# 存储

## Ceph 分布式存储



## MinIO 对象存储

# TopoLVM 本地存储

介绍

安装

前提条件

操作步骤

操作指南

# Ceph 分布式存储

## 介绍

### 介绍

功能概览

存储方案对比

## 安装

### 创建标准类型集群

前提条件

注意事项

操作步骤

相关操作

### 创建延伸类型集群

名词解释

典型部署方案

约束与限制

前提条件

操作步骤

相关操作

## 架构

### 架构

技术架构

## 核心概念

### 核心概念

Rook Operator

Ceph CSI

Ceph 模块功能

## 操作指南

### 接入存储服务

前提条件

操作步骤

后续操作

### 管理存储池

创建存储池

删除存储池

查看对象存储池地址

### 节点特定组件

更新组件部署配

重启存储组件

### 监控与告警

监控

告警

## 实用指南

### 配置专用集群以支持分布式存储

架构

基础架构要求

步骤

后续行动

### 清理分布式存储

注意事项

操作步骤

### 数据容灾

### 更新优化参数

操作步骤

## 权限说明

### 权限说明

# 介绍

**Alauda Container Platform (ACP) Storage with Ceph** 是平台在集群内提供的一种超融合存储解决方案。基于开源的 Rook + Ceph 存储方案，分布式存储实现自动管理、自动扩容和自动修复能力，满足中小型应用的块存储、文件存储和对象存储需求。

> **NOTE**
>
> 本文档中，分布式存储 指本集群内的 Ceph 存储，外部存储 指本集群外的 Ceph 存储。

## 目录



## 功能概览

- 简易部署：提供存储集群的图形化自动部署和管理服务；支持计算与存储的集成部署和解耦部署两种模式。
- 专业运维：提供持久卷快照备份和克隆新卷功能；容量、性能及组件级别的可视化监控；内置告警策略，满足大多数存储运维场景需求。

- 安全可靠：分布式多副本机制保障数据安全与可靠；简便可靠的自动化管理支持存储资源的在线扩容。
- 卓越性能：提供弹性高性能存储服务；支持混合磁盘设备部署，提升存储系统性能和效率。

# 存储方案对比

平台支持以下两种存储方案，您可以任选其一。

## 创建存储集群

| 需求 | 优势 |
| --- | --- |
| 您可以选择创建标准型集群或扩展型集群 | 无需额外准备存储方案，配置可在业务集群上完成，节省成本。 |

## 接入外部存储

方案一：接入平台内其他业务集群的分布式存储资源，确保存储与业务隔离，便于管理和维护。

方案二：将外部 Ceph 存储资源集成为分布式存储。

| 需求（任选其一） | 优势 |
| --- | --- |
| 方案一：其他业务集群已部署分布式存储。 | 可充分利用跨集群存储资源，避免业务变更带来的干扰。确保存储数据安全稳定，降低运维复杂度。<br><br>注意：若接入的存储为不同平台的分布式存储，如灾备环境中的主备平台，请采用集成外部 Ceph 的方式。 |
| 方案二：平台外部的 **Ceph** 存储，版本 **≥ 14.2.3**。 | 相较于直接创建存储类，该方式更便于使用平台界面进行卷快照、扩容等功能操作。 |

注意：若需维护外部存储的存储池、存储设备等配置，必须在存储集群的管理界面中进行操作。

# 安装

## 创建标准类型集群

前提条件

注意事项

操作步骤

相关操作

## 创建延伸类型集群

名词解释

典型部署方案

约束与限制

前提条件

操作步骤

相关操作

# 创建标准类型集群

标准类型集群是 Ceph 存储的最典型部署方式。它将数据副本分布在不同主机的硬盘上，确保单个主机故障时，其他主机上的数据副本仍能保持服务可用性。

## 目录



## 前提条件

- 存储集群中至少需要 3 个节点。
- 每个节点上必须至少有 1 块空白硬盘或 1 块未格式化的硬盘分区可用。
- 可用的硬盘容量建议大于 50 G。

- 如果您使用的是接入的 Kubernetes 集群，且运行时组件为 Containerd，请确保集群所有节点的 `/etc/systemd/system/containerd.service` 文件中 `LimitNOFILE` 参数值配置为 `1048576`，以确保分布式存储的成功部署。有关配置说明，请参考 修改 Containerd 配置信息。

  注意：当从 v3.10.2 之前的版本升级到当前版本时，如果你需要在自建的运行时组件为 Containerd 的 Kubernetes 集群上部署 Ceph 分布式存储，还必须将集群所有节点的 `/etc/systemd/system/containerd.service` 文件的 `LimitNOFILE` 参数值配置为 `1048576`。

# 注意事项

创建存储服务 和 接入存储服务 仅支持选择一种方式。

# 操作步骤

## 1 部署 **Operator**

1. 进入 平台管理。
2. 在左侧导航栏中，单击 存储管理 > 分布式存储。
3. 单击 立即配置。
4. 在 部署 **Operator** 向导页面，单击右下角的 部署 **Operator** 按钮。
   - 当页面自动进入下一步时，表示 Operator 部署成功。
   - 如果部署失败，请根据界面提示选择 清理已部署信息并重试，然后重新部署 Operator；如果您希望返回分布式存储选择页面，请单击 应用商店，先卸载已部署的 **rook-operator** 内的资源实例，然后再卸载 **rook-operator**。

## 2 创建集群

1. 在 创建集群 向导页面中，配置相关参数并单击右下角的 创建集群 按钮。

| 参数 | 说明 |
| --- | --- |
| 集群类型 | 选择 标准。 |
| 设备类类型 | 设备类是硬盘的分组；您可以根据存储需求自定义设备类，将不同存储内容分配给不同性能的硬盘。<br>• 默认设备类：平台将自动对集群节点中的硬盘类型进行分类。例如，创建名为 `hdd`、`ssd`、`nvme` 的设备类。<br>• 自定义设备类：为节点中特定硬盘组合自定义设备类名称；支持添加多个设备类。同一硬盘只能属于一个设备类。 |
| 设备类 - 名称 | 设备类的名称。当选择 自定义设备类 时，设备类不能使用以下名称：`hdd`、`ssd`、`nvme`。 |
| 设备类 - 存储设备 | 在节点中选择 空白硬盘 或 未格式化的硬盘分区。<br>• 当打开“所有空设备”开关时：所有空设备将被添加到该设备类；<br>• 当关闭“所有空设备”开关时：手动输入节点下的空设备名称，例如 `sda`。 |
| 快照 | 开启后，支持创建 PVC 快照并使用快照配置新的 PVC，以便快速备份和恢复业务数据。<br>如果在创建存储时未启用快照，仍然可以在存储集群详情页面的 操作 部分按需启用。<br>注意：请确保已为当前集群 部署卷快照插件。 |
| 监控告警 | 开启后，将提供开箱即用的监控指标采集和告警提醒功能，详见 监控与告警。<br>注意：如果此时未开启，您需要寻找其他解决方案来监控和告警存储。例如，在运维中心手动配置监控仪表板和告警策略。 |

2. 单击 高级配置 进行组件的高级配置。

| 参数 | 说明 |
| --- | --- |
| 网络配置 | • 主机网络：存储集群将使用主机网络，您在优化参数列中填写相关网络优化参数，例如配置 `public` 和 `cluster` 网段。如果为空，系统将使用默认主机网段。<br>注意：使用主机网络可能会因为明文（未加密）传输数据而产生安全风险。请联系平台支持团队以获取加密传输解决方案。<br>• 容器网络：存储集群将使用容器网络；您可以在网络管理中创建子网并将其分配到 `rook-ceph` 命名空间。如果为空，系统将使用默认子网。<br>注意：<br>不支持 IPv6。<br>当使用容器网络时，存储只可在集群内访问。<br>Ceph CSI Pod 的失败或重启可能会导致服务中断。 |
| 优化参数 | 支持使用 Ceph 配置文件格式填写参数；系统将根据填写的内容覆盖默认参数。<br>注意：首次填写或修改初始化参数后，请单击初始化参数，必须先成功初始化才能创建集群。 |
| 组件定点部署 | 您可以将组件定点部署到指定节点；至少需要三个节点以确保最小可用性。<br>可进行定点部署的组件包括 MON、MGR、MDS、RGW。 |

- 当页面自动进入下一步时，表示 Ceph 集群部署成功。
- 如果创建失败，您可以单击清理 已创建信息或重试，自动清理资源并重新创建集群，或根据文档 分布式存储服务资源清理 手动清理资源。

## 3 创建存储池

1. 在 创建存储池 向导页面中，配置相关参数并单击右下角的 创建存储池 按钮。

| 参数 | 说明 |
| --- | --- |
| 存储类型 | • 文件存储：提供安全、可靠、可扩展的共享文件存储服务。适用于文件共享、数据备份等场景。 |

| 参数 | 说明 |
| --- | --- |
| | • 块存储：提供高 IOPS 和低延迟的存储服务。适用于数据库、虚拟化等场景。<br>• 对象存储：提供标准 S3 接口的存储服务，适用于大数据、备份归档、云存储等场景。 |
| 副本数量 | 副本数量越大，冗余度与数据安全性越高；但存储的利用率将降低。通常设置为 3 以满足大多数需求。 |
| 设备类 | 对同一类型的设备或相同业务逻辑的硬盘进行统一分类，从上一步已添加的设备类中进行选择。<br>• 选择设备类时，数据将存储在所选设备类中。<br>• 如果未选择设备类，数据将随机存储在存储池中的所有设备中。 |

如果是对象存储，您还需配置以下参数：

| 参数 | 说明 |
| --- | --- |
| 区域 | 指定存储池所在的区域。 |
| 网关类型 | 默认是 S3，且无法修改。 |
| 内部端口 | 指定集群内部访问的端口。 |
| 外部访问 | 启用/禁用外部访问将创建/销毁 Nodeport 类型的 Service。 |
| 实例数量 | 对象存储的资源实例数量。 |

- 当页面自动进入下一步时，表示存储池部署成功。
- 如果部署失败，请根据界面提示检查核心组件，然后单击 清理已创建信息并重试 以重新创建存储池。

2. 单击 创建存储池。在 详细信息 页签中，您可以查看创建的存储池信息。

# 相关操作

## 创建延伸类型集群

具体请参考 创建延伸类型集群。

## 清理分布式存储

具体请参考 清理分布式存储。

# 创建延伸类型集群

延伸类型的集群能够在两个地理位置不同的区域进行扩展，为存储基础设施提供灾难恢复的能力。灾难发生时，当两个可用区中的一个完全不可用，Ceph 依然可以保持可用性。

## 目录



## 名词解释

| 名词 | 解释 |
| --- | --- |
| 仲裁可用区 | 通常位于独立的区域，不承载主要业务负载，专注于维护集群的一致性，主要用于在主数据中心发生故障或网络分区时进行仲裁决策。 |
| 数据可用区 | 是 Ceph 集群中实际存储和处理数据的主要区域，承载业务负载和数据存储任务，与仲裁区共同构成了一个完整的高可用存储系统。 |

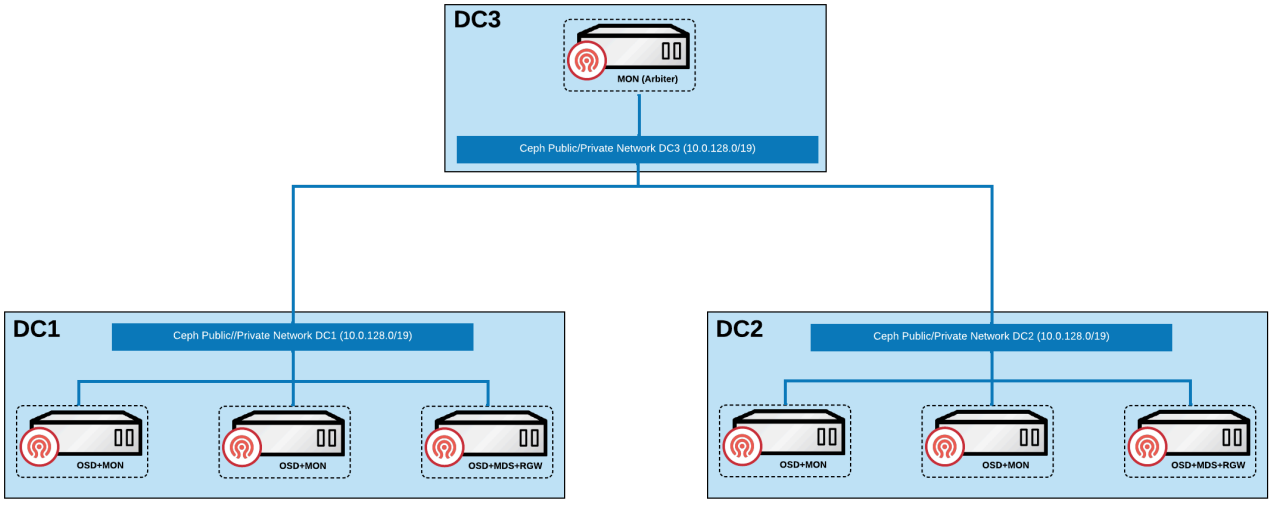
# 典型部署方案

下述内容提供了延伸类型集群的典型部署方案，并提供了组成说明及故障恢复的原理说明。

## 组成说明

节点需要划分在三个可用区中，包括两个数据可用区和一个仲裁可用区。

- 两个数据可用区均需要完整部署所有核心 Ceph 组件（MON、OSD、MGR、MDS、RGW），且每个数据可用区必须配置两个 MON 实例以实现高可用。当同一数据可用区内的两个 MON 实例均不可用时，系统将判定该可用区为异常状态。
- 仲裁可用区仅需部署一个 MON 实例，作为仲裁决策节点。




## 故障恢复说明

- 当某个数据可用区完全故障时，Ceph 集群将自动进入降级状态并触发告警通知，系统会将存储池的最小副本数（min_size）从默认的 2 调整为 1，由于另外一个数据可用区内依然维持双副本运行，此时集群状态为可用。当故障的数据可用区恢复后，系统会自动执行数据同步并恢复至健康状态；如果故障无法修复，则建议使用新的数据可用区进行替换。
- 当两个数据可用区之间的网络连接中断，但它们均可正常连接至仲裁可用区时，仲裁可用区将基于预设策略对两个数据可用区进行仲裁，并选择状态更优的一方作为主数据区继续提供服务。

