Menu

存储

Ceph 分布式存储

介绍

功能概览

存储方案对比

安装

架构

技术架构

核心概念

操作指南

实用指南

MinIO 对象存储

介绍

安装

前提条件

操作步骤

相关信息

架构

核心组件：

部署架构：

多池扩展：

结论：

核心概念

操作指南

实用指南

TopoLVM 本地存储

介绍

安装

前提条件

操作步骤

操作指南

实用指南

Menu

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 **Stretch** 类型集群

术语

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

访问存储服务

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

节点特定组件部署

更新组件部署配置

重启存储组件

添加设备/设备类

添加设备类

添加设备

硬盘状态

监控与告警

监控

告警

实用指南

配置专用集群用于分布式存储

架构

基础设施要求

操作步骤

后续操作

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

创建 **ceph** 对象存储用户

前提条件

操作步骤

介绍

Alauda Build of Rook-Ceph 是平台内集群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

- 易于部署：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、性能和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可靠；简便可靠的自动化管理支持存储资源的在线扩容。
- 性能卓越：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，可在业务集群上完成配置，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 存储资源作为分布式存储接入。

需求（任选其一）	优势
方案一：分布式存储已部署于其他业务集群。	可充分利用跨集群存储资源，避免业务变更干扰。确保存储数据安全稳定，降低运维复杂度。 注意：若接入的存储为不同平台的分布式存储，如灾备环境的主备平台，请采用外部 Ceph 集成方式。
方案二：平台外部 **Ceph** 存储，版本 **≥ 14.2.3**。	相较于直接创建存储类，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进行操作。

Menu

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 **Stretch** 类型集群

术语

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 存储最典型的部署方式。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

准备软件包

- 下载对应您平台架构的 **Alauda Container Platform Storage Essentials** 安装包。
- 通过上传软件包机制上传 **Alauda Container Platform Storage Essentials** 安装包。
- 下载对应您平台架构的 **Alauda Build of Rook-Ceph** 安装包。
- 通过上传软件包机制上传 **Alauda Build of Rook-Ceph** 安装包。

准备基础设施

- 存储集群至少需要 3 个节点。
- 每个节点必须至少有 1 块空白硬盘或 1 个未格式化的硬盘分区可用。
- 建议可用硬盘容量大于 50 G。
- 如果您使用的是以 Containerd 作为运行时组件的附加 Kubernetes 集群，请确保集群所有节点的 `/etc/systemd/system/containerd.service` 文件中的 `LimitNOFILE` 参数值配置为 `1048576`，以保证分布式存储部署成功。配置说明请参考修改 Containerd 配置信息。

  注意：从 v3.10.2 之前的版本升级到当前版本时，如果需要在自定义 Kubernetes 集群中部署以 Containerd 作为运行时组件的 Ceph 分布式存储，也必须将集群所有节点的 `/etc/systemd/system/containerd.service` 文件中的 `LimitNOFILE` 参数值设置为 `1048576`。

注意事项

创建存储服务 和 访问存储服务 仅支持选择一种方式。

操作步骤

1 部署 Alauda Container Platform Storage Essentials

1. 登录，进入 **Administrator** 页面。
2. 点击 **Marketplace** > **OperatorHub**，进入 **OperatorHub** 页面。

3. 找到 **Alauda Container Platform Storage Essentials**，点击 **Install**，进入 **Install Alauda Container Platform Storage Essentials** 页面。

   配置参数：

参数	推荐配置
Channel	默认通道为 `stable`。
安装模式	`Cluster`：集群内所有命名空间共享单个 Operator 实例进行创建和管理，资源占用较低。
安装位置	选择 `Recommended`，命名空间仅支持 **acp-storage**。
升级策略	`Manual`：Operator Hub 有新版本时，需要手动确认升级 Operator 到最新版本。

2 部署 Operator

1. 进入 **Administrator**。
2. 在左侧边栏点击 **Storage Management** > **Distributed Storage**。
3. 点击 **Configure Now**。
4. 在 **Deploy Operator** 向导页面，点击右下角 **Deploy Operator** 按钮。
   - 页面自动跳转下一步表示 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**，重新部署 Operator；若需返回分布式存储选择页面，点击 **Application Store**，先卸载已部署的 **rook-operator** 资源，再卸载 **rook-operator**。

3 创建集群

1. 在 **Create Cluster** 向导页面，配置相关参数，点击右下角 **Create Cluster** 按钮。

参数	说明
Cluster Type	选择 **Standard**。
Device Class Type	设备类是硬盘的分组；可根据存储需求自定义设备类，将不同性能的硬盘分配存储不同内容。 • 默认设备类：平台自动对集群节点中的硬盘类型进行分类，如创建名为 `hdd`、`ssd`、`nvme` 的设备类。 • 自定义设备类：自定义节点中特定组合硬盘的设备类名称，支持添加多个设备类。同一块硬盘只能属于一个设备类。
Device Class - Name	设备类名称。选择 自定义设备类 时，设备类名称不能使用以下名称：`hdd`、`ssd`、`nvme`。
Device Class - Storage Devices	选择节点上的 空白硬盘 或 未格式化硬盘分区。 • “开启全部空白设备”开关打开时：节点下所有空白设备将加入设备类； • 开关关闭时：手动输入节点下空白设备名称，如 `sda`。
Snapshot	启用后支持创建 PVC 快照，并使用快照配置新 PVC，实现业务数据快速备份与恢复。 创建存储时未启用快照，也可在存储集群详情页的 操作 中按需启用。 注意：使用前请确保已为当前集群部署卷快照插件。
Monitoring Alarm	启用后提供开箱即用的监控指标采集和告警能力，详见监控与告警。 注意：若此时未启用，需自行寻找存储监控和告警方案，如在运维中心手动配置监控面板和告警策略。

2. 点击 高级配置 进行组件高级配置。

参数	说明
网络配置	• 主机网络：存储集群使用主机网络，需在优化参数栏填写相关网络优化参数，如配置 `public` 和 `cluster` 子网。留空时使用默认主机子网。 注意：使用主机网络可能因通过主机端口明文传输数据存在安全风险，请联系平台支持团队获取加密传输方案。 • 容器网络：存储集群使用容器网络；可在网络管理中创建子网并分配给 `rook-ceph` 命名空间。留空时使用默认子网。 注意： 不支持 IPv6。 使用容器网络时，存储仅集群内可访问。 Ceph CSI Pod 故障或重启可能导致服务中断。
优化参数	支持填写 Ceph 配置文件格式参数，系统将根据填写内容覆盖默认参数。 注意：首次填写或修改初始化参数后，请点击初始化参数，需初始化成功后方可创建集群。
组件定点部署	可将组件部署到指定节点，至少需三个节点以保证最低可用性。支持定点部署配置的组件包括 MON、MGR、MDS、RGW。

