需要，删除它可释放相应的存储空间。

额外资源

- 创建 PVCs

创建 PVCs

创建一个持久卷声明（PVC），并根据需要设置请求的持久卷（PV）的参数。

您可以通过可视化 UI 表单或使用自定义 YAML 编排文件来创建持久卷声明。

目录

先决条件

确保在命名空间中有足够的剩余 存储 配额，以满足此创建操作所需的存储大小。

示例持久卷声明：

```yaml
# example-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-pvc
  namespace: k-1
  annotations: {}
  labels: {}
spec:
  storageClassName: cephfs
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 4Gi
```

通过 **Web** 控制台创建持久卷声明

1. 转到 容器平台。

2. 在左侧边栏中点击 存储 **>** 持久卷声明（**PVC**）。

3. 点击 创建 **PVC**。

4. 根据需要配置参数。

   注意：以下内容作为使用表单方法的示例提供；您也可以切换到 YAML 模式以完成操作。

参数	描述
名称	持久卷声明的名称，必须在当前命名空间内唯一。
创建	- 动态创建：根据存储类动态生成持久卷并绑定。 - 静态绑定：根据配置的参数和现有持久卷进行匹配和绑定。

参数	描述
方式	
存储类	选择动态创建方法后，平台将根据指定存储类中的描述动态创建持久卷。
访问模式	- ReadWriteOnce (RWO)：可以被单个节点以读写模式挂载。 - ReadWriteMany (RWX)：可以被多个节点以读写模式挂载。 - ReadOnlyMany (ROX)：可以被多个节点以只读模式挂载。 提示：建议考虑计划绑定到当前持久卷声明的工作负载实例数量和部署控制器的类型。例如，在创建多实例部署（Deployment）时，由于所有实例使用相同的持久卷声明，因此不建议选择只能附加到单个节点的 RWO 访问模式。
容量	请求的持久卷的大小。
卷模式	- 文件系统：将持久卷绑定为挂载到 Pod 的文件目录。此模式适用于任何类型的工作负载。 - 块设备：将持久卷绑定为挂载到 Pod 的原始块设备。此模式仅适用于虚拟机。
更多	- 标签 - 注释 - 选择器：选择静态绑定方法后，您可以使用选择器来定位带有特定标签的持久卷。持久卷标签可用于表示存储的特殊属性，例如磁盘类型或地理位置。

5. 点击 创建。等待持久卷声明状态变为 `Bound`，表示持久卷已成功匹配。

通过 **CLI** 创建持久卷声明

```
kubectl apply -f example-pvc.yaml
```

操作

- 绑定持久卷声明：在创建需要持久数据存储的应用程序或工作负载时，绑定持久卷声明以请求符合要求的持久卷。
- 使用卷快照创建持久卷声明：这有助于备份应用程序数据并在需要时恢复，确保业务应用程序数据的可靠性。请参阅 使用卷快照。
- 删除持久卷声明：您可以在详情页面的右上角点击 操作 按钮，根据需要删除持久卷声明。在删除之前，请确保持久卷声明未绑定到任何应用程序或工作负载，并且不包含任何卷快照。删除持久卷声明后，平台将根据回收策略处理持久卷，这可能会清除持久卷中的数据并释放存储资源。请根据数据安全考虑谨慎操作。

通过 **Web** 控制台扩展持久卷声明存储容量

1. 在左侧导航栏中，点击存储 > 持久卷声明（PVC）。
2. 找到持久卷声明并点击⋮> 扩展。
3. 填写新容量。
4. 点击扩展。扩展过程可能需要一些时间，请耐心等待。

通过 **CLI** 扩展持久卷声明存储容量

```
kubectl patch pvc example-pvc -n k-1 --type='merge' -p '{
  "spec": {
    "resources": {
      "requests": {
        "storage": "6Gi"
      }
    }
  }
}'
```