# 约束与限制

- 存储池限制：不支持纠删码存储池，只能使用副本机制进行数据保护。
- 设备分类限制：不支持设备类功能，无法按设备特性进行存储分层。
- 区域部署限制：仅支持存在两个数据可用区，不能存在两个以上的数据可用区。
- 数据均衡要求：两个数据可用区的 OSD 权重必须严格保持一致，以确保数据分布的均衡性。
- 存储介质要求：必须使用全闪存（All-Flash）OSD 配置，可最小化在连接恢复后恢复所需的时间，并尽可能减少数据丢失的可能性。
- 网络延迟要求：两个数据可用区之间的 RTT（往返延迟）不能超过 10ms，仲裁可用区需要满足 ETCD 规范的延迟要求，以确保仲裁机制的可靠性。

# 前提条件

请提前将集群中的全部或部分节点划分至三个可用区中，要求如下：

- 请确保至少有 5 个节点分布在一个仲裁可用区及两个数据可用区中。其中，仲裁可用区至少存在一个节点，可以使用虚拟机或云主机。
- 请确保三个可用区中至少有一个可用区存在 Master 节点（控制节点）。
- 请确保至少有 4 个计算节点均匀分布在 2 个数据可用区中，且每个数据可用区中至少配置 2 个计算节点。

- 请尽量保证两个数据可用区的节点数量和磁盘配置一致。

# 操作步骤

## 1 为节点添加标签

1. 进入 平台管理。
2. 在左侧导航栏中，单击 集群管理 > 集群。
3. 单击相应集群名称进入集群概览页面。
4. 切换至 节点 页签。
5. 根据 前提条件 中的规划，分别为这些节点添加 `topology.kubernetes.io/zone=<zone>` 标签，即可将节点划分至指定的可用区。其中，需使用可用区的名称替换 <zone> 部分。

## 2 创建存储服务

本文档中仅介绍与标准类型集群不同的参数，其他参数请参考 创建标准类型集群。

创建集群

| 参数 | 说明 |
| --- | --- |
| 集群类型 | 选择 延伸。 |
| 仲裁可用区 | 选择仲裁可用区的名称。 |
| 数据可用区 | 选择可用区名称，并选择节点。 |

创建存储池

| 参数 | 说明 |
| --- | --- |
| 副本数量 | 默认为 4。 |
| 实例数量 | 当存储类型为 对象存储 时，为保证可用性，实例数最小值为 2，最大不超过 5。 |

# 相关操作

## 创建标准类型集群

具体说明请参考 创建标准类型集群。

## 清理分布式存储

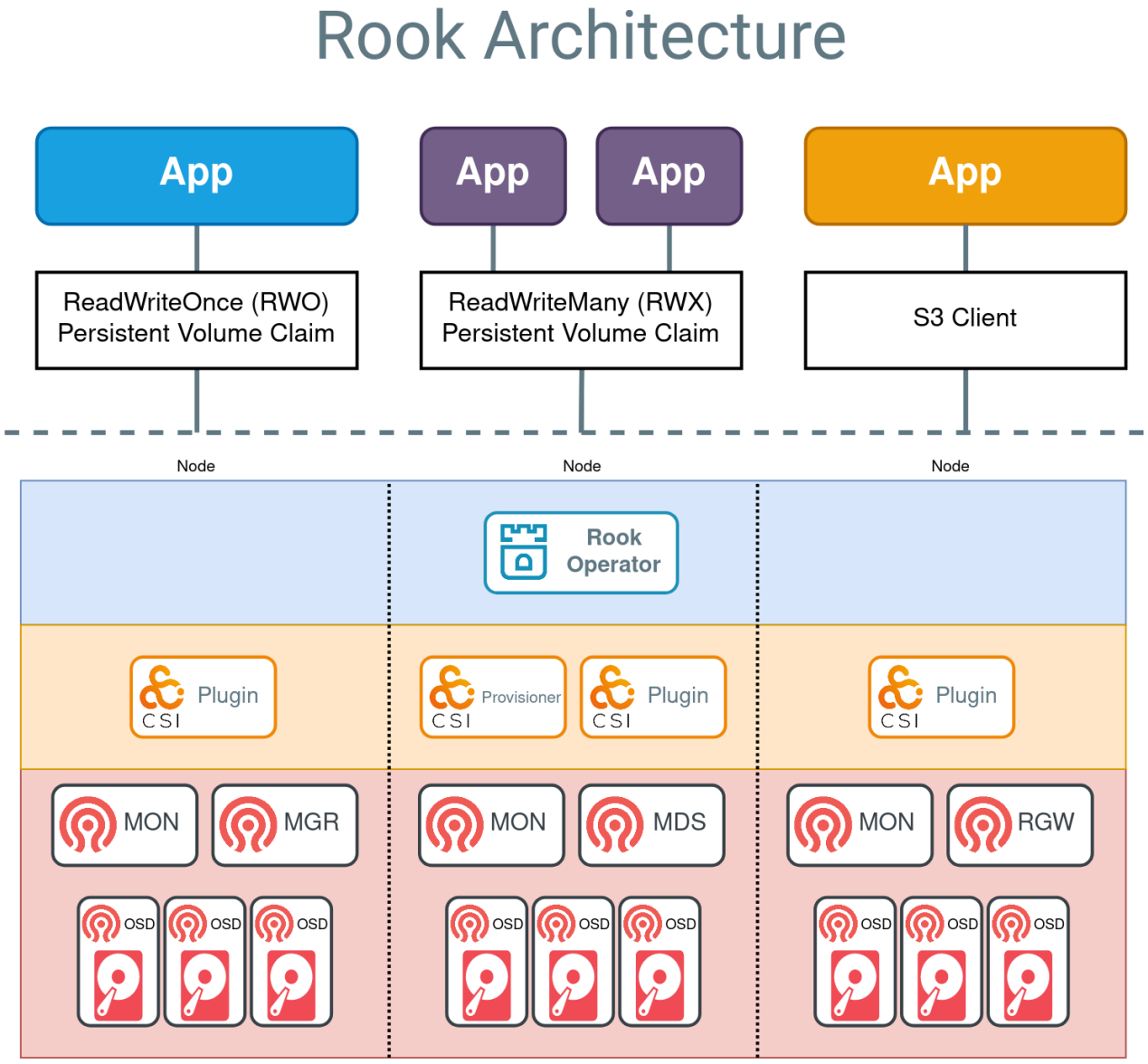
具体请参考 清理分布式存储。

# 架构

## 目录



## 技术架构




上面展示了三种存储类型的示例应用程序：

- 块存储通过一个蓝色应用表示，该应用挂载了一个支持 ReadWriteOnce（RWO）的存储卷。应用程序可以读写 RWO 卷，而 Ceph 管理 IO 操作。
- 共享文件系统由两个紫色应用表示，它们共同使用一个支持 ReadWriteMany（RWX）的存储卷。两个应用程序可以同时主动读取或写入该卷。Ceph 将通过 MDS 守护进程确保数据在多个写入者的情况下得到安全保护。
- 对象存储通过一个橙色应用来体现，该应用可以使用标准的 S3 客户端对存储桶进行读写操作。

在上述图表中，虚线以下的部分可以分为三大类：

- Rook Operator（蓝色层）：Operator 负责自动配置 Ceph。

- CSI 插件和 Provisioner（橙色层）：Ceph-CSI 驱动负责卷的供应和挂载。
- Ceph 守护进程（红色层）：Ceph 守护进程运行核心存储架构。

块存储
在上图中，创建一个带有 RWO 卷的应用程序的流程如下：

- （蓝色）应用程序创建 PVC 以请求存储。
- PVC 定义了 Ceph RBD 存储类（sc）以供应存储。
- Kubernetes 调用 Ceph-CSI RBD Provisioner 创建 Ceph RBD 镜像。
- Kubelet 调用 CSI RBD 卷插件将卷挂载到应用程序中。
- 卷现在可用于读写操作。
- 一个 ReadWriteOnce 卷一次只能挂载到一个节点上。

共享文件系统
在上图中，创建一个带有 RWX 卷的应用程序的流程如下：

- （紫色）应用程序创建 PVC 以请求存储。
- PVC 定义了 CephFS 存储类（sc）以供应存储。
- Kubernetes 调用 Ceph-CSI CephFS Provisioner 创建 CephFS 子卷。
- Kubelet 调用 CSI CephFS 卷插件将卷挂载到应用程序中。
- 卷现在可用于读写操作。
- 一个 ReadWriteMany 卷可以挂载到多个节点上，供应用程序使用。

对象存储 **S3**
在上图中，创建一个可以访问 S3 存储桶的应用程序的流程如下：

- （橙色）应用程序创建 BucketClaim 以请求存储桶。
- Ceph COSI Driver 创建 Ceph RGW 存储桶。
- Ceph COSI Driver 创建一个包含访问存储桶凭据的 Secret。
- 应用程序从 Secret 中获取凭据。
- 应用程序现在可以使用 S3 客户端对存储桶进行读写操作。

# 核心概念

# 核心概念

## 目录



## Rook Operator

Rook Operator 是一个简单的容器，包含了启动和监控存储集群所需的所有组件。Operator 会启动并监控 Ceph 监视器 Pods、Ceph OSD 守护进程，以提供 RADOS 存储，同时还会启动和管理其他 Ceph 守护进程。Operator 通过管理存储池、对象存储（S3/Swift）以及文件系统的 CRDs 来初始化运行服务所需的 Pods 和其他资源。

Operator 会监控存储守护进程，以确保集群的健康状态。Ceph 监视器会在必要时启动或进行故障转移，同时随着集群的扩展或收缩进行其他调整。Operator 还会监视 Ceph 自定义资源（CRs）中指定的期望状态变更，并应用这些变更。

Rook 会自动配置 Ceph-CSI 驱动，将存储挂载到您的 Pods 中。rook/ceph 镜像包含了管理集群所需的所有工具。

## Ceph CSI

Ceph CSI 插件实现了支持 CSI 的容器编排器（CO）与 Ceph 集群之间的接口。它们支持动态配置 Ceph 卷并将其挂载到工作负载中。

# Ceph 模块功能

| 模块 | 功能描述 |
| --- | --- |
| MON | 监视器（Monitor，MON）是 Ceph 集群中最重要的组件，负责管理集群并维护整个集群的状态。MON 确保集群的相关组件可以在同一时刻达到同步，作为集群的领导者，由 MON 守护进程负责收集、更新和发布集群信息。 |
| MGR | 管理器（Manager，MGR）是一个监控系统，提供数据采集、存储、分析（包括报警）和可视化的功能。它将某些集群参数提供给外部系统使用。通常，Ceph 的 MGR 守护进程与 MON 守护进程一起运行。 |
| OSD | 对象存储守护进程（OSD）存储实际的用户数据。每个 OSD 通常绑定到一个物理磁盘，负责处理来自客户端的读写请求。 |
| MDS | Ceph 元数据服务器（MDS）跟踪文件层次结构，并存储仅供 CephFS 使用的元数据。RBD 和 RGW 不需要元数据。MDS 不直接为客户端提供数据服务。 |
| RGW | RADOS 网关（RGW）是 Ceph 对象网关，提供与 S3 和 Swift 兼容的 RESTful API。同时 RGW 还支持多租户和 OpenStack 身份服务（Keystone）。 |
| RADOS | 可靠自我修复分布式对象存储（RADOS）是 Ceph 存储集群的核心，任何数据都以对象的形式存储，不论其数据类型。RADOS 层通过数据复制、故障检测与恢复，以及跨集群节点的数据恢复，确保数据的一致性和可靠性。 |
| LIBRADOS | Librados 是简化 RADOS 访问的方法，目前支持 PHP、Ruby、Java、Python、C 和 C++ 等编程语言，为 Ceph 存储集群提供 RADOS 的本地接口，并且是其他服务如 RADOS 块设备（RBD）和 RADOS 网关（RGW）的基础组件。此外，它还为 Ceph 文件系统（CephFS）提供可移植操作系统接口（POSIX）。通过 Librados API，开发者可以直接访问 RADOS，从而创建自己的访问 Ceph 集群存储的接口。 |
| RBD | RADOS 块设备（RBD）是 Ceph 的块设备，提供对外的块存储。它可以像磁盘一样被映射、格式化和挂载到服务器。 |
| CephFS | Ceph 文件系统（CephFS）提供一个兼容 POSIX 的分布式文件系统。它依赖 Ceph MDS 跟踪文件层次结构，即元数据的管理。 |

# 操作指南

## 接入存储服务

- 前提条件
- 操作步骤
- 后续操作

## 管理存储池

- 创建存储池
- 删除存储池
- 查看对象存储池地址

## 节点特定组件

- 更新组件部署配
- 重启存储组件

## 监控与告警

- 监控
- 告警

# 接入存储服务

接入存储服务支持两种接入方式：首先，接入平台内其他业务集群的分布式存储资源，以确保存储与业务的隔离，从而便于管理和维护；其次，连接外部 Ceph 存储资源以进行分布式存储使用。

## 目录



## 前提条件

### 准备存储

请选择以下任意一种方式：

- 已在其他业务集群中部署分布式存储并创建了存储池。请记录存储池的名称，以便后续接入使用。

- 已创建外部 Ceph 存储（版本 ≥ 14.2.3），并建立了存储池。请记录存储池的名称，以便后续接入使用。

## 开放端口

| 目的 **IP** | 目的端口 | 源 **IP** | 源端口 |
| --- | --- | --- | --- |
| Ceph 节点的 IP | 3300, 6789, 6800-7300, 7480 | 业务集群中所有节点的 IP | any |