- 页面自动跳转下一步表示 Ceph 集群部署成功。
- 创建失败时，可点击清理 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。

4 创建存储池

1. 在 **Create Storage Pool** 向导页面，配置相关参数，点击右下角 **Create Storage Pool** 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。

参数	说明
	• 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。 • 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、云存储等。
副本数	副本数越多，冗余度和数据安全性越高，但存储利用率降低。通常设置为 3，满足大多数需求。
设备类	对同类型设备或同一业务逻辑的磁盘进行统一分类，从上一步添加的设备类中选择。 • 选择设备类后，数据将存储在所选设备类中。 • 未选择设备类时，数据将在存储池所有设备中随机存储。

对象存储还需配置以下参数：

参数	说明
Region	指定存储池所在的地域。
Gateway Type	默认是 S3，且不可修改。
Internal Port	指定集群内访问端口。
External Access	启用/禁用外部访问将创建/销毁 NodePort 类型服务。
Instance Count	对象存储资源实例数量。

- 页面自动跳转下一步表示存储池部署成功。
- 部署失败时，请根据界面提示检查核心组件，然后点击 清理已创建信息并重试 重新创建存储池。

2. 点击 **Create Storage Pool**，在 详情 标签页查看已创建存储池信息。

相关操作

创建 **Stretch** 类型集群

详情请参见 创建 Stretch 类型集群。

清理分布式存储

详情请参见 清理分布式存储。

创建 **Stretch** 类型集群

Stretch 集群可以跨越两个地理位置不同的站点，提供存储基础设施的灾难恢复能力。当两个可用区中的一个完全不可用时，Ceph 仍能保持可用性。

目录

术语

术语	说明
Quorum 可用区	通常位于不承担主业务负载的独立可用区，专注于维护集群一致性，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	Ceph 集群中实际存储和处理数据的主要区域，承担业务负载和数据存储任务，与仲裁区共同构成完整的高可用存储系统。

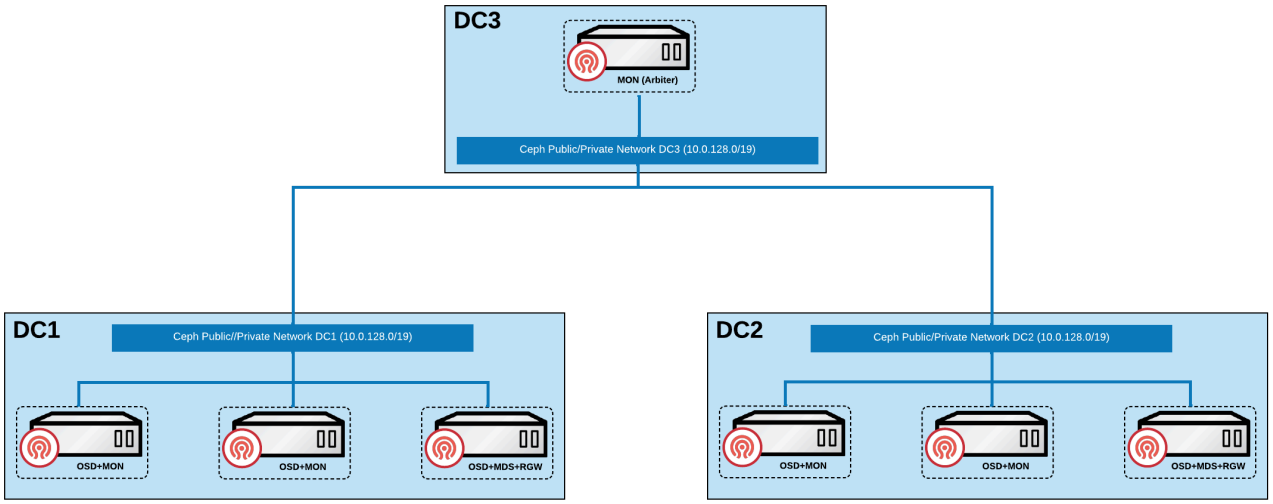
典型部署方案

以下内容提供了 Stretch 集群的典型部署方案，以及组件说明和灾难恢复原理。

组件说明

节点需要分布在三个可用区，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处于故障状态。
- 仲裁可用区只需部署一个 MON 实例，作为仲裁决策节点。



灾难恢复说明

- 当某个数据可用区完全故障时，Ceph 集群会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。由于另一个数据可用区仍保持双副本，集群依然可用。当故障数据可用区恢复后，系统会自动执行数据同步并恢复至健康状态；若故障无法修复，建议更换为新的数据可用区。

- 当两个数据可用区之间的网络连接中断，但仍能正常连接仲裁可用区时，仲裁可用区会根据预设策略对两个数据可用区进行仲裁，选择状态较好的数据可用区继续作为主数据区提供服务。

约束与限制

- 存储池限制：不支持纠删码存储池，仅支持副本机制进行数据保护。

- 设备分类限制：不支持设备类功能，无法基于设备特性进行存储分层。

- 区域部署限制：仅支持两个数据可用区，不允许超过两个数据可用区。

- 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，以确保数据分布均衡。

- 存储介质要求：仅允许全闪（All-Flash）OSD 配置，最大限度减少连接恢复后的恢复时间，降低数据丢失风险。

- 网络延迟要求：两个数据可用区间的 RTT（往返时延）不得超过 10ms，仲裁可用区必须满足 ETCD 规范的延迟要求，以保证仲裁机制的可靠性。

前提条件

请提前将集群中的全部或部分节点划分为三个可用区，具体如下：

- 确保至少 5 个节点分布在一个仲裁可用区和两个数据可用区中。其中，仲裁可用区至少包含一个节点，该节点可以是虚拟机或云主机。

- 确保三个可用区中至少有一个可用区包含 Master 节点（控制节点）。

- 确保至少 4 个计算节点均匀分布在两个数据可用区，每个数据可用区至少配置 2 个计算节点。

- 尽量保证两个数据可用区中的节点数量和磁盘配置一致。

操作步骤

1 标记节点

1. 进入 管理员。
2. 在左侧导航栏点击 集群管理 > 集群。
3. 点击对应集群名称，进入集群概览页面。
4. 切换到 节点 标签页。
5. 根据前提条件中的规划，为这些节点添加 `topology.kubernetes.io/zone=<zone>` 标签，将其划分到指定的可用区。此处将 <zone> 替换为可用区名称。