INFO

当 PVC 扩展在 Kubernetes 中失败时，管理员可以手动恢复持久卷声明（PVC）状态并取消扩展请求。请参阅 从 PVC 扩展失败中恢复

其他资源

- 如何注释第三方存储能力

使用卷快照

卷快照是持久卷声明（PVC）的一个时间点副本，可用于配置新的持久卷声明（预填充快照数据）或将现有持久卷声明回滚到先前状态，从而实现备份应用数据并根据需要恢复，确保应用数据的可靠性。

目录

先决条件

- 管理员已为当前集群部署了卷快照组件 **Snapshot Controller** 并在存储集群中启用了快照相关功能。
- 持久卷声明必须动态创建，并且其状态必须为 **Bound**。

- 绑定到持久卷声明的存储类必须支持快照功能，例如 **CephRBD** 内置存储、**CephFS** 内置存储或 **TopoLVM**。

示例 **VolumeSnapshot** 自定义资源（**CR**）

这将使用 CSI 快照类创建 example-pvc `PVC` 的快照。

```
# example-snapshot.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: example-pvc-20250527-111124
  namespace: k-1
  labels:
    snapshot.cpaas.io/sourcepvc: example-pvc
  annotations:
    cpaas.io/description: demo
spec:
  volumeSnapshotClassName: csi-cephfs-snapshotclass
  source:
    persistentVolumeClaimName: example-pvc
```

通过 **Web** 控制台创建卷快照

基于指定持久卷声明（**PVC**）创建卷快照

方法一

1. 进入 **Container Platform**。
2. 在左侧导航栏中，点击 **Storage > Persistent Volume Claims (PVC)**。
3. 在列表中点击相应持久卷声明旁的⋮，选择 **Create Volume Snapshot**。
4. 填写快照描述。此描述可以帮助您记录持久卷的当前状态，例如 *应用升级前*。

5. 点击 **Create**。快照所需时间取决于网络状况和数据量，请耐心等待。

   当快照状态变为 `Available` 时，表示创建成功。

方法二

1. 进入 **Container Platform**。
2. 在左侧导航栏中，点击 **Storage > Persistent Volume Claims (PVC)**。
3. 在列表中点击持久卷声明的名称。
4. 切换到 **Volume Snapshots** 标签。
5. 点击 **Create Volume Snapshot**，根据需要配置相关参数。
6. 点击 **Create**。快照所需时间取决于网络状况和数据量，请耐心等待。

   当快照状态变为 `Available` 时，表示创建成功。

以自定义方式创建卷快照

1. 进入 **Container Platform**。
2. 在左侧导航栏中，点击 **Storage > Volume Snapshots**。
3. 点击 **Create Volume Snapshot**，根据需要配置相关参数。
4. 点击 **Create**。快照所需时间取决于网络状况和数据量，请耐心等待。

   当快照状态变为 `Available` 时，表示创建成功。

通过 **CLI** 创建卷快照

```
kubectl apply -f example-snapshot.yaml
```

从卷快照创建持久卷声明

目前，平台仅支持使用从具有 动态配置 的存储类创建的 PVC 来创建卷快照。您可以基于该快照创建新的 PVC 并进行绑定。

注意：从快照创建 PVC 时支持的访问模式与从存储类创建 PVC 时支持的访问模式不同，如表中 加粗 部分所示。

用于创建卷快照的存储类	单节点读写（**RWO**）	多节点只读（**ROX**）	多节点读写（**RWX**）
TopoLVM	支持	不支持	不支持
CephRBD 块存储	支持	不支持	不支持
CephFS 文件存储	支持	支持	支持

方法一

1. 进入 **Container Platform**。
2. 在左侧导航栏中，点击 **Storage > Persistent Volume Claims (PVC)**。
3. 在列表中点击持久卷声明的名称。
4. 切换到 **Volume Snapshots** 标签。
5. 点击相应卷快照旁的⋮，选择 **Create Persistent Volume Claim**，配置相关参数。
6. 点击 **Create**。

方法二

1. 进入 **Container Platform**。
2. 在左侧导航栏中，点击 **Storage > Volume Snapshots**。
3. 在列表中点击相应卷快照旁的⋮，选择 **Create Persistent Volume Claim**，配置相关参数。

4. 点击 **Create**。

额外资源

- 创建 PVCs

实用指南

设置 NFS 共享存储类的子目录命.