## 获取认证信息（外部 **Ceph**）

如果准备的存储是外部 Ceph 存储，需要通过以下命令获取认证信息。

| 参数 | 获取方式 |
| --- | --- |
| **FSID** | `ceph fsid` |
| **MON** 组件信息 | `ceph mon dump`<br>需为 {name= IP} 格式，例如 *a=192.168.100.100:6789* 。 |
| **Admin Key** | `ceph auth get-key client.admin` |
| 存储池 | • 文件存储：使用 `ceph fs ls` 命令获得的 `name` 值。<br>• 块存储：`ceph osd dump` \| `grep "application rbd"` \| `awk '{print $3}'` |
| *数据存储池* | （仅文件存储需要）使用 `ceph fs ls` 命令获得的 `data pools` 值。 |

# 操作步骤

说明：以下步骤以 接入外部 **Ceph** 存储 为例，接入分布式存储的操作类似。

> 1. 在左侧导航栏中，单击 存储管理 > 分布式存储。

2. 单击 接入存储。

3. 在 接入配置 向导页面中，选择 外部 **Ceph**。

| 参数 | 说明 |
| --- | --- |
| 快照 | 开启后，支持创建 PVC 快照并使用快照快速配置新 PVC 以备份和恢复业务数据。<br>如果在接入存储时未开启快照，您仍可以在存储集群详细信息页面的 操作 部分根据需要后续开启。<br>注意：使用前请确保已为当前集群 部署了卷快照插件。 |
| 网络配置 | • 主机网络：本集群中的计算组件将使用 主机网络 访问 存储集群。<br>• 容器网络：本集群中的计算组件将使用 容器网络 访问 存储集群。您可以在网络管理中创建子网并将其分配到 `rook-ceph` 命名空间。如果不指定，则使用默认子网。 |
| 其他参数 | 请填写在前提条件中获取的外置 Ceph 认证参数。 |

4. 在 创建存储类 向导页面中，完成配置并单击 接入。

| 参数 | 说明 |
| --- | --- |
| 类型 | 根据前述创建的存储池类型，将默认对应存储类：<br>• 文件存储：CephFS 文件存储<br>• 块存储：CephRBD 块存储 |
| 回收策略 | 持久卷的回收策略。<br>• 删除：当持久卷声明被删除时，绑定的持久卷也将被删除。<br>• 保留：即使持久卷声明被删除，被绑定的持久卷仍将被保留。 |

| 参数 | 说明 |
| --- | --- |
| 分配项目 | 可以使用此类型存储的项目。<br>如果当前没有项目需要此类存储，您可以暂时不分配项目，并在后续更新时进行分配。 |

5. 等待大约 1-5 分钟以完成接入。

# 后续操作

- 创建存储类：CephFS 文件存储, CephRBD 块存储
- 使用上述存储类创建持久卷声明的开发人员可以通过卷快照和扩容功能进行扩展使用。

说明：如果需要维护外部存储的存储池、存储设备配置等，操作必须在存储集群的管理平台进行。

# 管理存储池

存储池指用于存储数据的逻辑分区。单个存储集群支持同时使用不同类型的存储池，如文件存储和块存储，以满足各种业务需求。

## 目录



## 创建存储池

除了在配置分布式存储时创建的存储池外，您还可以创建其他类型的存储池。

提示：在同一存储集群中，仅允许创建一个文件存储和一个对象存储池，同时最多可以创建八个块存储池。

### 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 > 分布式存储。

3. 在 集群信息 页签下，滑动至 存储池 区域，单击 创建存储池。

4. 根据以下说明配置相关参数。

| 参数 | 说明 |
| --- | --- |
| 存储类型 | 选择当前未部署的存储类型。<br>- 文件存储：提供安全、可靠、可扩展的共享文件存储服务，适用于文件共享、数据备份等。<br>- 块存储：提供高 IOPS 和低延迟存储服务，适用于数据库、虚拟化等。<br>- 对象存储：提供标准 S3 接口存储服务，适用于大数据、备份归档、云存储服务等。 |
| 副本数量 | • 当集群类型为标准：较高的副本数量增加冗余和数据安全性，但也会降低存储利用率。通常，设置为 3 可满足大部分需求。<br>• 当集群类型为延伸：默认副本数量为 4，无法修改。 |
| 设备类 | • 当集群类型为标准：选择在创建的存储池中已经添加的设备类别。<br>  • 选择设备类时，数据将存储在所选设备类中。<br>  • 如果未选择设备类，则数据将随机存储在存储池中的所有设备中。<br>• 当集群类型为标准：不支持添加设备类别。 |

如果是对象存储类型，您还可以配置以下参数：

| 参数 | 说明 |
| --- | --- |
| 区域 | 指定存储池所在区域。 |
| 网关类型 | 默认为 S3，无法修改。 |
| 内部端口 | 指定集群内部访问的端口。 |
| 外部访问 | 开启/关闭外部访问将创建/销毁 NodePort 类型的服务。 |

| 参数 | 说明 |
| --- | --- |
| 实例数量 | 对象存储的资源实例数量。 |

5. 单击 创建。

# 删除存储池

如果某种存储不再需要，可以在解除与存储类的关联后删除存储池。

## 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 > 分布式存储。
3. 在 集群信息 页签下，滑动至 存储池 区域，单击要删除的存储池旁边的⋮> 删除。
4. 阅读提示信息并输入存储池名称。
5. 单击 删除。

# 查看对象存储池地址

创建对象存储池后，可以查看存储池的内部和外部访问地址。

## 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 > 分布式存储。
3. 在 集群信息 页签下，滑动至 存储池 区域，单击对象存储池旁边的⋮> 查看地址。

# 节点特定组件部署

在创建分布式存储后，您仍然可以查看和修改组件的部署位置，以便于存储扩展和维护。

## 目录



## 更新组件部署配置

### 注意事项

- 更新配置将触发系统自动重建组件实例，可能会影响服务访问存储系统，建议在非高峰时段进行更新。
- 当集群类型为 **Extend** 时，不支持组件的固定部署功能。

### 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 > 分布式存储。

3. 在 存储组件 页签下，单击 组件部署配置。

4. 根据业务需求启用/禁用 固定部署 开关，并将组件部署到指定节点。节点数量必须不少于三个，以确保最小可用性。适用于固定部署配置的组件包括 MON、MGR、MDS、RGW。

5. 单击 更新，组件将开始调度至指定节点。

# 重启存储组件

当您删除已部署的存储组件时，系统将根据当前组件部署策略自动重新调度和重新部署组件到节点。

## 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 > 分布式存储。

3. 在 存储组件 页签下，单击组件名称右侧的⋮> 删除。

# 添加设备/设备类

## 目录



## 添加设备类

为集群节点中同一类型的设备或同一业务逻辑的硬盘统一划分类型，根据存储需求自定义设备类，将不同存储内容分配至不同类型的存储硬盘中。

### 注意事项

当集群类型为 延伸 时，不支持添加设备类。

### 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 > 分布式存储。

3. 单击 设备类 页签。

4. 单击 添加设备类，参考以下说明配置相关参数。

| 参数 | 说明 |
| --- | --- |
| 名称 | 设备类的名称，设备类不能使用以下名称：`hdd`、`ssd`、`nvme`。 |
| 存储设备 | 在节点中选择 空白硬盘 或 未被格式化的硬盘分区。<br>• 打开所有空设备开关时：将节点下的所有空设备添加至该设备类；<br>• 关闭所有空设备开关时：手动输入节点下的空设备名称，例如 `sda`。 |

# 添加设备

将可用硬盘映射成存储设备来使用和管理。

说明：添加硬盘为存储设备后，不支持在界面上更新或移除。

## 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 > 分布式存储。

3. 单击 设备类 页签。

4. 在设备类右侧，单击 添加设备，参考以下说明配置相关参数。

| 参数 | 说明 |
| --- | --- |
| 节点类型 | 选择要添加为存储设备的硬盘所在节点的类型。<br>计算节点：未添加过存储设备的节点。<br>存储节点：已添加过存储设备的节点。 |
| | |

| 参数 | 说明 |
| --- | --- |
| 添加类型 | 选择添加硬盘为存储设备的方式。<br>所有空硬盘：选择添加节点中全部未分区挂载的硬盘为存储设备。<br>指定硬盘：选择添加节点中的部分硬盘为存储设备，包括空硬盘或已分区挂载的硬盘。<br>当节点类型为 存储节点 时，只能选择 指定硬盘。 |
| 指定硬盘 | 当添加类型为 指定硬盘 时，输入所有要添加为存储设备的硬盘名称，例如 `sda`、`sdb`，每输入一个硬盘名称之后，按回车确定。<br>注意：建议使用整个硬盘作为存储设备，而不是硬盘上的分区。 |

5. 单击 添加。

# 硬盘状态

- 正常：对应存储设备状态为 IN+UP。
- 异常：对应存储设备状态为 IN+DOWN。
- 离线：对应存储设备状态为 OUT+UP。
- 故障：对应存储设备状态为 OUT+DOWN。

# 监控与告警

分布式存储提供了开箱即用的监控指标采集和告警提醒能力。启用监控与告警功能后，可从存储集群、存储性能及存储组件等方面进行监控和告警，且支持配置通知策略。

直观呈现的监控数据可用于为运维巡检或性能调优提供决策支持，完善的告警和通知机制也将帮助保障存储系统的稳定运行。

提示：如果创建分布式存储时未启用监控与告警功能，您只能另行寻找存储监控与告警的替代方案。例如，在运维中心手动配置监控面板和告警策略。

## 目录



## 监控

平台默认会收集分布式存储的读写性能、CPU 及内存使用量等常用监控指标。在 存储管理 > 分布式存储 的 监控 页签中，可查看指标的实时监控数据。

## 存储概览

监控存储的健康状态、物理容量使用情况以及 OSD/MON 组件活动数，存储状态异常时可查看告警原因。

## 性能监控

从集群、存储池及 OSD 三个维度监控读写带宽和读写 IOPS；同时，针对 OSD 还可监控读写延迟。

## 组件监控

监控 MON、OSD 等组件的 CPU 使用量和内存使用量。

# 告警

平台默认启用了一批告警策略，一旦资源异常或监控数据达到预警状态，将自动触发告警。预置策略已能满足组件和集群状态告警、设备容量告警，以及用户数据告警等常见运维需求。

# 配置通知

为了能及时收到告警，建议您在运维中心设置通知策略：将告警信息以邮件、短信等方式发送给相关人员，提醒其采取必要的措施解决问题或避免故障发生。单击 告警配置 可切换至运维中心完成操作，参考 创建告警策略。

# 处理告警

- 若监控到存储集群为 `告警` 状态，表示当前已触发告警，且相关异常可能导致故障。请及时查看 实时告警 详情，并结合故障原因定位及排障。
- 若监控到存储集群为 `故障` 状态，表示存储集群已无法正常运行。请立即定位问题并排障。

下表为预置策略所用告警等级的含义，可作为您制定告警处理原则的参考。

| 告警等级 | 含义 |
| --- | --- |
| 灾难 | 告警规则对应的资源发生故障，导致平台业务中断、数据丢失，影响程度重大。 |
| 严重 | 告警规则对应的资源存在已知问题，可能导致平台功能故障，影响业务正常运行。 |
| 警告 | 告警规则对应的资源存在运行风险，如不及时处理，可能影响业务正常运行。 |

## 故障复盘

告警历史 中记录了所有曾经触发，当前已无须处理的告警。借助告警历史进行故障复盘时，为了能有效地达到经验总结目的，您可能需要回答以下问题。

- 事故发生时，具体的异常情况是什么。
- 告警列表中某条告警反复出现，是否有规律可循，能否在下次发生之前提前避免。
- 时间轴显示某个时段告警激增，是不可抗力导致还是运维事故，是否需要调整运维方案。

# 实用指南

## 配置专用集群以支持分布式存储



## 清理分布式存储



## 数据容灾

术语
备份配置
故障切换



# 更新优化参数

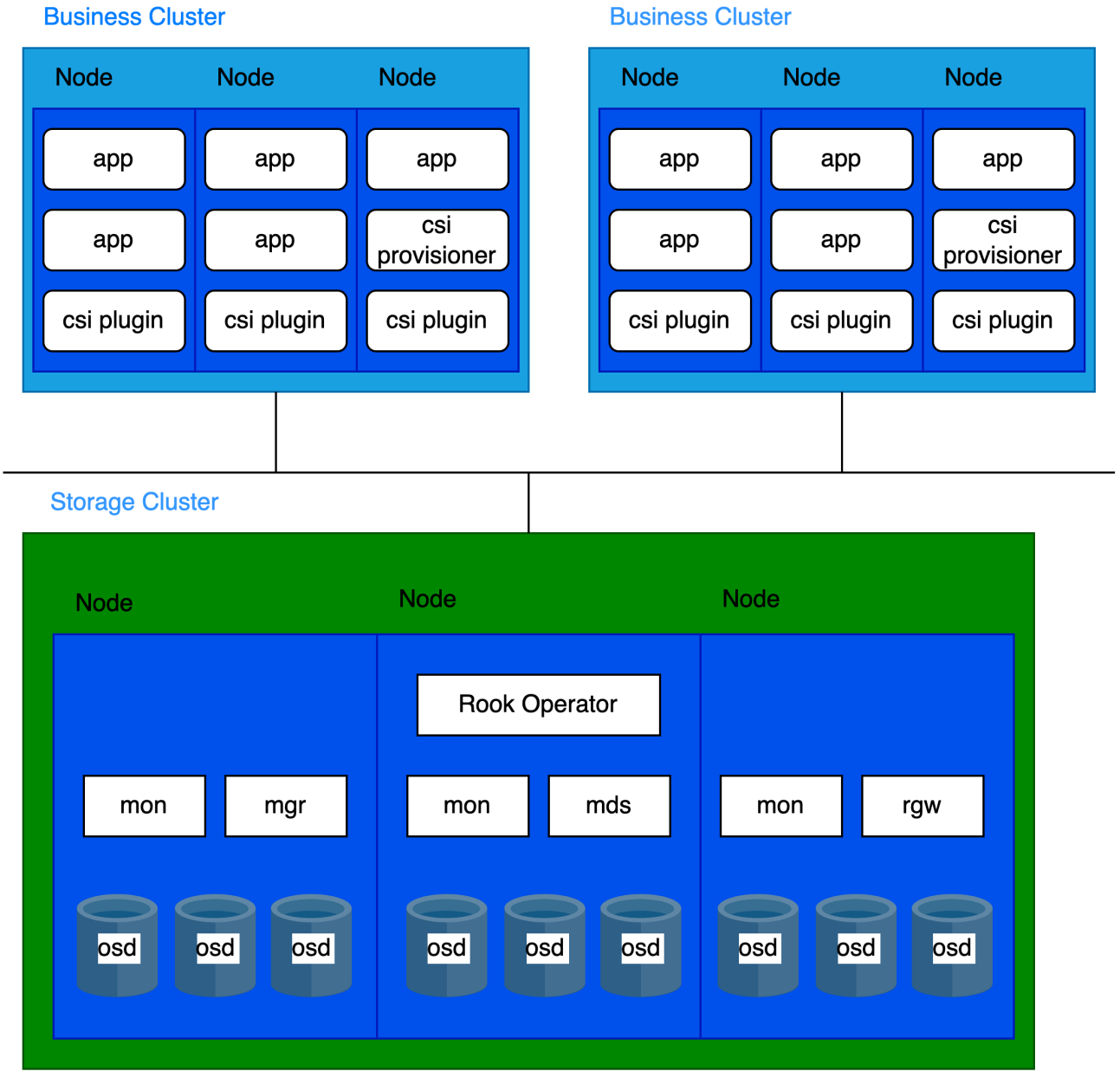
# 配置专用集群以支持分布式存储

专用集群部署是指使用一个独立的集群来部署平台的分布式存储，平台内的其他业务集群通过接入来访问和利用它提供的存储服务。
为了保证平台分布式存储的性能和稳定性，只将平台核心组件和分布式存储组件部署在专用存储集群中，避免其他业务工作负载的共置。这种分离部署方法是平台分布式存储的最佳实践。