2 创建存储服务

本文档仅描述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 **Stretch**。
仲裁可用区	选择仲裁可用区名称。
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	默认值为 4。

参数	说明
实例数	当存储类型为 对象存储 时，为保证可用性，实例最小数为 2，最大数为 5。

相关操作

创建标准类型集群

详情请参考创建标准类型集群。

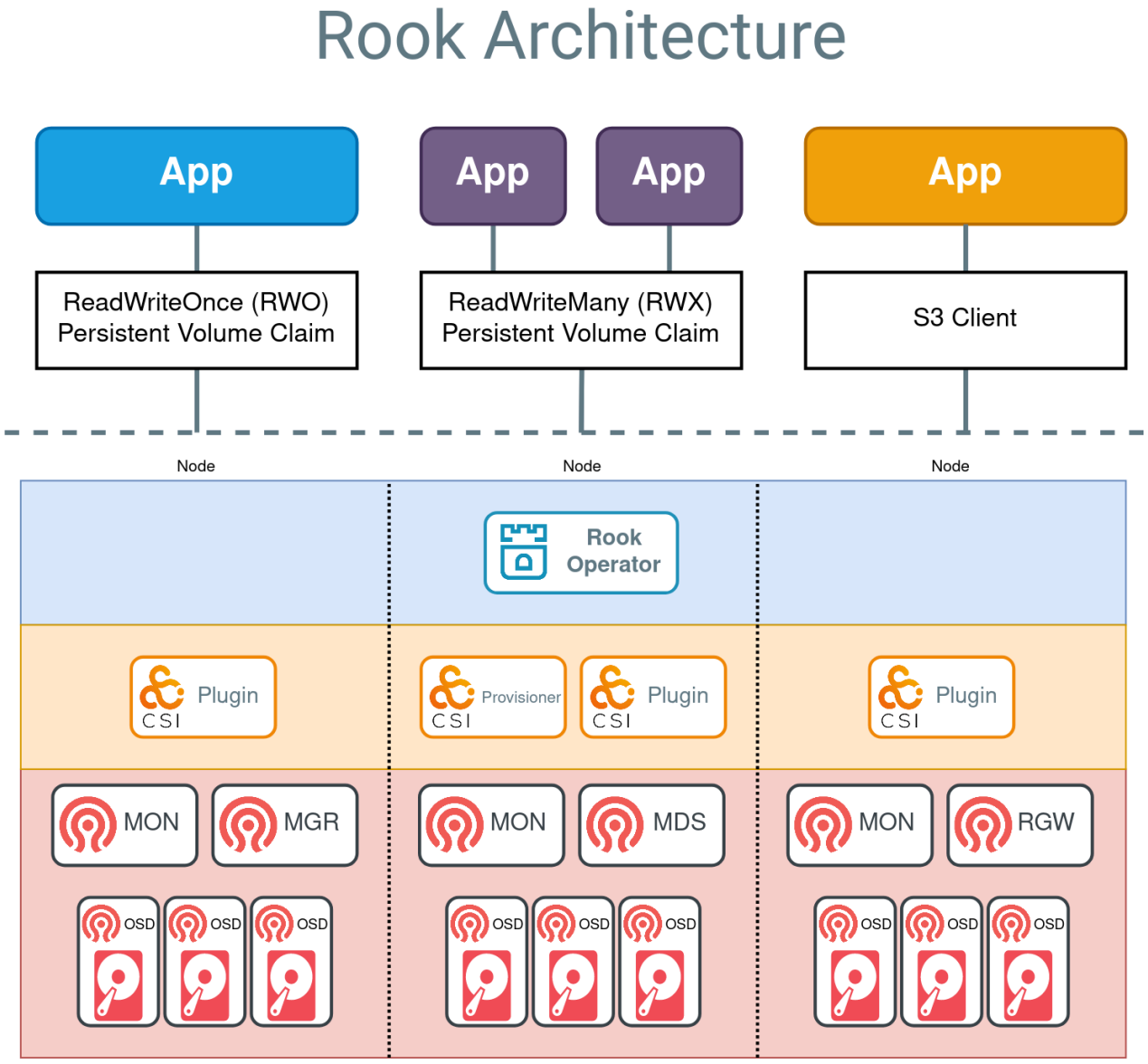
清理分布式存储

详情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。

Menu

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。

模块	功能描述
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。

Menu

操作指南

访问存储服务

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

节点特定组件部署

更新组件部署配置

重启存储组件

添加设备/设备类

添加设备类

添加设备

硬盘状态

监控与告警

访问存储服务

访问存储服务支持两种集成方式：一是集成平台内其他业务集群的分布式存储资源，实现存储与业务隔离，便于管理和维护；二是接入外部 Ceph 存储资源进行分布式存储使用。

目录

前提条件

准备软件包

- 下载对应平台架构的 **Alauda Container Platform Storage Essentials** 安装包。
- 通过 Upload Packages 机制上传 **Alauda Container Platform Storage Essentials** 安装包。

- 下载对应平台架构的 **Alauda Build of Rook-Ceph** 安装包。
- 通过 Upload Packages 机制上传 **Alauda Build of Rook-Ceph** 安装包。

准备存储

选择以下之一：

- 其他业务集群已部署分布式存储，并创建了存储池。请记录存储池名称以备后续集成使用。
- 平台外部已创建外部 Ceph 存储（版本 ≥ 14.2.3）并创建了存储池。请记录存储池名称以备后续集成使用。

开放端口

目标 **IP**	目标端口	源 **IP**	源端口
Ceph 节点 IP	3300, 6789, 6800-7300, 7480	业务集群所有节点 IP	任意

获取认证信息（外部 **Ceph**）

若准备的存储为外部 Ceph 存储，需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` 必须为 {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` 命令获取 `name` 值。 • 块存储：`ceph osd dump` | `grep "application rbd"` | `awk '{print $3}'`

参数	获取方式
数据存储池	（仅文件存储需要）使用 `ceph fs ls` 命令获取 `data pools` 值。

操作步骤

注意：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 部署 Alauda Container Platform Storage Essentials

1. 登录，进入 管理员 页面。
2. 点击 **Marketplace** > **OperatorHub**，进入 **OperatorHub** 页面。
3. 找到 **Alauda Container Platform Storage Essentials**，点击 **Install**，进入 **Install Alauda Container Platform Storage Essentials** 页面。