- 功能概述
- 使用案例
- 前提条件
- 操作步骤

通用临时卷

- 示例临时卷
- 主要特性
- 何时使用通用临时卷
- 它们与 emptyDir 的区别

使用 emptyI

- 示例 emptyDir
- 可选的 Medium
- 主要特性
- 常见用例

如何注释第三方存储能力

- 第一步：打开存储类配置
- 第二步：填写存储类信息
- 第三步：使用 ConfigMap 注释存储能力
- 第四步：了解支持的存储能力字段
- 第五步：完成存储类配置
- 可选：使用注释的存储类创建 PVC

设置 NFS 共享存储类的子目录命名规则

目录

功能概述

每个使用 NFS 共享存储类创建的持久卷声明（PVC）对应于 NFS 共享中的一个子目录。默认情况下，子目录的命名规则为 `${pv.metadata.name}`（即持久卷名称）。如果默认生成的名称不符合您的要求，您可以自定义子目录命名规则。本文档提供了配置方法和最佳实践，以便自定义命名约定。

使用案例

在 NFS 服务器端，子目录名称可用于识别其在 Kubernetes 中对应的持久卷声明（PVC）。这使得管理员能够监控每个 PVC 的存储使用情况，从而简化运维管理。

前提条件

- 必须配置 NFS 服务器，并获取其访问方式。目前，平台支持三种 NFS 协议版本：`v3`、`v4.0` 和 `v4.1`。您可以在服务器端执行 `nfsstat -s` 来检查版本信息。

操作步骤

1 部署 NFS 共享存储插件

参见 部署 NFS 共享存储插件。

2 创建 NFS 共享存储类

1. 参见 创建 NFS 共享存储类。

2. 在点击 创建 之前，切换到 YAML 视图，并在参数部分添加 `subDir` 配置，以定义子目录的命名规则。

   配置示例

   ```
   parameters:
     subDir: ${pvc.metadata.namespace}_${pvc.metadata.name}_${pv.metadata.name}
   ```

   子目录命名示例

   ```
   default_nfs-pvc-01_pvc-4411db0b-8ec4-461a-8bbd-062d50666249
   ```

   `default` 是 PVC 命名空间，`nfs-pvc-01` 是 PVC 名称，`pvc-4411db0b-8ec4-461a-8bbd-062d50666249` 是 PV 名称。

   注意：

   - `subDir` 字段仅支持以下三个变量，NFS CSI 驱动程序会自动解析：

- `${pvc.metadata.namespace}`：PVC 命名空间。
- `${pvc.metadata.name}`：PVC 名称。
- `${pv.metadata.name}`：PV 名称。

- `subDir` 命名规则 必须 确保子目录名称的唯一性。否则，多个 PVC 可能共享同一子目录，从而导致数据冲突。

推荐配置：

- `${pvc.metadata.namespace}_${pvc.metadata.name}_${pv.metadata.name}`
- `<cluster-identifier>_${pvc.metadata.namespace}_${pvc.metadata.name}_${pv.metadata.name}`

  适用于多个 Kubernetes 集群共享同一 NFS 服务器，此配置通过在子目录命名规则中加入集群特定标识符（例如集群名称）来确保清晰的集群区分。

不推荐配置：

- `${pvc.metadata.namespace}-${pvc.metadata.name}-${pv.metadata.name}`

  避免使用 `-` 作为分隔符，可能导致子目录名称模糊。例如：如果两个 PVC 名称为 `ns-1/test` 和 `ns/1-test`，则都可能生成相同的子目录 `ns-1-test`。

- `${pvc.metadata.namespace}/${pvc.metadata.name}/${pv.metadata.name}`

  请勿将 `subDir` 配置为创建嵌套目录。NFS CSI 驱动程序在 PVC 被删除时仅删除最后一级目录 `${pv.metadata.name}`，从而在 NFS 服务器上留下孤立的父目录。