## 目录

# 架构

存算分离架构




# 基础架构要求

# 平台要求

仅支持 3.18 及以上版本。

# 集群要求

推荐使用裸金属集群作为专用存储集群。

# 资源要求

有关分布式存储部署的组成部分，请参考核心概念。

每个组件有不同的 CPU 和内存需求，推荐配置如下：

| 进程 | **CPU** | 内存 |
| --- | --- | --- |
| MON | 2c | 3Gi |
| MGR | 3c | 4Gi |
| MDS | 3c | 8Gi |
| RGW | 2c | 4Gi |
| OSD | 4c | 8Gi |

一个集群通常运行：

- 3 个 MON
- 2 个 MGR
- 多个 OSD
- 2 个 MDS（如果使用 CephFS）
- 2 个 RGW（如果使用 Ceph 对象存储）

基于组件分布，以下每个节点的资源建议适用：

| **CPU** | 内存 |
| --- | --- |
| 16c + (4c * OSD 每节点) | 20Gi + (8Gi * OSD 每节点) |

# 存储设备要求

建议每个节点部署 12 个或更少的存储设备。这有助于限制节点故障后的恢复时间。

## 存储设备类型要求

推荐使用企业级 SSD，单个设备容量不超过 10TiB，并确保所有硬盘的大小和类型一致。

## 容量规划

在部署之前，根据具体业务需求规划存储容量。默认情况下，分布式存储系统采用 3 副本冗余策略。因此，可用容量是所有存储设备总原始容量除以 3。

以30(N)个节点（副本数 = 3）为例，可用容量场景如下：

| 存储设备大小**(D)** | 节点上存储设备数量**(M)** | 总容量**(D*M*N)** | 可用容量**(D*M*N/3)** |
| --- | --- | --- | --- |
| 0.5 TiB | 3 | 45 TiB | 15 TiB |
| 2 TiB | 6 | 360 TiB | 120 TiB |
| 4 TiB | 9 | 1080 TiB | 360 TiB |

## 容量监控与扩容

1. 主动容量规划

   始终确保可用存储容量超过消耗量。如果存储完全耗尽，恢复需要手动干预，无法通过简单删除或迁移数据来解决。

2. 容量告警

   集群会在两个阈值触发告警：

- **80%** 使用率（“接近满”）：需主动 释放空间 或进行集群扩容。
- **95%** 使用率（“已满”）：存储已完全耗尽，标准命令无法释放空间。请立即联系平台支持。

务必及时处理警报，并定期监控存储使用情况，以避免服务中断。

3. 扩容建议

- 避免：为现有节点添加存储设备。
- 推荐：通过添加新存储节点进行扩容。
- 要求：新节点必须使用与现有节点相同大小、类型和数量的存储设备。

# 网络要求

分布式存储必须使用 主机网络。

## 网络隔离

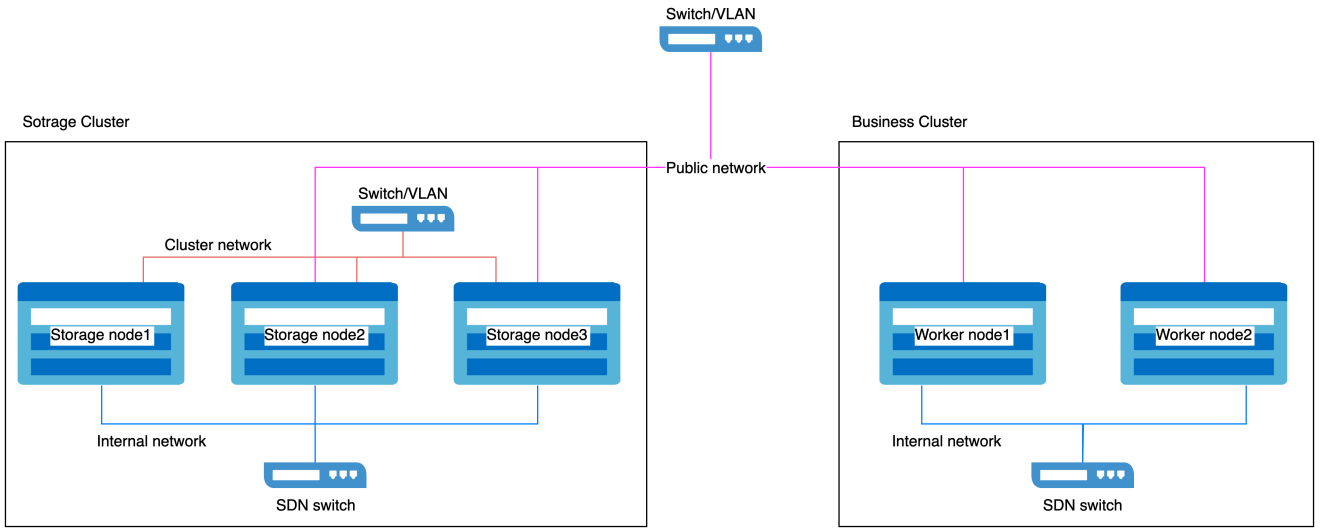
网络分为两种类型：

- 公共网络：用于客户端与存储组件之间的交互（例如，I/O 请求）。
- 集群网络：专门用于副本之间的数据复制和数据再平衡（例如，恢复）。

为了确保服务质量和性能稳定：

1. 对于专用存储集群：
   每个主机保留两个网络接口：
   - 公共网络：用于客户端和组件的通信。
   - 集群网络：用于内部复制和再平衡流量。
2. 对于业务集群：
   每个主机保留一个网络接口以访问存储公共网络。

网络隔离配置示例




## 网络接口速度要求

1. 存储节点

   - 公共网络和集群网络要求 10GbE 或更高的网络接口。

2. 业务集群节点

   - 用于访问存储 公共网络 的网络接口必须为 10GbE 或更高。

# 步骤

## 1 部署 **Operator**

1. 进入 平台管理。

2. 在左侧导航栏中，单击 存储管理 > 分布式存储。

3. 单击 立即配置。

4. 在 部署 **Operator** 向导页中，单击右下角 部署 **Operator** 按钮。

   - 当页面自动进入下一步骤时，表示 Operator 已成功部署。
   - 如果部署失败，请参考界面提示 清理已部署信息并重试，重新部署 Operator；如果希望返回分布式存储选择页面，请单击 应用商店，首先卸载已经部署的 **rook-operator** 内的资源，然后卸载 **rook-operator**。

## 2 创建 Ceph 集群

在存储集群的 控制节点 执行命令。

▶ 点击查看

参数：

- 公共网络 **CIDR**：存储的 公共网络 的 CIDR（例如，`- 10.0.1.0/24`）。
- 集群网络 **CIDR**：存储的 集群网络 的 CIDR（例如，`- 10.0.2.0/24`）。
- 存储设备：指定分布式存储使用的存储设备。
  格式示例：

```
nodes:
- name: storage-node-01
  devices:
  - name: /dev/disk/by-id/wwn-0x5000cca01dd27d60
  useAllDevices: false
- name: storage-node-02
  devices:
  - name: sdb
  - name: sdc
  useAllDevices: false
- name: storage-node-03
  devices:
  - name: sdb
  - name: sdc
  useAllDevices: false
```

> **提示**
>
> 使用磁盘的 WWN（世界唯一标识符）进行稳定命名，避免依赖于重启后可能发生变化的 `sdb` 等易变设备路径。

## 3 创建存储池

提供三种存储池类型，根据您的业务需求选择并创建相应的存储池。

### 创建文件存储池

在存储集群的 控制节点 执行命令。

▶ 点击查看

### 创建块存储池

在存储集群的 控制节点 执行命令。

▶ 点击查看

### 创建对象存储池

在存储集群的 控制节点 执行命令。

▶ 点击查看

## 后续行动

当其他集群需要使用分布式存储服务时，请参考以下指南。
接入存储服务

# 清理分布式存储

如果需要删除 rook-ceph 集群并重新部署一个新的集群，请按照本文依次清理与分布式存储服务相关的资源。

## 目录



## 注意事项

在清理 rook-ceph 之前，请确保所有使用 Ceph 存储的 PVC 和 PV 资源已被删除。

# 操作步骤

## 1 删除快照类

1. 删除快照类。

```
kubectl delete VolumeSnapshotClass csi-cephfs-snapshotclass csi-rbd-snapshotclass
```

2. 验证快照类是否已清理完毕。

```
kubectl get VolumeSnapshotClass | grep csi-cephfs-snapshotclass
kubectl get VolumeSnapshotClass | grep csi-rbd-snapshotclass
```

当这两个命令没有任何输出时，表示清理完毕。

## 2 删除存储类

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⋮> 删除，删除所有使用 Ceph 存储解决方案的存储类。

## 3 删除存储池

此步骤需要在上一个步骤完成后执行。

1. 进入 平台管理。
2. 在左侧导航栏中，单击 存储管理 > 分布式存储。
3. 在 存储池区域，单击⋮> 删除，逐个删除所有存储池。当存储池区域显示 无存储池 时，说明存储池已成功删除。
4. （可选）如果集群模式为 延伸，还需在集群的 Master 节点上执行以下命令，以删除创建的内置存储池。

```
kubectl -n rook-ceph delete cephblockpool -l cpaas.io/builtin=true
```

回显信息：

```
cephblockpool.ceph.rook.io "builtin-mgr" deleted
```

## 4 删除 **ceph-cluster**

此步骤需要在上一个步骤完成后执行。

1. 更新 ceph-cluster，并启用清理策略。

```
kubectl -n rook-ceph patch cephcluster ceph-cluster --type merge -p '{"spec":{"cleanupPolicy":{"confirmation":"yes-really-destroy-data"}}}'
```

2. 删除 ceph-cluster。

```
kubectl delete cephcluster ceph-cluster -n rook-ceph
```

3. 删除执行清理的作业。

```
kubectl delete jobs --all -n rook-ceph
```

4. 验证 ceph-cluster 清理是否完毕。

```
kubectl get cephcluster -n rook-ceph | grep ceph-cluster
```

当该命令没有任何输出时，表示清理完毕。

## 5 删除 **rook-operator**

此步骤需要在上一个步骤完成后执行。

1. 删除 rook-operator。

```
kubectl -n rook-ceph delete subscriptions.operators.coreos.com roo
k-operator
```

2. 验证 rook-operator 清理是否完毕。

```
kubectl get subscriptions.operators.coreos.com -n rook-ceph | grep
rook-operator
```

当该命令没有任何输出时，表示清理完毕。

3. 验证 ConfigMap 是否清理完毕。

```
kubectl get configmap -n rook-ceph
```

当该命令没有任何输出时，表示清理完毕。如果有输出结果，请执行以下命令清理，替换 `<configmap>` 为实际输出。

```
kubectl -n rook-ceph patch configmap <configmap> --type merge -p
'{"metadata":{"finalizers": []}}'
```

4. 验证 Secret 是否清理完毕。

```
kubectl get secret -n rook-ceph
```

当该命令没有任何输出时，表示清理完毕。如果有输出结果，请执行以下命令清理，替换 `<secret>` 为实际输出。

```
kubectl -n rook-ceph patch secrets <secret> --type merge -p '{"met
adata":{"finalizers": []}}'
```

5. 验证 rook-ceph 清理是否完毕。

```
kubectl get all -n rook-ceph
```

当该命令没有任何输出时，表示清理完毕。

## 6 执行清理脚本

完成上述步骤后，表示 Kubernetes 和 Ceph 相关资源已被清空，接下来需要清理宿主机上残留的 rook-ceph。

### 清理脚本

清理脚本 clean-rook.sh 的内容如下：

▶ 点击查看

### 注意事项

清理脚本依赖于 sgdisk 命令，请确保在执行清理脚本之前已安装该命令。

- Ubuntu 安装命令：`sudo apt install gdisk`
- RedHat 或 CentOS 安装命令：`sudo yum install gdisk`