   配置参数：

参数	推荐配置
Channel	默认 channel 为 `stable`。
Installation Mode	`Cluster`：集群内所有命名空间共享单个 Operator 实例进行创建和管理，资源占用较低。
Installation Place	选择 `Recommended`，命名空间仅支持 **acp-storage**。
Upgrade Strategy	`Manual`：Operator Hub 有新版本时，需要手动确认升级 Operator 到最新版本。

2 访问存储

1. 在左侧导航栏，点击 存储管理 > 分布式存储。
2. 点击 访问存储。

3. 在 访问配置 向导页面，选择 外部 **Ceph**。

参数	说明
快照	启用后，支持创建 PVC 快照，并使用快照配置新的 PVC，实现业务数据的快速备份与恢复。 若访问存储时未启用快照，后续仍可在存储集群详情页的 操作 中根据需要启用。 注意：请确保当前集群已部署卷快照插件后方可使用。
网络配置	• **Host Network**：本集群计算组件将通过 主机网络 访问 存储集群。 • **Container Network**：本集群计算组件将通过 容器网络 访问 存储集群。可在网络管理中创建子网，并将子网分配给 `rook-ceph` 命名空间。若留空，则使用默认子网。
其他参数	请填写前提条件中获取的外部 Ceph 认证参数。

4. 在 创建存储类 向导页面，完成配置后点击 访问。

参数	说明
类型	根据上述创建的存储池类型，默认对应的存储类为： • 文件存储：CephFS 文件存储 • 块存储：CephRBD 块存储
回收策略	持久卷的回收策略。 • 删除：删除持久卷声明时，绑定的持久卷也会被删除。 • 保留：即使删除持久卷声明，绑定的持久卷仍会被保留。
项目分配	可使用该类型存储的项目。 若当前无项目需要该类型存储，可暂不分配项目，后续再更新。

5. 等待约 1-5 分钟，完成集成成功。

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者，可扩展使用卷快照和弹性扩缩容功能。

注意：若需维护外部存储的存储池、存储设备配置等，需在存储集群的管理平台进行操作。

管理存储池

存储池是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

除了在分布式存储配置过程中创建的存储池外，还可以创建其他类型的存储池。

提示：同一存储集群内，只允许存在一个文件存储和一个对象存储池，而块存储池最多可创建八个。

操作步骤

1. 进入 管理员。

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 创建存储池。

4. 按照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。 - 文件存储：提供安全、可靠且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。 - 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。 - 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、云存储服务等。
副本数	• 当集群类型为**Standard**时：副本数越高，冗余和数据安全性越强，但存储利用率降低。通常设置为3即可满足大多数需求。 • 当集群类型为**Extended**时：副本数默认为4，且不可修改。

设备类型	• 当集群类型为**Standard**时：选择已添加到创建的存储池中的设备类型。
	  • 选择设备类型后，数据将存储在所选设备类型中。
	  • 若未选择设备类型，数据将在存储池内所有设备中随机存储。
	• 当集群类型为**Extended**时：不支持添加设备类型。

如果是对象存储类型，还可以配置以下参数：

参数	说明
Region	指定存储池所在的地域。
Gateway Type	默认为S3，且不可修改。
Internal Port	指定集群内部访问端口。
External Access	启用/禁用外部访问将创建/销毁NodePort类型的Service。

参数	说明
Instance Count	对象存储的资源实例数量。

5. 点击 创建。

删除存储池

如果某种类型的存储不再需要，可以在解除与存储类的关联后删除该存储池。

操作步骤

1. 进入 管理员。
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，点击要删除的存储池旁的⋮> 删除。
4. 阅读提示信息并输入存储池名称。
5. 点击 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外部访问地址。

操作步骤

1. 进入 管理员。
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⋮并选择 查看地址。

节点特定组件部署

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，便于存储扩容和维护。

目录

更新组件部署配置

注意事项

- 更新配置会触发系统自动重建组件实例，可能会影响服务对存储系统的访问，建议在业务低峰期进行更新。
- 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 管理员。
>
> 2. 在左侧导航栏点击 存储管理 > 分布式存储。

3. 在 存储组件 标签页下，点击 组件部署配置。
4. 根据业务需求启用/禁用 固定部署 开关，将组件部署到指定节点。节点数量不得少于三个，以保证最低可用性。适用于固定部署配置的组件包括 MON、MGR、MDS、RGW。
5. 点击 更新，组件将开始调度到指定节点。

重启存储组件

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

操作步骤

1. 进入 管理员。
2. 在左侧导航栏点击 存储管理 > 分布式存储。
3. 在 存储组件 标签页下，点击组件名称旁的⋮> 删除。

添加设备/设备类

目录

添加设备类

统一集群节点中同类型设备或具有相同业务逻辑的硬盘的分类，根据存储需求自定义设备类，将不同存储内容分配到不同类型的存储盘。

注意事项

集群类型为 **Extend** 时，不支持添加设备类。

操作步骤

1. 进入 管理员。

2. 在左侧导航栏点击 存储管理 > 分布式存储。

3. 点击 设备类 标签页。

4. 点击 添加设备类，并根据以下说明配置相关参数。

参数	说明
名称	设备类的名称。设备类名称不能使用以下名称：`hdd`、`ssd`、`nvme`。
存储设备	选择节点中的 空白磁盘 或 未格式化磁盘分区。 • 当“所有空设备”开关开启时：将节点下所有空设备添加到该设备类； • 当“所有空设备”开关关闭时：手动输入节点下空设备的名称，例如 `sda`。

添加设备

将可用硬盘映射为存储设备以供使用和管理。

注意：硬盘一旦添加为存储设备，界面不支持更新或移除。

操作步骤

1. 进入 管理员。

2. 在左侧导航栏点击 存储管理 > 分布式存储。

3. 点击 设备类 标签页。

4. 在设备类右侧点击 添加设备，并根据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的节点类型。 计算节点：尚未添加存储设备的节点。 存储节点：已添加存储设备的节点。
添加类型	选择添加硬盘为存储设备的方式。 所有空磁盘：选择将节点中所有未分区挂载的磁盘添加为存储设备。 指定磁盘：选择将节点中部分磁盘添加为存储设备，包括空磁盘或已分区挂载的磁盘。 当节点类型为 存储节点 时，仅可选择 指定磁盘。
指定磁盘	当添加类型为 指定磁盘 时，输入所有要添加为存储设备的硬盘名称，如 `sda`、`sdb`。输入每个硬盘名称后按回车确认。 注意：建议使用整个硬盘作为存储设备，而非硬盘上的分区。