3. 点击 创建。

INFO

已存在的 StorageClass 不能被修改。

通用临时卷

Kubernetes 中的通用临时卷是一项功能，允许您使用现有的存储类和 CSI 驱动程序为每个 Pod 提供临时（短暂）卷，而无需预先定义持久卷声明（PVC）。

它们结合了动态供给的灵活性和 Pod 级卷声明的简单性。

- 它们是临时卷，自动：
  - 在 Pod 启动时创建
  - 在 Pod 终止时删除
- 使用与持久卷声明相同的底层机制
- 需要支持动态供给的 CSI（容器存储接口）驱动程序

目录

示例临时卷

这将使用指定的 `StorageClass` 自动为 Pod 创建一个临时 PVC。

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: ephemeral-demo
spec:
  containers:
    - name: app
      image: busybox
      command: ["sh", "-c", "echo hello > /data/hello.txt && sleep 3600"]
      volumeMounts:
        - mountPath: /data
          name: ephemeral-volume
  volumes:
    - name: ephemeral-volume
      ephemeral: 1
        volumeClaimTemplate:
          metadata:
            labels:
              type: temporary
          spec:
            accessModes: [ "ReadWriteOnce" ]
            resources:
              requests:
                storage: 1Gi
            storageClassName: standard
```

1. `Pod` 将使用此模板创建一个 `PVC`。

主要特性

特性	描述
临时性	当 Pod 被删除时，卷也会被删除
动态供给	由任何支持动态供给的 CSI 驱动程序提供支持
无需单独的 **PVC**	卷声明直接嵌入在 Pod 规格中
基于 **CSI**	与任何兼容的 CSI 驱动程序（EBS、RBD、Longhorn 等）一起工作

何时使用通用临时卷

- 当您需要具有以下特性的临时存储时：
  - 可调整大小的卷
  - 快照
  - 加密
  - 非节点本地存储（例如，云块存储）
- 理想的用途：
  - 缓存中间数据
  - 临时工作目录
  - 流水线、AI/ML 工作流

它们与 **emptyDir** 的区别

特性	`emptyDir`	通用临时卷
支持的存储	节点的本地磁盘或内存	任何支持 CSI 的后端
存储特性	基本	支持快照、加密等
使用场景	简单的临时存储	复杂的临时存储需求
可重新调度	否（与节点绑定）	是（如果 CSI 卷可附加）

使用 emptyDir

在 Kubernetes 中，emptyDir 是一种简单的临时卷类型，为 Pod 在其生命周期内提供临时存储。它在 Pod 被分配到节点时创建，并在 Pod 从该节点移除时删除。

目录

示例 emptyDir

该 Pod 创建了一个挂载在 /data 的临时卷，并与容器共享。

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: emptydir-demo
spec:
  containers:
    - name: app
      image: busybox
      command: ["sh", "-c", "echo hello > /data/hello.txt && sleep 3600"]
      volumeMounts:
        - mountPath: /data
          name: cache-volume
  volumes:
    - name: cache-volume
      emptyDir: {}
```

可选的 **Medium** 设置

您可以选择数据存储的位置：

```yaml
emptyDir:
  medium: "Memory"
```

Medium	描述
(默认)	根据您的环境使用节点的磁盘、SSD 或网络存储
`Memory`	使用 RAM (`tmpfs`) 以获得更快的访问速度（但不持久）

主要特性

特性	描述
初始为空	创建时没有数据

特性	描述
跨容器共享	同一卷可以被 Pod 中的多个容器使用
随 Pod 删除	当 Pod 被移除时，卷也会被销毁
节点本地	卷存储在节点的本地磁盘或内存中
快速	适合性能敏感的临时存储空间

常见用例

- 缓存中间构建工件
- 缓冲日志
- 临时工作目录
- 在同一 Pod 中的容器之间共享数据（如 sidecar）

如何注释第三方存储能力

随着公有云和私有云环境的广泛使用，第三方存储集成变得越来越重要。本指南将引导您如何使用 ConfigMap 注释第三方存储能力，使您的平台能够自动识别并展示这些能力。