### 操作步骤

1. 在每台部署分布式存储的业务集群机器上执行清理脚本 clean-rook.sh。

   ```
   sh clean-rook.sh /dev/[设备名称]
   ```

   示例：`sh clean-rook.sh /dev/vdb`

   执行时将提示确认是否真正清空该设备。如果确认，输入 yes 即可开始清理。

2. 使用 `lsblk -f` 命令检查分区信息。当该命令的输出中 `FSTYPE` 列为空时，表示清理完毕。

# 数据容灾

# 文件存储灾备

CephFS Mirror 是 Ceph 文件系统的一个功能，旨在实现不同 Ceph 集群之间的异步数据复制，从而提供跨集群的灾难恢复。其核心功能是以主备模式同步数据，确保当主集群发生故障时，备份集群能够快速接管服务。

**WARNING**

- CephFS Mirror 基于快照进行增量同步，默认快照间隔为每小时一次（可配置）。主备集群之间的差异数据通常是一个快照周期内写入的数据量。
- CephFS Mirror 仅提供底层存储数据的备份，无法处理 Kubernetes 资源的备份。请结合平台的备份与恢复 功能对 PVC 和 PV 资源进行备份。

## 目录

# 术语

| 术语 | 说明 |
| --- | --- |
| 主集群 | 当前提供存储服务的集群。 |
| 备集群 | 用于备份的集群。 |

# 备份配置

## 前提条件

- 准备两个适合部署 Alauda Container Platform (ACP) Storage 并使用 Ceph 的集群，分别为主集群和备集群，确保集群间网络互通。
- 两个集群使用的平台版本（v3.12 及以上）必须保持一致。
- 在主集群和备集群中分别创建分布式存储服务。
- 在主集群和备集群中创建同名的文件存储池。

## 操作步骤

### 1 在备集群启用文件存储池的镜像功能

在备集群的控制节点执行以下命令：

命令行

```
kubectl -n rook-ceph patch cephfilesystem <fs-name> \
--type merge -p '{"spec":{"mirroring":{"enabled": true}}}'
```

输出

```
cephfilesystem.ceph.rook.io/<fs-name> patched
```

参数说明：

- `<fs-name>`：文件存储池名称。

## 2 获取 Peer Token

该令牌是建立两个集群镜像连接的关键凭证。

在备集群的控制节点执行以下命令：

命令

```
kubectl get secret -n rook-ceph \
$(kubectl -n rook-ceph get cephfilesystem <fs-name> -o jsonpath='{.st
atus.info.fsMirrorBootstrapPeerSecretName}') \
-o jsonpath='{.data.token}' | base64 -d
```

输出

```
# 由于涉及敏感信息，输出已截断。
eyJmc2lkIjogImMyYjAyNmMzLTA3ZGQtNDA3Z...
```

参数说明：

- `<fs-name>`：文件存储池名称。

## 3 在主集群创建 Peer Secret

获取到备集群的 Peer Token 后，需要在主集群创建 Peer Secret。

在主集群的控制节点执行以下命令：

命令

```
kubectl -n rook-ceph create secret generic fs-secondary-site-secret \
--from-literal=token=<token> \
--from-literal=pool=<fs-name>
```

输出

```
secret/fs-secondary-site-secret created
```

参数说明：

- `<token>`：在步骤 2中获取的令牌。
- `<fs-name>`：文件存储池名称。

## 4 在主集群启用文件存储池的镜像功能

在主集群的控制节点执行以下命令：

命令

```
kubectl -n rook-ceph patch cephfilesystem <fs-name> --type merge -p \
'{
  "spec": {
    "mirroring": {
      "enabled": true,
      "peers": {
        "secretNames": [
          "fs-secondary-site-secret"
        ]
      },
      "snapshotSchedules": [
        {
          "path": "/",
          "interval": "<schedule-interval>"
        }
      ],
      "snapshotRetention": [
        {
          "path": "/",
          "duration": "<retention-policy>"
        }
      ]
    }
  }
}'
```

示例

```
kubectl -n rook-ceph patch cephfilesystem cephfs --type merge -p \
'{
  "spec": {
    "mirroring": {
      "enabled": true,
      "peers": {
        "secretNames": [
          "fs-secondary-site-secret"
        ]
      },
      "snapshotSchedules": [
        {
          "path": "/",
          "interval": "1h"
        }
      ],
      "snapshotRetention": [
        {
          "path": "/",
          "duration": "h 1"
        }
      ]
    }
  }
}'
```

输出

```
cephfilesystem.ceph.rook.io/<fs-name> patched
```

参数说明：

- `<fs-name>`：文件存储池名称。
- `<schedule-interval>`：快照执行周期。详情请参考官方文档↗。
- `<retention-policy>`：快照保留策略。详情请参考官方文档↗。

## 5 在主集群部署 Mirror Daemon

Mirror Daemon 持续监控文件存储池（启用镜像）的数据变化，定期创建快照并将快照差异通过网络推送到备集群。

在主集群的控制节点执行以下命令：

命令

```
cat << EOF | kubectl apply -f -
apiVersion: ceph.rook.io/v1
kind: CephFilesystemMirror
metadata:
  name: cephfs-mirror
  namespace: rook-ceph
spec:
  placement:
    tolerations:
    - key: NoSchedule
      operator: Exists
  resources:
    limits:
      cpu: "500m"
      memory: "1Gi"
    requests:
      cpu: "500m"
      memory: "1Gi"
  priorityClassName: system-node-critical
EOF
```

输出

```
cephfilesystemmirror.ceph.rook.io/cephfs-mirror created
```

# 故障切换

当主集群发生故障时，可以直接继续使用备集群中的 CephFS。

## 前提条件

主集群的 Kubernetes 资源已备份并恢复到备集群，包括 PVC、PV 以及应用的工作负载。

# 块存储灾难恢复

RBD Mirror 是 Ceph 块存储（RBD）的一项功能，支持不同 Ceph 集群之间的异步数据复制，实现跨集群的灾难恢复（DR）。其核心功能是以主备模式同步数据，确保主集群故障时备份集群能够快速接管服务。

**WARNING**

- RBD Mirror 基于快照进行增量同步，默认快照间隔为每小时一次（可配置）。主备集群之间的差异数据通常对应于一个快照周期内的写入。
- RBD Mirror 仅提供底层存储数据备份，不包含 Kubernetes 资源备份。请使用平台的 备份与恢复 功能备份 PVC 和 PV 资源。

## 目录

---

# 术语

| 术语 | 说明 |
| --- | --- |
| **Primary Cluster** | 当前提供存储服务的集群。 |
| **Secondary Cluster** | 用作备份的备用集群。 |

# 备份配置

## 前提条件

- 准备两个能够部署 Alauda Container Platform (ACP) Storage 并使用 Ceph 的集群：一个 Primary 集群和一个 Secondary 集群，且两者之间网络互通。
- 两个集群必须运行相同的平台版本（v3.12 或更高）。
- 在 Primary 和 Secondary 集群中分别创建分布式存储服务。
- 在 Primary 和 Secondary 集群中创建同名的块存储池。
- 请确保以下三个镜像已上传至平台私有镜像仓库：
  - `quay.io/csiaddons/k8s-controller:v0.5.0`
  - `quay.io/csiaddons/k8s-sidecar:v0.8.0`
  - `quay.io/brancz/kube-rbac-proxy:v0.8.0`

# 操作步骤

## 1 启用 **Primary** 集群块存储池镜像功能

在 Primary 集群的 Control 节点执行以下命令：

Command

```
kubectl -n rook-ceph patch cephblockpool <block-pool-name> \
--type merge -p '{"spec":{"mirroring":{"enabled":true,"mode":"imag
e"}}}'
```

Output

```
cephblockpool.ceph.rook.io/<block-pool-name> patched
```

参数说明：

- `<block-pool-name>`：块存储池名称。

## 2 获取 **Peer Token**

该令牌是建立集群镜像连接的关键凭证。

在 Primary 集群的 Control 节点执行以下命令：

Command

```
kubectl get secret -n rook-ceph \
$(kubectl get cephblockpool.ceph.rook.io <block-pool-name> -n rook-ce
ph -o jsonpath='{.status.info.rbdMirrorBootstrapPeerSecretName}') \
-o jsonpath='{.data.token}' | base64 -d
```

Output

```
# 输出因敏感信息被截断
eyJmc2lkIjoiMjc2N2I3ZmEtY2YwYi00N...
```

参数说明：

- `<block-pool-name>`：块存储池名称。

## 3 在 Secondary 集群创建 Peer Token Secret

在 Secondary 集群的 Control 节点执行以下命令：

Command

```
kubectl -n rook-ceph create secret generic rbd-primary-site-secret \
--from-literal=token=<token> \
--from-literal=pool=<block-pool-name>
```

Output

```
secret/rbd-primary-site-secret created
```

参数说明：

- `<token>`：从步骤 2获取的令牌。
- `<block-pool-name>`：块存储池名称。

## 4 启用 Secondary 集群块存储池镜像功能

在 Secondary 集群的 Control 节点执行以下命令：

Command

```
kubectl -n rook-ceph patch cephblockpool <block-pool-name> --type mer
ge -p \
'{
  "spec": {
    "mirroring": {
      "enabled": true,
      "mode": "image",
      "peers": {
        "secretNames": [
          "rbd-primary-site-secret"
        ]
      }
    }
  }
}'
```

Output

```
cephblockpool.ceph.rook.io/<block-pool-name> patched
```

参数说明：

- `<block-pool-name>`：块存储池名称。

## 5 在 Secondary 集群部署 Mirror Daemon

该守护进程负责监控和管理 RBD 镜像同步进程，包括数据同步和错误处理。

在 Secondary 集群的 Control 节点执行以下命令：

Command

```
cat << EOF | kubectl apply -f -
apiVersion: ceph.rook.io/v1
kind: CephRBDMirror
metadata:
  name: rbd-mirror
  namespace: rook-ceph
spec:
  count: 1
EOF
```

Output

```
cephrbdmirror.ceph.rook.io/rbd-mirror created
```

## 6 验证镜像状态

在 Secondary 集群的 Control 节点执行以下命令：

Command

```
kubectl get cephblockpools.ceph.rook.io <block-pool-name> -n rook-ceph -o jsonpath='{.status.mirroringStatus.summary}'
```

Output

```
# 所有 "OK" 状态表示运行正常
{"daemon_health":"OK","health":"OK","image_health":"OK","states":{}}
```

参数说明：

- `<block-pool-name>`：块存储池名称。

## 7 启用 **Replication Sidecar**

该功能支持高效的数据复制和同步，且不会中断主应用操作，提升系统的可靠性和可用性。

1. 部署 csiaddons-controller

在 Primary 和 Secondary 集群的 Control 节点执行以下命令：

▶ 点击查看

参数说明：

- `<registry>`：平台的镜像仓库地址。

2. 启用 csi sidecar

在 Primary 和 Secondary 集群的 Control 节点执行以下命令：

```
kubectl patch cm rook-ceph-operator-config -n rook-ceph --type json --patch \
'[
  {
    "op": "add",
    "path": "/data/CSI_ENABLE_OMAP_GENERATOR",
    "value": "true"
  },
  {
    "op": "add",
    "path": "/data/CSI_ENABLE_CSIADDONS",
    "value": "true"
  }
]'
```

## 8 创建 VolumeReplicationClass

在 Primary 和 Secondary 集群的 Control 节点执行以下命令：

Command

```
cat << EOF | kubectl apply -f -
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplicationClass
metadata:
  name: rbd-volumereplicationclass
spec:
  provisioner: rook-ceph.rbd.csi.ceph.com
  parameters:
    mirroringMode: snapshot
    schedulingInterval: "<scheduling-interval>" 1
    replication.storage.openshift.io/replication-secret-name: rook-csi-rbd-provisioner
    replication.storage.openshift.io/replication-secret-namespace: rook-ceph
EOF
```

Output

```
volumereplicationclass.replication.storage.openshift.io/rbd-volumereplicationclass created
```

1. `<scheduling-interval>`：调度间隔（例如，schedulingInterval: "1h" 表示每小时执行一次）。

## 9 为 PVC 启用镜像功能

在 Primary 集群的 Control 节点执行以下命令：

Command

```
cat << EOF | kubectl apply -f -
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplication
metadata:
  name: <vr-name> 1
  namespace: <namespace> 2
spec:
  autoResync: false
  volumeReplicationClass: rbd-volumereplicationclass
  replicationState: primary
  dataSource:
    apiGroup: ""
    kind: PersistentVolumeClaim
    name: <pvc-name> 3
EOF
```

Output

```
volumereplication.replication.storage.openshift.io/<mirror-pvc-name>
created
```

1. `<vr-name>`：VolumeReplication 对象名称，建议与 PVC 名称相同。
2. `<namespace>`：VolumeReplication 所属命名空间，必须与 PVC 命名空间一致。
3. `<pvc-name>`：需要启用镜像的 PVC 名称。

注意 启用后，Secondary 集群中的 RBD 镜像将变为只读。

# 故障切换

当 Primary 集群发生故障时，需要切换 RBD 镜像的主备关系。

## 前提条件

- 已将 Primary 集群的 Kubernetes 资源备份并恢复至 Secondary 集群，包括 PVC、PV、应用工作负载等。

# 操作步骤

## 创建 **VolumeReplication**

在 Secondary 集群的 Control 节点执行以下命令：

```
cat << EOF | kubectl apply -f -
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplication
metadata:
  name: <vr-name> 1
  namespace: <namespace> 2
spec:
  autoResync: false
  dataSource:
    apiGroup: ""
    kind: PersistentVolumeClaim
    name: <mirror-pvc-name> 3
  replicationHandle: ""
  replicationState: primary
  volumeReplicationClass: rbd-volumereplicationclass
EOF
```

1. `<vr-name>`：VolumeReplication 名称。
2. `<namespace>`：PVC 命名空间。
3. `<mirror-pvc-name>`：PVC 名称。

注意 创建后，Secondary 集群上的 RBD 镜像将变为主用且可写。

# 对象存储灾备

Ceph RGW Multi-Site 功能是一种跨集群异步数据复制机制，旨在同步地理分布的 Ceph 集群之间的对象存储数据，提供高可用（HA）和灾难恢复（DR）能力。