5. 点击 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。

Menu

实用指南

配置专用集群用于分布式存储

清理分布式存储

数据容灾

块存储灾难恢复

术语

备份配置

故障切换

对象存储灾难恢复

术语

前提条件

操作步骤

故障切换

更新优化参数

更新优化参数

操作步骤

创建 **ceph** 对象存储用户

创建 **ceph** 对象存储用户

前提条件

操作步骤

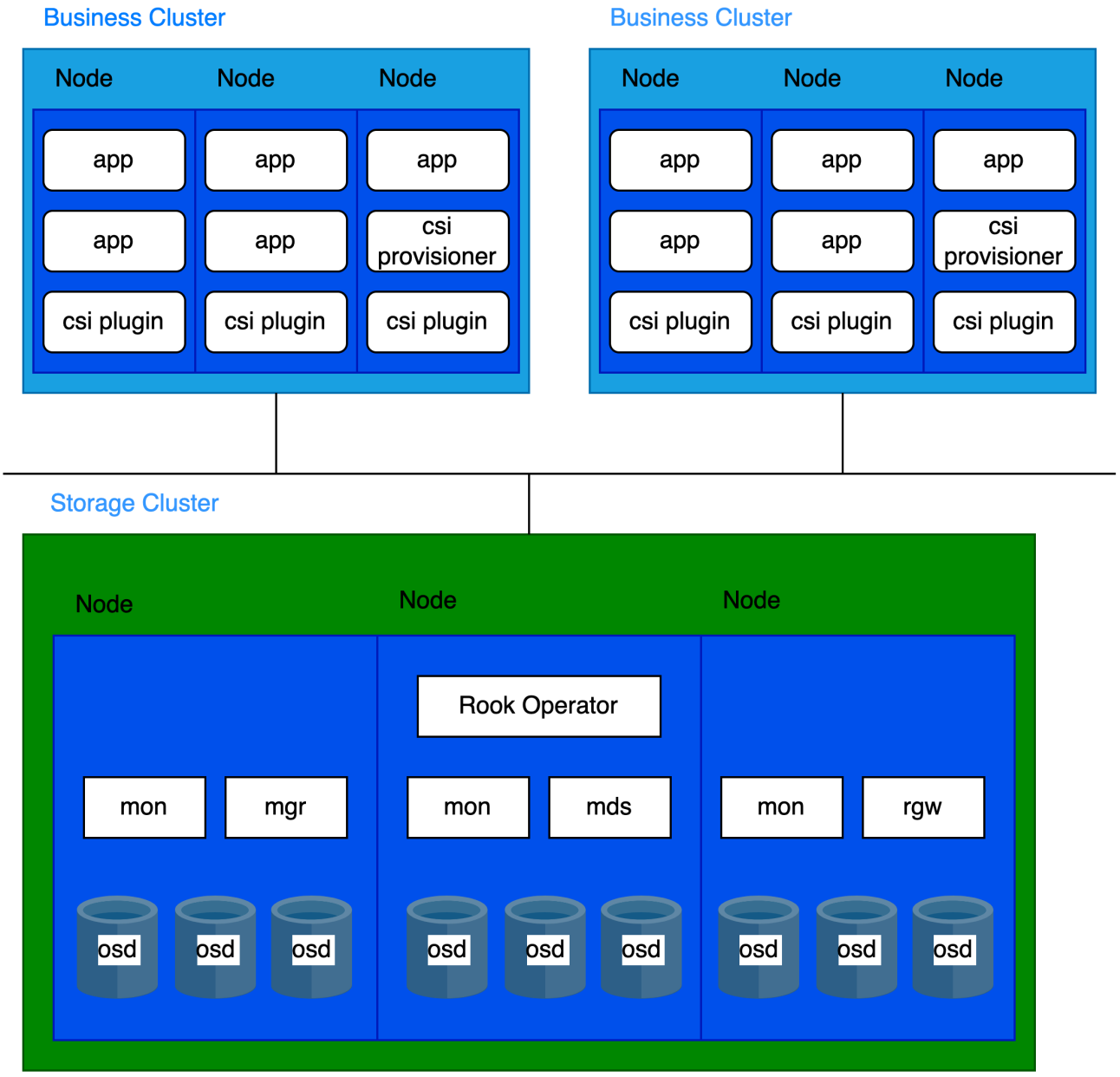
配置专用集群用于分布式存储

专用集群部署是指使用独立集群部署平台的分布式存储，平台内其他业务集群通过集成访问并使用该存储服务。
为保证平台分布式存储的性能和稳定性，专用存储集群仅部署平台核心组件和分布式存储组件，避免与其他业务负载混合部署。此种分离部署方式是平台分布式存储的推荐最佳实践。

目录

架构

存储计算分离架构



基础设施要求

平台要求

支持版本为 3.18 及以后版本。

集群要求

建议使用裸金属集群作为专用存储集群。

资源要求

请参考核心概念了解分布式存储部署的组件。

各组件对 CPU 和内存有不同需求，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 CephObjectStorage）

基于组件分布，单节点资源推荐如下：

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

存储设备要求

建议每节点部署不超过 12 个存储设备，有助于限制节点故障后的恢复时间。

存储设备类型要求

建议使用企业级 SSD，单个设备容量不超过 10TiB，且所有磁盘大小和类型保持一致。

容量规划

部署前需根据具体业务需求规划存储容量。默认分布式存储系统采用 3 副本冗余策略，因此可用容量为所有存储设备总原始容量除以 3。

以 30（N）节点（副本数 = 3）为例，可用容量示例如下：

存储设备大小**(D)**	每节点存储设备数**(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大于消耗容量。存储空间耗尽后，恢复需人工干预，无法通过删除或迁移数据自动解决。