目录

第一步：打开存储类配置

1. 在平台的 UI 中，进入 **Platform Management**。
2. 从左侧边栏导航至 **Storage Management** > **Storage Classes**。
3. 点击 **Create Storage Class** 开始定义新的存储类。

第二步：填写存储类信息

在表单中提供以下详细信息：

字段	描述
Name	新存储类的名称。
Storage Class	选择或定义存储类标识符。
Provisioner	输入存储插件使用的 provisioner 名称。

第三步：使用 **ConfigMap** 注释存储能力

要启用能力注释，请在 `kube-public` 命名空间中创建一个带有适当标签和数据格式的 **ConfigMap**。

示例 **YAML**：

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: sd-built-in
  namespace: kube-public
  labels:
    features.alauda.io/type: StorageDescription
data:
  storage.type1.com: |-
    type: Filesystem
    volumeMode:
    - Filesystem
    accessModes:
    - ReadWriteOnce
    - ReadWriteMany
    - ReadWriteOncePod
  storage.type2.com: |-
    type: Filesystem
    snapshot: true
    volumeMode:
    - Filesystem
    - Block
    accessModes:
    - ReadWriteOnce
    - ReadOnlyMany
    - ReadWriteOncePod
```

关键点：

- **metadata.name**：必须以 `sd-` 开头，例如 `sd-configmap1`。
- **metadata.namespace**：必须是 `kube-public`。
- **metadata.labels**：包含 `features.alauda.io/type = StorageDescription`。
- **data**：
  - 每个 **key** 对应存储类中的 `provisioner` 字段。
  - 每个 **value** 是描述存储支持能力的 YAML 字符串。

第四步：了解支持的存储能力字段

以下是您可以在 ConfigMap 中定义的支持字段：

能力	字段	选项	默认值	说明
类型	`type`	`Filesystem`、`Block`	—	如果省略或无效，类型显示为未知。
快照	`snapshot`	`true`、`false`	`false`	如果为 `false` 或无效，表单 UI 中禁用快照创建功能。
卷模式	`volumeMode`	`Filesystem`、`Block`	`Filesystem`	使用 `Block` 模式的 PVC 不支持目录挂载。
访问模式	`accessModes`	`ReadWriteOnce`、`ReadOnlyMany`、`ReadWriteMany`、`ReadWriteOncePod`	—	如果省略或无效，UI 中不显示访问模式选项。`ReadWriteOncePod` 当前不支持表单启用。

第五步：完成存储类配置

设置完上述内容后：

1. 点击 **Create** 保存您的存储类。
2. 平台会自动匹配 `provisioner` 与 ConfigMap，并用定义的能力注释存储类。

可选：使用注释的存储类创建 PVC

通过 表单 **UI** 创建 Persistent Volume Claim (PVC) 时，只会显示来自注释 ConfigMap 的支持能力。未支持的选项将不会出现。

故障排除

从 PVC 扩容失败中恢复

操作步骤

其他提示

从 PVC 扩容失败中恢复

当 Kubernetes 中的 PVC 扩容失败时，管理员可以手动恢复 Persistent Volume Claim (PVC) 状态并取消扩容请求。

目录

操作步骤

1. 修改绑定到 PVC 的 Persistent Volume (PV) 的回收策略为 `Retain`。为此，编辑对应的 PV 并将 `persistentVolumeReclaimPolicy` 字段设置为 `Retain`。

2. 删除原有的 PVC。

3. 手动编辑 PV，删除其规格中的 `claimRef` 条目。这样可以确保新的 PVC 能绑定到该 PV，使 PV 状态变为 `Available`。

4. 重新创建一个较小尺寸或底层存储提供商支持的尺寸的新 PVC。

5. 在新 PVC 中显式指定 `volumeName` 字段，使其与原 PV 名称匹配。这样可以确保新 PVC 准确绑定到指定的 PV。

6. 最后，恢复 PV 的原始回收策略。

其他提示

- 确保所使用的 `StorageClass` 已通过将 `allowVolumeExpansion` 设置为 `true` 启用卷扩容功能。
- 请谨慎执行这些操作，以避免数据丢失的风险。