## 目录



## 术语

| 术语 | 说明 |
| --- | --- |
| Primary Cluster | 当前提供存储服务的集群。 |
| Secondary Cluster | 用于备份的备用集群。 |
| Realm, ZoneGroup, Zone | • Realm：Ceph 对象存储中最高级别的逻辑分组。它代表一个完整的对象存储命名空间，通常用于多站点复制和同步。一个 Realm 可以跨越不同的地理位置或数据中心。<br>• ZoneGroup：Realm 内的逻辑分组，包含多个 Zone。ZoneGroup 实现跨 Zone 的数据同步和复制，通常在同一地理区域内。<br>• Zone：ZoneGroup 内的逻辑分组，物理存储数据。每个 Zone 独立管理和存储对象，并可以拥有自己的数据/元数据池配置。 |

# 前提条件

- 准备两个可部署 Rook-Ceph 的集群（Primary 和 Secondary 集群），且它们之间网络连通。
- 两个集群必须使用相同的平台版本（v3.12 或更高）。
- 确保 Primary 和 Secondary 集群均未部署 Ceph 对象存储。
- 参考创建存储服务文档部署 Operator 并创建集群。集群创建后不要通过向导创建对象存储池，需使用 CLI 工具按下述方式配置。

# 操作流程

本指南提供同一 ZoneGroup 内两个 Zone 之间的同步方案。

## 1 在 **Primary** 集群创建对象存储

本步骤创建 Realm、ZoneGroup、Primary Zone 及 Primary Zone 的网关资源。

在 Primary 集群的 Control 节点执行以下命令：

命令

```
cat << EOF | kubectl apply -f -
---
apiVersion: ceph.rook.io/v1
kind: CephObjectRealm
metadata:
  name: <realm-name>
  namespace: rook-ceph

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZoneGroup
metadata:
  name: <zonegroup-name>
  namespace: rook-ceph
spec:
  realm: <realm-name>

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZone
metadata:
  name: <primary-zone-name>
  namespace: rook-ceph
spec:
  zoneGroup: <zonegroup-name>
  metadataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
  dataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
    parameters:
      compression_mode: none
  preservePoolsOnDelete: false

---
cat << EOF | kubectl apply -f -
apiVersion: ceph.rook.io/v1
kind: CephObjectStore
```

```
metadata:
  name: <object-store-name>
  namespace: rook-ceph
spec:
  gateway:
    port: 7480
    instances: 2
  zone:
    name: <zone-name>
EOF
```

输出

```
cephobjectrealm.ceph.rook.io/<realm-name> created
cephobjectzonegroup.ceph.rook.io/<zonegroup-name> created
cephobjectzone.ceph.rook.io/<zone-name> created
cephobjectstore.ceph.rook.io/<object-store-name> created
```

参数：

- `<realm-name>`：Realm 名称。
- `<zonegroup-name>`：ZoneGroup 名称。
- `<primary-zone-name>`：Primary Zone 名称。
- `<object-store-name>`：网关名称。

## 2 配置 **Primary Zone** 的外部访问

1. 获取 ObjectStore 的 UID {#uid}

   ```
   kubectl -n rook-ceph get cephobjectstore <object-store-name> -o jsonpath='{.metadata.uid}'
   ```

   参数

   - `<object-store-name>`：在步骤 1 中配置的网关名称。

2. 创建外部访问 Service

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-rgw-<object-store-name>-external
  namespace: rook-ceph
  labels:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: <object-store-name>
  ownerReferences:
    - apiVersion: ceph.rook.io/v1
      kind: CephObjectStore
      name: <object-store-name>
      uid: <object-store-uid>
spec:
  ports:
    - name: rgw
      port: 7480
      targetPort: 7480
      protocol: TCP
  selector:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: <object-store-name>
  sessionAffinity: None
  type: NodePort
EOF
```

参数：

- `<object-store-name>`：在此处配置的网关名称。
- `<object-store-uid>`：在此处获取的 UID。

3. 向 CephObjectZone 添加外部端点。

```
kubectl -n rook-ceph patch cephobjectzone <primary-zone-name> --ty
pe merge -p '{"spec":{"customEndpoints":["<external-endpoint>"]}}'
```

参数：

- `<zone-name>`：在此处配置的 Primary Zone 名称。
- `<external-endpoint>`：从 Primary 集群获取的外部地址。

## 3 获取 `access-key` 和 `secret-key`

```
kubectl -n rook-ceph get secrets <realm-name>-keys -o yaml | grep acc
ess-key
kubectl -n rook-ceph get secrets <realm-name>-keys -o yaml | grep sec
ret-key
```

参数：

- `<realm-name>`：在此处配置的 Realm 名称。

## 4 创建 **Secondary Zone** 并配置 **Realm** 同步

本节说明如何创建 Secondary Zone 并通过从 Primary 集群拉取 Realm 信息配置同步。

在 Secondary 集群的 Control 节点执行以下命令：

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: <realm-name>-keys
  namespace: rook-ceph
data:
  access-key: <access-key>
  secret-key: <secret-key>

---
apiVersion: ceph.rook.io/v1
kind: CephObjectRealm
metadata:
  name: <realm-name>
  namespace: rook-ceph
spec:
  pull:
    endpoint: <realm-endpoint>

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZoneGroup
metadata:
  name: <zone-group-name>
  namespace: rook-ceph
spec:
  realm: <realm-name>

---
apiVersion: ceph.rook.io/v1
kind: CephObjectZone
metadata:
  name: <new-zone-name>
  namespace: rook-ceph
spec:
  zoneGroup: <zone-group-name>
  metadataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
  dataPool:
```

```
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
  preservePoolsOnDelete: false

---
apiVersion: ceph.rook.io/v1
kind: CephObjectStore
metadata:
  name: <secondary-object-store-name>
  namespace: rook-ceph
spec:
  gateway:
    port: 7480
    instances: 2
  zone:
    name: <secondary-zone-name>
EOF
```

参数：

- `<access-key>`：从此处获取的 AK。
- `<secret-key>`：从此处获取的 SK。
- `<realm-endpoint>`：从 Primary 集群获取的外部地址。
- `<realm-name>`：Realm。
- `<zone-group-name>`：ZoneGroup。
- `<secondary-zone-name>`：Secondary Zone 名称。
- `<secondary-object-store-name>`：Secondary 网关名称。

## 5 配置 Secondary Zone 的外部访问

1. 获取 Secondary 网关的 UID {#uids}

```
kubectl -n rook-ceph get cephobjectstore <secondary-object-store-name> -o jsonpath='{.metadata.uid}'
```

参数：

- `<secondary-object-store-name>`：Secondary 集群中网关名称。

2. 创建外部访问 Service

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-rgw-<object-store-name>-external
  namespace: rook-ceph
  labels:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: <object-store-name>
  ownerReferences:
    - apiVersion: ceph.rook.io/v1
      kind: CephObjectStore
      name: <object-store-name>
      uid: <object-store-uid>
spec:
  ports:
    - name: rgw
      port: 7480
      targetPort: 7480
      protocol: TCP
  selector:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: <object-store-name>
  sessionAffinity: None
  type: NodePort
EOF
```

参数：

- `<secondary-object-store-name>`：Secondary 网关。
- `<secondary-object-store-uid>`：Secondary 网关 UID。

3. 向 Secondary CephObjectZone 添加外部端点

```
kubectl -n rook-ceph patch cephobjectzone <secondary-zone-name> --
type merge -p '{"spec":{"customEndpoints":["<external-endpoint
>"]}}'
```

参数：

- `<secondary-zone-name>`：Secondary Zone 名称。
- `<secondary-zone-external-endpoint>`：从 Secondary 集群获取的外部地址。

# 故障切换

当 Primary 集群发生故障时，需要将 Secondary Zone 提升为 Primary Zone。切换后，Secondary Zone 的网关可继续提供对象存储服务。

## 操作流程

在 Secondary 集群的 `rook-ceph-tool` pod 中执行以下命令：

```
radosgw-admin zone modify --rgw-realm=<realm-name> --rgw-zonegroup=<zone-
group-name> --rgw-zone=<secondary-zone-name> --master
```

参数

- `<realm-name>`：Realm 名称。
- `<zone-group-name>`：Zone Group 名称。
- `<secondary-zone-name>`：Secondary Zone 名称。

# 相关操作

## 获取外部地址

1. 进入 平台管理。

2. 在左侧导航栏点击 存储管理 > 分布式存储。

3. 在 集群信息 标签页，向下滚动至 存储池 区域，点击对象存储池旁的⋮，选择 查看地址。

# 更新优化参数

平台支持在创建存储集群时以 Ceph 配置文件格式填写优化参数，但创建后未提供界面修改方式，您需要根据以下步骤手动更新。

## 目录



## 操作步骤

1. 首先，将存储优化参数更新至名为 `rook-config-override-user` 的 ConfigMap 中，替换 `.data.config` 字段，并将 `.metadata.annotations[rook.cpaas.io/need-sync]` 字段的值设置为 `true`。举例如下：

```
apiVersion: v1
data:
  config: |
    [global]
    mon_memory_target=1073741824
    mds_cache_memory_limit=2147483648
    osd_memory_target=4147483648
kind: ConfigMap
metadata:
  annotations:
    cpaas.io/creator: admin
    cpaas.io/updated-at: "2022-03-01T12:24:04Z"
    rook.cpaas.io/need-sync: "true"
    rook.cpaas.io/sync-status: synced
  creationTimestamp: "2022-03-01T12:24:04Z"
  finalizers:
  - rook.cpaas.io/config-merge
  name: rook-config-override-user
  namespace: default
  resourceVersion: "38816864"
  uid: ce3a8f3e-6453-4bdd-bff0-e16cf7d5d5fa
```

2. 在 rook-ceph-tools 的 Pod 中执行 `ceph tell [mon|osd|mgr|mds|rgw].* config set [key] [value]` 以实时应用配置。

3. 若要启动工具的 Pod，需要编辑 rook-ceph 命名空间下的 ClusterServiceVersion (CSV)，将 Deployments 部分的 rook-ceph-tools 的 replicas 值设置为 1。

# 权限说明

| 功能 | 操作 | 平台管理员 | 平台审计人员 | 项目管理员 | 命名空间管理员 | 开发人员 |
| --- | --- | --- | --- | --- | --- | --- |
| 内置存储 `acp-builtinstorage` | 查看 | ✓ | ✓ | ✓ | ✓ | ✓ |
| | 创建 | ✓ | × | × | × | × |
| | 更新 | ✓ | × | × | × | × |
| | 删除 | ✓ | × | × | × | × |

# MinIO 对象存储

## 介绍

介绍

## 安装

安装

前提条件

部署 Operator

创建集群

创建 Bucket

上传/下载文件

相关信息

## 架构

架构

核心组件：

部署架构：

多池扩展：

结论：

## 核心概念

核心概念

## 操作指南

添加存储池

注意事项

操作步骤

**Monitoring & Alerts**

**Monitoring**

**Alerts**

## 实用指南

数据灾难恢复

适用场景

名词解释

前提条件

操作步骤

相关操作

# 介绍

灵雀云容器平台 (ACP) 的对象存储服务以 MinIO 为基础，采用 Apache License v2.0 许可。它兼容亚马逊 S3 云存储服务接口，非常适合存储大量非结构化数据，如图片、视频、日志文件、备份数据以及容器/虚拟机镜像。对象文件的大小可以从几 KB 到最大 5T。

主要优势如下：

- 简单：极简主义是 MinIO 的指导性设计原则，开箱即可使用。简单性减少了出错的机会，提高了正常运行时间，同时增强了可靠性并提升了性能。
- 高性能：MinIO 是全球领先的对象存储。在标准硬件上，读/写速度可高达 183 GB/秒和 171 GB/秒。
- 可扩展：可以建立多个小型到中型、易于管理的集群，支持将多个集群聚合成一个超大资源池，跨数据中心展开，而不是直接采用大规模、统一管理的分布式集群。
- 云原生：符合所有原生云计算的架构和构建过程，并结合云计算中的最新技术和概念，使对象存储对于 Kubernetes 更加友好。

# 安装

Alauda Container Platform (ACP) 基于 MinIO 的对象存储是一种基于 Apache License v2.0 开源协议的对象存储服务。它兼容 Amazon S3 云存储服务接口，适合存储大量非结构化数据，如图片、视频、日志文件、备份数据以及容器/虚拟机镜像。一个对象文件的大小可以是任意的，从几千字节到最大 5 TB。

## 目录



## 前提条件

MinIO 构建于底层存储之上，请确保当前集群已创建存储类。推荐使用 TopoLVM。

# 部署 **Operator**

1. 在左侧导航栏，点击 **Storage** > **Object Storage**。

2. 点击 **Configure Now**。

3. 在 **Deploy MinIO Operator** 向导页面，点击右下角的 **Deploy Operator**。

   - 页面自动跳转到下一步时，表示 Operator 部署成功。
   - 若部署失败，请根据界面提示选择 **Clean Up Deployed Information and Retry**，然后重新部署 Operator。

# 创建集群

1. 在 **Create Cluster** 向导页面，配置基本信息。

| 参数 | 说明 |
| --- | --- |
| **Access Key** | 访问密钥 ID。与私有访问密钥关联的唯一标识符；用于与访问密钥 ID 一起对请求进行加密和签名。 |
| **Secret Key** | 与访问密钥 ID 配合使用的私有访问密钥，用于加密和签名请求，识别发送方，防止请求篡改。 |

2. 在 **Resource Configuration** 区域，根据以下说明配置规格。

| 参数 | 说明 |
| --- | --- |
| **Small scale** | 适用于处理最多 100,000 个对象，支持测试环境或数据备份场景中不超过 50 个并发访问。CPU 资源请求和限制默认设置为 2 核，内存资源请求和限制默认设置为 4 Gi。 |
| **Medium scale** | 适用于企业级应用，需存储 1,000,000 个对象，支持最多 200 个并发请求。CPU 资源请求和限制默认设置为 4 核，内存资源请求和限制默认设置为 8 Gi。 |

| 参数 | 说明 |
| --- | --- |
| **Large scale** | 适用于存储需求为 10,000,000 个对象，支持最多 500 个并发请求的集团用户，适合高负载场景。CPU 资源请求和限制默认设置为 8 核，内存资源请求和限制默认设置为 16 Gi。 |
| **Custom** | 为有特定需求的专业用户提供灵活配置选项，确保服务规模和性能需求的精准匹配。注意：配置自定义规格时，请确保： |

- CPU 资源请求大于 100 m。
- 内存资源请求大于或等于 2 Gi。
- CPU 和内存资源限制大于或等于资源请求。 |

3. 在 **Storage Pool** 区域，根据以下说明配置相关信息。

| 参数 | 说明 |
| --- | --- |
| **Instance Number** | 增加 MinIO 集群中的实例数量可以显著提升系统性能和可靠性，确保数据高可用。但实例过多可能导致以下问题： |

- 资源消耗增加。
- 如果一个节点承载多个实例，节点故障可能导致多个实例同时离线，降低集群整体可靠性。 注意：
- 最小可输入实例数为 4。
- 实例数大于 16 时，输入值必须是 8 的倍数。
- 添加额外存储池时，实例数不得少于第一个存储池的实例数。 | | **Single Storage Volume** | 单个存储卷 PVC 的容量。每个存储服务管理一个存储卷。输入单个存储卷容量后，平台会自动计算存储池容量及其他信息，可在 **Storage Pool Overview** 中查看。 | | **Underlying Storage** | MinIO 集群使用的底层存储。请选择当前集群中已创建的存储类。推荐使用 TopoLVM。 | | **Storage Nodes** | 选择 MinIO 集群所需的存储节点，建议使用 4-16 个存储节点。平台会为每个选中的存储节点部署一个存储服务。 | | **Storage Pool Overview** | 具体参数及计算公式，请参见 Storage Pool Overview。 |

4. 在 **Access Configuration** 区域，根据以下说明配置相关信息。

| 参数 | 说明 |
| --- | --- |
| **External Access** | 启用时支持跨集群访问 MinIO；禁用时仅支持集群内访问。 |
| **Protocol** | 支持 HTTP 和 HTTPS；选择 HTTPS 时需填写 **Domain** 并导入域名证书的 **Public Key** 和 **Private Key**。 |
| **Note**: | |

- 访问协议为 HTTP 时，集群内 pod 可通过获取的 IP 或域名直接访问 MinIO，无需配置 IP 与域名映射；集群内节点可通过获取的 IP 直接访问 MinIO，若需域名访问，则需手动配置 IP 与域名映射；外部访问可直接通过获取的 IP 访问。
- 访问协议为 HTTPS 时，集群内外均无法通过 IP 访问 MinIO。需手动配置获取的 IP 与集群创建时填写的域名映射，才能通过域名正常访问。 | | **Access Method** |
- **NodePort**：在每个计算节点主机上开启固定端口，将服务暴露到外部。配置域名访问时，建议使用 VIP 进行域名解析以保证高可用。
- **LoadBalancer**：通过负载均衡器转发流量到后端服务。使用前请确保当前集群已部署 MetalLB 插件且外部地址池有可用 IP。 |

5. 点击右下角 **Create Cluster**。

- 页面自动跳转至 **Cluster Details**，表示集群创建成功。
- 若集群仍处于创建中，可点击 **Cancel**。取消后会清理已部署的集群信息，可返回集群创建页面重新创建。

# 创建 **Bucket**

登录集群控制节点，使用命令创建 bucket。

## 操作步骤

1. 在集群详情页，点击 **Access Method** 标签查看 MinIO 访问地址，或使用以下命令查询。