2. 容量告警

   集群在两个阈值触发告警：

   - **80%** 利用率（“接近满”）：主动释放空间或扩容集群。

- **95%** 利用率（“已满”）：存储空间耗尽，常规命令无法释放空间，请立即联系平台支持。

请及时处理告警并定期监控存储使用，避免服务中断。

3. 扩容建议

- 避免：向现有节点添加存储设备。
- 推荐：通过新增存储节点进行横向扩容。
- 要求：新增节点存储设备需与现有节点在容量、类型和数量上保持一致。

网络要求

分布式存储必须使用 **HostNetwork**。

网络隔离

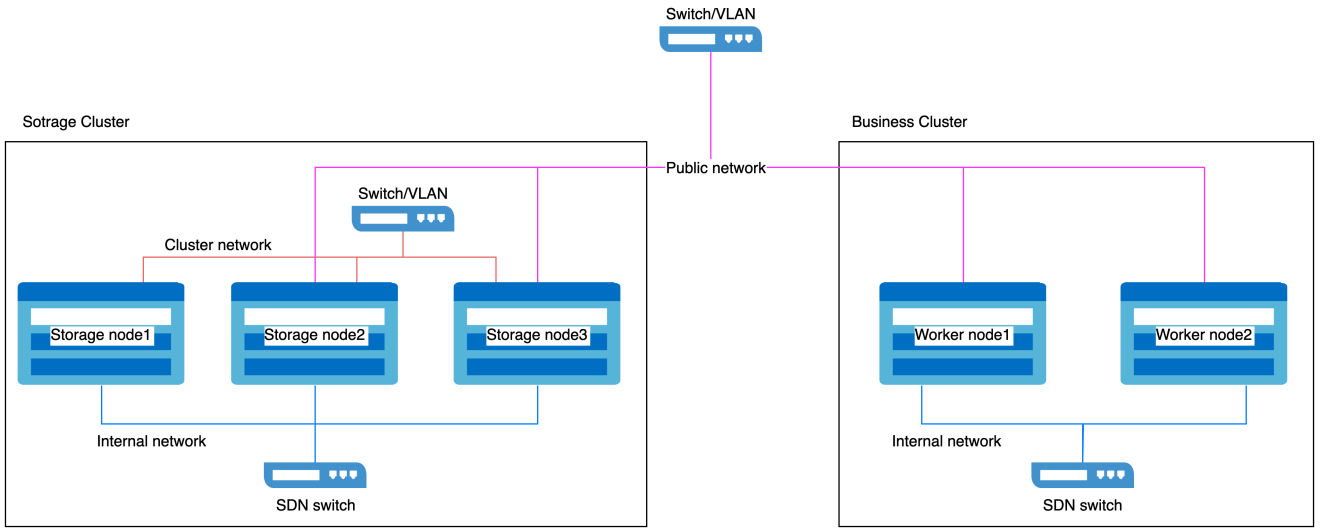
网络分为两类：

- 公网网络：用于客户端与存储组件交互（如 I/O 请求）。
- 集群网络：专用于副本间数据复制及数据重平衡（如恢复）。

为保证服务质量和性能稳定：

1. 对专用存储集群：
   每台主机预留两块网卡：
   - 公网网络：用于客户端与组件通信。
   - 集群网络：用于内部复制和重平衡流量。
2. 对业务集群：
   每台主机预留一块网卡访问存储公网网络。

示例网络隔离配置



网卡速率要求

1. 存储节点

   - 公网网络和集群网络均需 10GbE 或更高速率网卡。

2. 业务集群节点

   - 用于访问存储公网网络的网卡需 10GbE 或更高。

操作步骤

1 部署 **Operator**

1. 进入 管理员。

2. 在左侧菜单点击 存储管理 > 分布式存储。

3. 点击 立即创建。

4. 在 部署 **Operator** 向导页面，点击右下角 部署 **Operator** 按钮。

   - 页面自动跳转到下一步表示 Operator 部署成功。
   - 若部署失败，请根据界面提示选择 清理已部署信息并重试，重新部署 Operator；若需返回分布式存储选择页，点击 应用商店，先卸载已部署的 **rook-operator** 资源，再卸载 **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

参数说明：

- **public network cidr**：存储公网网络的 CIDR（例如 `- 10.0.1.0/24`）。
- **cluster network cidr**：存储集群网络的 CIDR（例如 `- 10.0.2.0/24`）。
- **storage devices**：指定分布式存储使用的存储设备。
  示例格式：

```
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 World Wide Name (WWN) 进行稳定命名，避免依赖重启后可能变化的设备路径如 `sdb`。

3 创建存储池

提供三种存储池类型，根据业务需求选择创建。

创建文件池

在存储集群的 控制节点 执行命令。

▶ 点击查看

创建块池

在存储集群的 控制节点 执行命令。

▶ 点击查看

创建对象池

在存储集群的 控制节点 执行命令。

▶ 点击查看

后续操作

当其他集群需要使用分布式存储服务时，请参考以下指南。
访问存储服务

清理分布式存储

如果需要删除 rook-ceph 集群并重新部署新的集群，应按照本文档依次清理分布式存储服务相关资源。

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 VolumeSnapshotClasses

1. 删除 VolumeSnapshotClasses。

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

2. 验证 VolumeSnapshotClasses 是否已清理。

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

当这些命令无输出时，表示清理完成。

2 删除 StorageClasses

1. 进入 **Administrator**。
2. 在左侧导航栏点击 **Storage Management** > **Storage Classes**。
3. 点击⋮> **Delete**，删除所有使用 Ceph 存储方案的 StorageClasses。

3 删除存储池

此步骤应在完成上一步后执行。

1. 进入 **Administrator**。
2. 在左侧导航栏点击 **Storage Management** > **Distributed Storage**。
3. 在 **Storage Pool Area**，点击⋮> **Delete**，逐个删除所有存储池。当存储池区域显示 **No Storage Pools** 时，表示存储池已成功删除。
4. （可选）如果集群模式为 **Extended**，还需在集群的 Master 节点执行以下命令，删除创建的内置存储池。

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

响应：

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

4 删除 **ceph-cluster**

此步骤应在完成上一步后执行。

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 jobs。

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

4. 验证 ceph-cluster 是否清理完成。

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

当此命令无输出时，表示清理完成。

5 删除 **rook-operator**

此步骤应在完成上一步后执行。

1. 删除 rook-operator。

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

2. 验证 rook-operator 是否清理完成。

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

当此命令无输出时，表示清理完成。

3. 验证所有 ConfigMaps 是否已清理。

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

当此命令无输出时，表示清理完成。如有输出结果，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. 验证所有 Secrets 是否已清理。

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

当此命令无输出时，表示清理完成。如有输出结果，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 '{"metadata":
{"finalizers": []}}'
```

5. 验证 rook-ceph 是否清理完成。

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

当此命令无输出时，表示清理完成。

6 执行清理脚本

完成以上步骤后，表示 Kubernetes 和 Ceph 相关资源已清理干净。接下来需要清理主机上 rook-ceph 的残留。

清理脚本

clean-rook.sh 清理脚本内容如下：

▶ 点击查看

注意事项

清理脚本依赖 sgdisk 命令，请确保在执行清理脚本前已安装该命令。

- 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. 在业务集群中部署分布式存储的每台机器上执行清理脚本 clean-rook.sh。