```
kubectl get svc -n <tenant ns> minio | grep -w minio | awk '{print $3}'
```

注意：

- 将 `tenant ns` 替换为实际命名空间 `minio-system`。
- 示例：`kubectl get svc -n minio-system minio | grep -w minio | awk '{print $3}'`

2. 获取 mc 命令。

```
wget https://dl.min.io/client/mc/release/linux-amd64/mc -O /bin/mc && chmod a+x /bin/mc
```

3. 配置 MinIO 集群别名。

- IPv4：

```
mc --insecure alias set <minio cluster alias> http://<minio endpoint>:<port> <accessKey> <secretKey>
```

- IPv6：

```
mc --insecure alias set <minio cluster alias> http://[<minio endpoint>]:<port> <accessKey> <secretKey>
```

- 域名：

```
mc --insecure alias set <minio cluster alias> http://<domain name>:<port> <accessKey> <secretKey>
mc --insecure alias set <minio cluster alias> https://<domain name>:<port> <accessKey> <secretKey>
```

注意：

- `minio endpoint` 填写步骤 1 中获取的 IP 地址。

- `accessKey` 和 `secretKey` 填写集群创建时配置的 **Access Key** 和 **Secret Key**。
- 配置示例：
  - IPv4: `mc --insecure alias set myminio http://12.4.121.250:80 07Apples@ 07Apples@`
  - IPv6: `mc --insecure alias set myminio http://[2004::192:168:143:117]:80 07Apples@ 07Apples@`
  - 域名: `mc --insecure alias set myminio http://test.minio.alauda:80 07Apples@ 07Apples@` 或 `mc --insecure alias set myminio https://test.minio.alauda:443 07Apples@ 07Apples@`

4. 创建 bucket。

```
mc --insecure mb <minio cluster alias>/<bucket name>
```

# 上传/下载文件

创建 bucket 后，可使用命令行上传文件到 bucket，或从 bucket 下载已有文件。

## 操作步骤

1. 创建用于上传测试的文件。若上传已有文件，此步骤可跳过。

```
touch <file name>
```

2. 上传文件到 bucket。

```
mc --insecure cp <file name> <minio cluster alias>/<bucket name>
```

3. 查看 bucket 中的文件，确认上传成功。

```
mc --insecure ls <minio cluster alias>/<bucket name>
```

4. 删除上传的文件。

```
mc --insecure rm <minio cluster alias>/<bucket name>/<file name>
```

# 相关信息

## 冗余因子映射表

注意：添加额外存储池时，冗余因子需根据第一个存储池的实例数计算。

| 实例数 | 冗余因子 |
| --- | --- |
| **4 - 5** | 2 |
| **6 - 7** | 3 |
| **>= 8** | 4 |

## Storage Pool Overview

| **Storage Pool Overview** 参数 | 计算公式 |
| --- | --- |
| 可用容量 | 当实例数 ≤ 16 时，可用容量 = 单个存储卷容量 × (实例数 - 冗余因子)。 |
| 当实例数 > 16 时，可用容量 = 单个存储卷容量 × (实例数 - 4 × (实例数 + 15) / 16)。其中“4 × (实例数 + 15) / 16”结果向下取整。 | |
| 总容量 | 总容量 = 实例数 × 单个存储卷容量 |
| 可容忍故障存储服务数量 | 当实例数 > 2 × 冗余因子时，可容忍故障存储服务数量 = 冗余因子。 |

| **Storage Pool Overview** 参数 | 计算公式 |
| --- | --- |
| 当实例数 = 2 × 冗余因子时，可容忍故障存储服务数量 = 冗余因子 - 1 | |

# 架构

Alauda Container Platform (ACP) Object Storage with MinIO 是一个高性能的分布式对象存储系统，专为云原生环境设计。它利用纠删码、分布式存储池和高可用机制，确保 Kubernetes 中的数据持久性和可扩展性。

## 目录



## 核心组件：

- **MinIO Operator**：管理 MinIO 集群的部署和升级。
- **MinIO Peer**：配置和管理 MinIO 的站点复制功能。
- **MinIO Pool**：MinIO 的核心组件，负责处理对象存储请求。每个池对应一个 StatefulSet，提供存储资源。

## 部署架构：

在 Kubernetes 中部署 MinIO 需要定义一个 MinIO tenant，指定服务器实例（pod）的数量以及每个实例的卷（驱动器）数量。每个 MinIO 服务器通过 StatefulSet 管理，确保稳定的身份和持

久存储。MinIO 将所有驱动器聚合成一个或多个纠删集，并应用纠删码以实现容错。

## 多池扩展：

MinIO 集群可以通过添加额外的服务器池来扩展，每个池拥有自己的纠删集。虽然这提供了更大的存储容量，但也增加了集群维护的复杂性并降低了整体集群的可靠性。任何服务器池的故障都可能导致整个 MinIO 集群不可用，即使其他池仍在运行。

## 结论：

MinIO 是一个高度可扩展的云原生对象存储解决方案，兼顾性能和可靠性。在设计 MinIO 集群架构时，必须仔细规划存储池、配置纠删码设置，并实施高可用策略，以确保 Kubernetes 环境中的数据完整性和运行稳定性。

# 核心概念

# 核心概念

- 纠删码（**Erasure Coding**）：MinIO 使用 Reed-Solomon 纠删码，将对象分割为数据块和奇偶校验块，分布在多个驱动器上，以确保容错性。例如，在 16 驱动器设置中，数据可以被分割为 12 个数据块和 4 个奇偶校验块，即使最多 4 个驱动器发生故障，系统仍能够重建数据。

- 存储池和纠删集：MinIO 存储池是逻辑上对存储资源的分组，每个池由多个节点组成，共享存储和计算能力。在一个存储池内，驱动器会被自动组织成一个或多个 纠删集。

  - 数据分布：当一个对象被存储时，它会被分割为数据分片和奇偶校验分片，并分布到同一个纠删集内的不同驱动器上。
  - 冗余模型：纠删集构成数据冗余的基本单元，基于配置的数据信息与奇偶校验分片的比例确保系统的健壮性。
  - 可扩展性：一个 MinIO 存储池可以包含多个纠删集，新写入的数据总是被存储到具有最多可用容量的纠删集中。

# 操作指南

# 添加存储池

存储池指用于存储数据的逻辑分区。同一存储集群中支持同时使用不同类型的底层存储，以满足不同的业务需求。

除了在对象存储配置时创建的存储池外，您还可以添加额外的存储池。

## 目录



## 注意事项

添加存储池将会导致 MinIO 服务短暂中断，但随后将自动恢复至正常状态。

## 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 > 对象存储。
3. 在 集群信息 页签下，滚动至 存储池 区域并单击 添加存储池。
4. 根据以下说明配置相关参数。

| 参数 | 说明 |
| --- | --- |
| 底层存储 | MinIO 集群使用的底层存储。请选择当前集群中已创建的存储类，推荐使用 TopoLVM。 |
| 存储节点 | 选择 MinIO 集群所需的存储节点。建议使用 4-16 个存储节点；平台会为每个选择的存储节点部署 1 个存储服务。<br>说明：当使用 3 个存储节点时，为确保可靠性，将为每个节点部署 2 个存储服务。 |
| 单个存储卷 | 单个存储卷 PVC 的容量。每个存储服务管理 1 个存储卷，一旦输入单个存储卷的容量，平台将自动计算存储池的容量及其他信息，这些信息可以在 存储池概览 中查看。 |

5. 单击 确定。

# Monitoring & Alerts

对象存储系统内置了监控和告警功能，涵盖存储集群、服务健康状况和资源利用率。同时支持可配置的通知策略，确保运维团队及时获知。实时监控洞察有助于性能调优和运维决策，自动告警则保障存储系统的稳定性和可靠性。

## 目录



# Monitoring

平台默认采集存储集群和服务状态的关键指标。您可以在 **Storage Management** > **Object Storage** > **Monitoring** 中查看实时监控数据。

## Storage Overview

本节提供存储系统健康状况、服务状态及原始容量利用率的高层视图。如存储状态异常，告警详情将指明根因，帮助您高效诊断和解决问题。

## Cluster Monitoring

跟踪存储集群的原始容量使用情况和 I/O 性能趋势，有助于识别存储瓶颈、优化资源分配，确保数据操作顺畅。

## Object Monitoring

监控访问模式，包括总请求数和失败请求数。这些洞察有助于分析存储负载，检测可能导致服务中断或安全风险的异常情况。

# Alerts

平台内置告警策略，用于检测异常并在达到预设阈值时触发通知。内置规则覆盖组件健康、容量使用和用户数据完整性等关键领域。

## Configuring Notifications

为确保及时响应，请在 **Operations Center** 配置通知策略。告警可通过邮件、短信或其他渠道发送，通知相关人员。根据组织的事件响应流程，细化设置以匹配需求。

## Handling Alerts

- **Cluster in "Alert" state**：已触发警告，系统稳定性可能受影响。请查看 **Live Alerts** 部分获取详情，定位根因并采取纠正措施。
- **Cluster in "Failure" state**：存储集群已无法正常运行。需立即干预以恢复服务可用性。

平台将告警分为不同严重级别，帮助团队优先处理事件：

| Severity | Description |
| --- | --- |
| **Critical** | 影响业务运营或导致数据丢失的系统故障，需立即处理。 |

| Severity | Description |
| --- | --- |
| **Major** | 可能导致功能中断的已知问题，可能影响业务流程。 |
| **Warning** | 潜在风险，若不处理可能影响性能或可用性。 |

## Post-Incident Analysis

**Alert History** 记录所有历史事件，为事后分析和系统改进提供宝贵数据。回顾过去告警时，请考虑：

1. 事件发生时的具体症状是什么？
2. 是否有告警重复出现？是否能采取主动措施防止复发？
3. 是否存在某个时间段告警激增？是由运维问题还是外部因素引起？是否需要调整响应策略？

通过持续分析告警模式和优化监控策略，团队可提升系统韧性，减少停机时间，确保存储业务的顺畅运行。

# 实用指南

## 数据灾难恢复

# 数据灾难恢复

MinIO 支持通过远程数据备份或双活部署的方式建立灾难恢复中心，以确保灾难发生时原有数据不会丢失或遭到损坏，从而保证数据的安全性和可靠性。

## 目录



## 适用场景

- 热备：在同城市或不同地点存在两个数据中心，一个为主中心，一个为备中心。数据实时从主集群复制到备集群，以确保数据一致性。当主集群发生灾难时，业务流量可以无缝切换到备集群，以确保业务的连续性。
- 市级双活：在市级双活（多集群）架构中，存在两个位于不同集群的数据中心。两个数据中心均处于活动状态，能够同时接收业务流量。当其中一个数据中心遭遇灾难时，业务可以在另一个数据中心不间断地继续运行。

## 名词解释

- 主集群：指当前处于活动状态且正在处理业务请求的集群。它是数据的源头或操作的发起者。在主集群中，数据被创建、修改或更新，业务流量首先发送到这个集群进行处理。
- 目标集群：指接收数据复制、迁移或故障转移的集群。它通常处于备份或待命状态，准备接收来自主集群的数据或接管业务流量。当主集群发生故障或需要切换时，目标集群将接收来自主集群的数据副本或接管业务流量，以确保业务连续性。在双活场景中，两集群可以相互作为对方的目标集群。

# 前提条件

- 主集群和目标集群必须启用外网访问。具体配置方法请参考 创建对象存储。
- 主集群必须使用 **LoadBalancer** 访问方式，而目标集群建议支持 负载均衡 功能。
- 主集群和目标集群必须使用相同的访问协议，即均使用 HTTP 或均使用 HTTPS。
- 使用 HTTPS 协议时，主集群和目标集群需要分别配置自身及对方的 DNS 解析。
- 使用 HTTPS 协议时，建议主集群和目标集群使用 CA 签发的证书，以确保通信安全和信任；如果使用自签名证书，则需要双方相互导入并信任对方的自签名证书，以便成功建立安全的 HTTPS 连接。

# 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 > 对象存储。
3. 在 数据灾难恢复 页签下，单击 添加目标集群。
4. 根据以下说明配置目标集群的相关参数。

| 参数 | 说明 |
| --- | --- |
| 访问地址 | 目标集群的外网访问地址，以 http:// 或 https:// 开头。 |
| | |

| 参数 | 说明 |
| --- | --- |
| **Access Key** | 目标集群的访问密钥 ID。与私有访问密钥关联的唯一标识符；与私有访问密钥一起使用，对请求进行加密。 |
| **Secret Key** | 与访问密钥 ID 结合使用的私有访问密钥，用于加密请求、标识发送方并防止请求被修改。 |

5. 单击 添加。

- 添加成功后，将能够查看目标集群的状态以及集群间的同步状态。

| 参数 | 说明 |
| --- | --- |
| 集群状态 | 目标集群的状态，包括 健康、异常 或 未知。 |
| 存储桶 | 待同步和已同步的存储桶数量。<br>• 在热备场景中，待同步的数量是主集群需要同步到目标集群的存储桶数量。<br>• 在市级双活场景中，待同步的数量是主集群和目标集群之间需要同步的存储桶总数。 |
| 对象 | 存储桶中同步失败的对象数量。<br>说明：此数量仅供参考，因 MinIO 在同步过程中会同步相关的文件配置。 |
| 网络流量速率 | 主集群的网络入口和出口速率。<br>• 在热备场景中，网络入口速率始终为 0。<br>• 在市级双活场景中，入口和出口速率均有数据。 |

- 如果目标集群添加失败，可以单击 重新添加 清除集群信息并返回添加目标集群页面，以便重新添加目标集群。

# 相关操作

当不再需要数据灾难恢复时，可以单击 移除目标集群。移除目标集群不会删除已同步的数据；如果有数据正在同步，将会被中断。

# TopoLVM 本地存储

## 介绍

介绍

## 安装

安装

前提条件

操作步骤

## 操作指南

设备管理

前提条件

添加设备

监控与告警

监控功能

告警功能

# 介绍

TopoLVM 是一个专为 Kubernetes 设计的容器存储接口（CSI）插件，致力于提供高效、便捷的本地存储卷管理解决方案。

主要特点和优势：

- 本地卷管理：TopoLVM 专注于对 Kubernetes 节点上的本地存储设备（如磁盘、SSD）进行管理。与传统的网络存储相比，本地存储卷具有更低的延迟和更高的性能。
- 拓扑感知能力：TopoLVM 能够识别 Kubernetes 集群的拓扑结构（例如节点、可用区），使存储卷能够根据 Pod 实际的调度位置自动分配到同一节点，从而进一步优化性能。
- 动态卷分配：TopoLVM 支持动态创建、删除以及调整存储卷的容量，无需人工干预，显著简化了操作，降低了运维复杂性。
- 与 **Kubernetes** 深度集成：作为 CSI 插件，TopoLVM 无缝衔接 Kubernetes 的存储管理 API。用户可以通过标准的 Kubernetes 资源对象（如 PersistentVolumeClaims）直接进行本地卷的管理。

简而言之，TopoLVM 致力于解决在 Kubernetes 集群中使用本地存储时经常遇到的手动管理、缺乏拓扑感知和动态分配能力不足等挑战，为数据库、缓存等需要高性能本地存储的应用场景提供更加高效和易用的解决方案。

# 安装

本地存储是一种软件定义的服务器本地存储解决方案，提供简单、易于维护和高性能的本地存储服务能力。基于社区的 TopoLVM 方案，通过系统的 LVM 方法实现本地存储的持久卷编排管理。

## 目录



## 前提条件

存储集群的每个节点上必须安装 lvm2 软件包。如果未安装，请在节点上执行 `yum install -y lvm2` 命令。

## 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 > 本地存储。
3. 单击 立即配置。
4. 在 安装 **Operator** 向导页中，单击 开始部署。

- 待页面自动进入下一步时，说明 Operator 部署成功。
- 如果部署失败，请参考界面提示进行处理，然后单击 清理 重新部署 Operator。

5. 在 创建集群 向导页中，添加设备。

| 参数 | 说明 |
| --- | --- |
| 选择节点 | 至少具备 1 块裸盘的节点。 |
| 设备类 | 每个设备类对应一组相同特性的存储设备，建议根据磁盘性质填写名称，例如 *hdd*、*ssd*。 |
| 设备类型 | 仅支持磁盘类型。 |
| 存储设备 | 例如 */dev/sda*。如有多个盘，可逐个添加。 |
| 快照 | 开启后，支持创建 PVC 快照并使用快照配置新的 PVC，以快速备份和恢复业务数据。<br>若创建存储时未开启快照，您仍可以在存储集群详情页面的 操作 部分按需开启。<br>注意：使用前请确保已为当前集群 部署卷快照插件。 |

- 单击下一步，待页面自动进入下一步时，说明集群部署成功。
- 如果创建失败，请参考界面提示并及时清理资源。

6. 在 创建存储类 向导页中，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。 |
| 显示名称 | 可帮助您进行识别或筛选的名称，例如存储类的中文描述。 |

| 参数 | 说明 |
| --- | --- |
| 设备类 | 设备类是 TopoLVM 中用于对存储设备进行分类的方式，每个设备类对应一组相同特性的存储设备。如无特殊要求，可使用集群中的 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，擅长处理并行 I/O 工作负载，支持大文件处理，提供平滑的数据传输。<br>- **EXT4** 是 Linux 下的一种日志文件系统，提供扩展文件存储方法，支持大文件处理。文件系统容量可达 1 EiB，支持的最大文件大小可达 16 TiB。 |
| 回收策略 | 持久卷的回收策略。<br>- 删除：当持久卷声明被删除时，绑定的持久卷也会被删除。<br>- 保留：即使持久卷声明被删除，绑定的持久卷仍会保留。 |
| 访问模式 | 单节点读写 (RWO)：可以被一个节点以读写方式挂载。 |
| 分配项目 | 仅可在特定项目中创建此类型的持久卷声明。<br>如果暂时未分配项目，也可后续 更新项目。 |

7. 单击 下一步，等待资源创建完成。

# 操作指南

## 设备管理

- 前提条件
- 添加设备

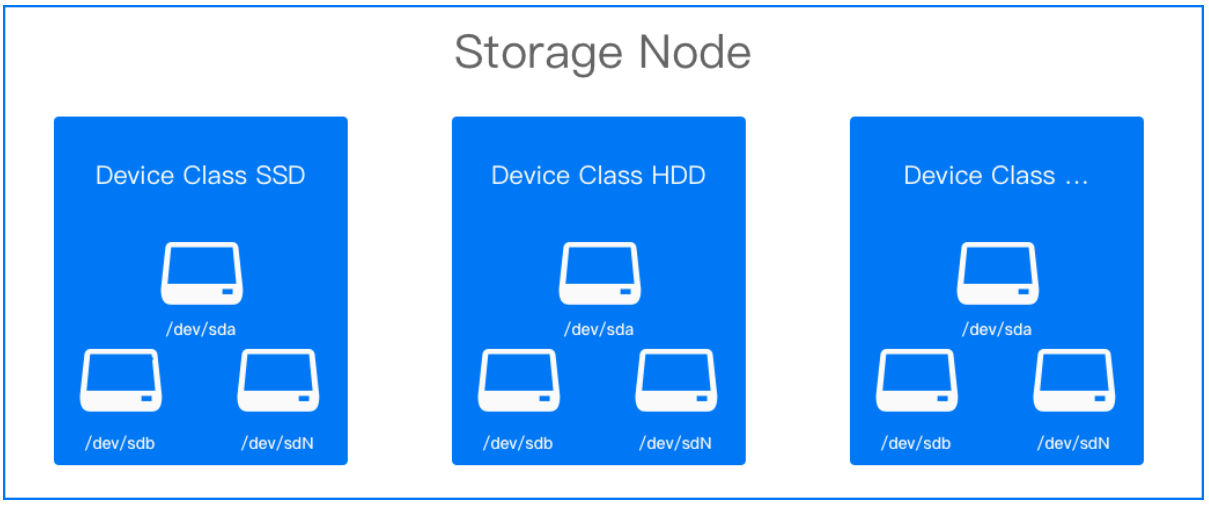
## 监控与告警

- 监控功能
- 告警功能

# 设备管理

无论是初始部署还是资源扩容，都需要将节点上可用的磁盘映射为存储设备以供使用和管理。

具有相似特性的存储设备通常集中使用，这些设备在本地存储中归类于设备类（**Device Classes**）。使用设备类相当于直接使用磁盘，确保零损耗和高性能，同时减少应用对特定设备的感知和依赖。




## 目录



## 前提条件

- 创建本地存储集群时，必须至少添加 1 个设备类（deviceClasses.classes），且设备类中包含设备。

- 节点上必须至少存在 1 个裸盘。

# 添加设备

1. 进入 平台管理。

2. 在左侧导航栏点击 存储管理 > 本地存储。

3. 在 详情 标签页，点击 添加存储节点。

4. 按照以下说明配置相关参数。

| 参数 | 说明 |
| --- | --- |
| 存储节点 | 至少有 1 个裸盘的节点。 |
| 设备类 | 每个设备类对应一组具有相同特性的存储设备；建议根据磁盘性质命名，如 *hdd*、*ssd*。 |
| 存储设备 | 例如 */dev/sda*。如果有多个磁盘，可以逐个添加。<br><br>注意：存储设备应为整个硬盘，而非硬盘上的分区，否则会导致错误。 |

5. 点击 添加。

   注意：如果设备类状态因未添加设备而显示为 `Unavailable`，可以继续执行以下操作。

6. 切换到 存储设备 标签页，点击 添加存储设备。

7. 根据界面提示添加设备。

8. 点击 添加。

# 监控与告警

本地存储提供开箱即用的监控指标收集和告警能力。一旦平台监控组件被启用，用户即可针对存储集群、存储性能和存储容量配置监控和告警功能，并支持配置通知策略。

平台提供直观的监控数据，能够为运维检查和性能优化提供决策支持。同时，完善的告警机制有助于确保存储系统的稳定运行。

## 目录



# 监控功能

## 性能监控

平台默认收集本地存储的常见性能指标，包括读写带宽、IOPS和延迟等信息。用户可在存储管理下的本地存储页面中的监控标签页实时查看这些性能数据。平台以图表方式直观展示指标，便于管理员实时了解当前存储性能状况，并快速定位潜在问题。

## 容量监控

由于本地存储只能使用节点本地的可用存储资源，因此用户在声明本地存储前需要确保节点具有足够的可用容量，以避免因超量声明导致的问题。

为此，平台在本地存储的详情部分提供了按设备类型分类的容量监控，用户可直观地查看不同设备类型的剩余可用空间。当发现某一设备类型容量不足时，应及时清理空间或添加新的磁盘设备后再使用本地存储。

# 告警功能

平台内置了一套默认的告警策略。当资源出现异常或监控数据达到警告阈值时，系统会自动触发告警。这些预置的告警策略能覆盖常见的运维需求，如集群状态告警和设备容量告警。

## 配置通知

为确保及时接收到告警信息，建议用户在运维中心配置通知策略。通知可通过邮件、短信等方式发送给相关人员，以提醒及时处理问题或预防故障。用户可以直接在运维中心界面进行通知策略设置，具体配置步骤可参考[创建告警策略]文档。

## 告警处理

- 若存储集群的健康状态变为“告警”，管理员需立即进行排查。在本地存储的详情部分提供了问题排查和解决指导。常见的告警原因包括节点服务异常或设备类型问题。

| 检查项 | 对应状态 | 可能原因 |
| --- | --- | --- |
| 健康状态 | 告警 | 节点服务异常或设备类型出现问题 |
| 服务状态 | 未知 | 节点处于 `notready` 状态，可能由网络故障或断电引起 |
| 设备类型状态 | 不可用 | 使用的磁盘可能不是裸盘，或者磁盘缺失 |

- 若在告警页面触发了实时告警，即使当前存储集群状态显示“健康”，也应迅速响应，防止问题恶化。告警等级及其含义如下：

| 告警等级 | 含义 |
| --- | --- |
| 紧急 | 告警关联的资源出现严重故障，导致服务中断或数据丢失，影响重大 |
| 重要 | 告警关联的资源存在已知问题，可能影响平台功能和业务正常运行 |
| 警告 | 告警关联的资源存在潜在风险，若不及时处理可能影响业务运行 |

## 事后分析

告警历史记录了过去曾触发但当前已无需处理的所有告警。在进行事后分析时，应重点考虑以下问题：

- 当时发生的具体异常情况是什么？
- 告警历史中是否存在重复出现的告警模式？如何预防类似情况再次发生？
- 特定时间段内告警次数的激增是否与外部因素或操作事件有关？是否需要相应调整运维策略？