   ```
   sh clean-rook.sh /dev/[device_name]
   ```

   示例：`sh clean-rook.sh /dev/vdb`

   执行时会提示确认是否真的清理该设备，确认后输入 yes 开始清理。

2. 使用 `lsblk -f` 命令查看分区信息。当该命令输出中 `FSTYPE` 列为空时，表示清理完成。

Menu

数据容灾

文件存储灾备

术语

备份配置

故障切换

块存储灾难恢复

术语

备份配置

故障切换

对象存储灾难恢复

术语

前提条件

操作步骤

故障切换

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，确保在主集群发生故障时，备份集群能够快速接管服务。

WARNING

- CephFS Mirror 基于快照执行增量同步，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
Primary Cluster	当前提供存储服务的集群。
Secondary Cluster	备份集群。

备份配置

前提条件

- 准备两个适合部署 Alauda Build 版本 Rook-Ceph 的集群，分别为 Primary 集群和 Secondary 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 Primary 和 Secondary 集群中分别创建分布式存储服务。
- 在 Primary 和 Secondary 集群中创建同名的文件存储池。

操作步骤

1 在 **Secondary** 集群启用文件存储池的 **Mirror** 功能

在 Secondary 集群的 Control 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间镜像连接的关键凭证。

在 Secondary 集群的 Control 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='{.status.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 Primary 集群创建 Peer Secret

获取 Secondary 集群的 Peer Token 后，需要在 Primary 集群创建 Peer Secret。

在 Primary 集群的 Control 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 **Primary** 集群启用文件存储池的 **Mirror** 功能

在 Primary 集群的 Control 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 Primary 集群部署 Mirror Daemon

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

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: 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 Primary 集群发生故障时，可以直接继续使用 Secondary 集群中的 CephFS。

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 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，不负责 Kubernetes 资源的备份。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 Build 版本 Rook-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 及以上）。
- 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 为 **Primary** 集群的块存储池启用镜像功能

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":"image"}}}'
```

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-ceph -o
jsonpath='{.status.info.rbdMirrorBootstrapPeerSecretName}') \
-o jsonpath='{.data.token}' | base64 -d
```

Output

```
# 输出因敏感信息已截断
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 为 Secondary 集群的块存储池启用镜像功能

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 merge -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 启用复制 **Sidecar**

该功能支持高效的数据复制与同步，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 1 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。

对象存储灾难恢复

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。

术语	说明
Secondary Cluster	用作备份的备用集群。
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。 • 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。 • 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 集群上均未部署任何 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，而是使用以下描述的 CLI 工具进行配置。

操作步骤

本指南提供同一 ZoneGroup 中两个 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：

1. 设置参数

```
export REALM_NAME=<realm-name>
export ZONE_GROUP_NAME=<zonegroup-name>
export PRIMARY_ZONE_NAME=<primary-zone-name>
export PRIMARY_OBJECT_STORE_NAME=<primary-object-store-name>
```

参数说明：

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

2. 创建对象存储

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: $ZONE_GROUP_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: $ZONE_GROUP_NAME
  metadataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
  dataPool:
    failureDomain: host
    replicated:
      size: 3
      requireSafeReplicaSize: true
    parameters:
      compression_mode: none
  preservePoolsOnDelete: false

---
apiVersion: ceph.rook.io/v1
kind: CephObjectStore
metadata:
```

```
  name: $PRIMARY_OBJECT_STORE_NAME
  namespace: rook-ceph
spec:
  gateway:
    port: 7480
    instances: 2
  zone:
    name: $PRIMARY_ZONE_NAME
EOF
```

输出

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

2 配置 **Primary Zone** 的外部访问

1. 获取 ObjectStore 的 UID

```
export PRIMARY_OBJECT_STORE_UID=$(kubectl -n rook-ceph get cephobjectstore
$PRIMARY_OBJECT_STORE_NAME -o jsonpath='{.metadata.uid}')
```

2. 创建外部访问 Service

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-rgw-$PRIMARY_OBJECT_STORE_NAME-external
  namespace: rook-ceph
  labels:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: $PRIMARY_OBJECT_STORE_NAME
  ownerReferences:
    - apiVersion: ceph.rook.io/v1
      kind: CephObjectStore
      name: $PRIMARY_OBJECT_STORE_NAME
      uid: $PRIMARY_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: $PRIMARY_OBJECT_STORE_NAME
  sessionAffinity: None
  type: NodePort
EOF
```

3. 向 CephObjectZone 添加外部端点。

```
IP=$(kubectl get nodes -l 'node-role.kubernetes.io/control-plane' -o
jsonpath='{.items[0].status.addresses[?(@.type=="InternalIP")].address}' | cut -
d ' ' -f1 | tr -d '\n')
PORT=$(kubectl -n rook-ceph get svc rook-ceph-rgw-$PRIMARY_OBJECT_STORE_NAME-
external -o jsonpath='{.spec.ports[0].nodePort}')
ENDPOINT=http://$IP:$PORT
kubectl -n rook-ceph patch cephobjectzone $PRIMARY_ZONE_NAME --type merge -p "
{\"spec\":{\"customEndpoints\":[\"$ENDPOINT\"]}}"
```

3 获取 `access-key` 和 `secret-key`

```
kubectl -n rook-ceph get secrets $REALM_NAME-keys -o jsonpath='{.data.access-key}'
kubectl -n rook-ceph get secrets $REALM_NAME-keys -o jsonpath='{.data.secret-key}'
```

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：

1. 设置参数

```
export REALM_NAME=<realm-name>
export ZONE_GROUP_NAME=<zonegroup-name>
export PRIMARY_ZONE_NAME=<primary-zone-name>
export PRIMARY_OBJECT_STORE_NAME=<primary-object-store-name>

export REALM_ENDPOINT=<realm-endpoint>
export ACCESS_KEY=<access-key>
export SECRET_KEY=<secret-key>

export SECONDARY_ZONE_NAME=<secondary-zone-name>
export SECONDARY_OBJECT_STORE_NAME=<secondary-object-store-name>
```

参数说明：

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

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

2. 创建 **Secondary Zone** 并配置 **Realm** 同步

```
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: $SECONDARY_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
```

5 配置 Secondary Zone 的外部访问

1. 获取 Secondary 网关的 UID

```
export SECONDARY_OBJECT_STORE_UID=$(kubectl -n rook-ceph get cephobjectstore
$SECONDARY_OBJECT_STORE_NAME -o jsonpath='{.metadata.uid}')
```

2. 创建外部访问 Service

```
cat << EOF | kubectl apply -f -
apiVersion: v1
kind: Service
metadata:
  name: rook-ceph-rgw-$SECONDARY_OBJECT_STORE_NAME-external
  namespace: rook-ceph
  labels:
    app: rook-ceph-rgw
    rook_cluster: rook-ceph
    rook_object_store: $SECONDARY_OBJECT_STORE_NAME
  ownerReferences:
    - apiVersion: ceph.rook.io/v1
      kind: CephObjectStore
      name: $SECONDARY_OBJECT_STORE_NAME
      uid: $SECONDARY_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: $SECONDARY_OBJECT_STORE_NAME
  sessionAffinity: None
  type: NodePort
EOF
```

3. 向 Secondary CephObjectZone 添加外部端点

```
IP=$(kubectl get nodes -l 'node-role.kubernetes.io/control-plane' -o jsonpath='{.items[0].status.addresses[?(@.type=="InternalIP")].address}' | cut -d ' ' -f1 | tr -d '\n')
PORT=$(kubectl -n rook-ceph get svc rook-ceph-rgw-$SECONDARY_OBJECT_STORE_NAME-external -o jsonpath='{.spec.ports[0].nodePort}')
ENDPOINT=http://$IP:$PORT
kubectl -n rook-ceph patch cephobjectzone $SECONDARY_ZONE_NAME --type merge -p "{\"spec\":{\"customEndpoints\":[\"$ENDPOINT\"]}}"
```

6 检查 Ceph 对象存储同步状态

在 Primary 集群的 `rook-ceph-tools` pod 中执行以下命令：

```
# 进入 rook-ceph-tools pod
kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get po -l app=rook-ceph-tools
-o jsonpath='{range .items[*]}{@.metadata.name}') -- bash

radosgw-admin sync status
```

输出示例

```
          realm d713eec8-6ec4-4f71-9eaf-379be18e551b (india)
      zonegroup ccf9e0b2-df95-4e0a-8933-3b17b64c52b7 (shared)
           zone 04daab24-5bbd-4c17-9cf5-b1981fd7ff79 (primary)
   current time 2022-09-15T06:53:52Z
zonegroup features enabled: resharding
  metadata sync no sync (zone is master)
      data sync source: 596319d2-4ffe-4977-ace1-8dd1790db9fb (secondary)
                        syncing
                        full sync: 0/128 shards
                        incremental sync: 128/128 shards
                        data is caught up with source
```

`data is caught up with source` 表示同步状态正常。

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 的网关可以继续提供对象存储服务。

操作步骤

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

```
# 进入 rook-ceph-tools pod
kubectl -n rook-ceph exec -it $(kubectl -n rook-ceph get po -l app=rook-ceph-tools -o
jsonpath='{range .items[*]}{@.metadata.name}') -- bash

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

参数说明

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

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。

创建 **ceph** 对象存储用户

我们允许通过自定义资源定义（CRDs）创建和自定义对象存储用户。

目录

前提条件

- 对象存储池已创建

操作步骤

1 创建用户

在集群的控制节点上执行命令。

```
cat << EOF | kubectl apply -f -
apiVersion: ceph.rook.io/v1
kind: CephObjectStoreUser
metadata:
  name: <name>
  namespace: <namespace>
spec:
  store: <ObjectStore>
  displayName: <displayName>
  clusterNamespace: <clusterNamespace>
  quotas:
    maxBuckets: -1
    maxSize: -1
    maxObjects: -1
  capabilities:
    user: "*"
    bucket: "*"
EOF
```

参数说明

参数	描述
name	要创建的对象存储用户的名称。
namespace	创建对象存储用户的命名空间。
displayName	显示名称。
clusterNamespace	父级 `CephCluster` 和 `CephObjectStore` 所在的命名空间。如果未指定，用户必须与集群和对象存储处于同一命名空间。要启用此功能，CephObjectStore 的 `allowUsersInNamespaces` 必须包含该用户的命名空间。
ObjectStore	用户将被创建的对象存储。这与对象存储池的名称相匹配。
quotas	可选 表示可以为用户设置的配额限制。 • maxBuckets：用户的最大桶数量限制。设置为 `-1` 表示无限制。

参数	描述
	• maxSize：所有用户桶中对象的最大总大小限制。设置为 `-1` 表示无限制。 • maxObjects：所有用户桶中对象的最大数量限制。设置为 `-1` 表示无限制。
capabilities	可选 Ceph 允许为用户赋予额外权限。此设置目前仅能在创建对象存储用户时使用。如果需要修改用户权限，必须删除并重新创建用户。详情请参见 Ceph docs↗。我们支持为以下资源添加 `read`、`write`、`read,write` 或 `*` 权限： • user • buckets • usage • metadata • zone • roles • info • amz-cache • bilog • mdlog • datalog • user-policy • odic-provider • ratelimit

2 允许在其他命名空间创建用户

如果在除 Rook 集群命名空间之外的命名空间中创建 CephObjectStoreUser，则必须将该命名空间添加到允许的命名空间列表中，或者指定 "*" 以允许所有命名空间。这对于需要在其自身命名空间中创建对象存储凭据的应用程序非常有用。

在集群的控制节点上执行命令。

```
kubectl -n rook-ceph patch cephobjectstore <ObjectStore> --type merge -p '{"spec":{"allowUsersInNamespaces":["*"]}}'
```

3 获取用户信息

在集群的控制节点上执行命令。

```
user_secret=$(kubectl -n <namespace> get cephobjectstoreuser <user-name> -o
jsonpath='{.status.info.secretName}')

# ACCESS_KEY
kubectl -n <namespace> get secret $user_secret -o jsonpath='{.data.AccessKey}' |
base64 --decode

# SECRET_KEY
kubectl -n <namespace> get secret $user_secret -o jsonpath='{.data.SecretKey}' |
base64 --decode
```

Menu

MinIO 对象存储

介绍

介绍

安装

安装

前提条件

操作步骤

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

操作指南

实用指南

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) Object Storage with MinIO 是基于 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，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。单个对象文件大小可任意，范围从几 KB 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。
- 下载对应您平台架构的 **Alauda Container Platform Storage Essentials** 安装包。

- 通过 Upload Packages 机制上传 **Alauda Container Platform Storage Essentials** 安装包。
- 下载对应您平台架构的 **Alauda Container Platform (ACP) Object Storage with MinIO** 安装包。
- 通过 Upload Packages 机制上传 **Alauda Container Platform (ACP) Object Storage with MinIO** 安装包。

操作步骤

1 部署 Alauda Container Platform Storage Essentials

1. 登录，进入 **Administrator** 页面。
2. 点击 **Marketplace** > **OperatorHub**，进入 **OperatorHub** 页面。
3. 找到 **Alauda Container Platform Storage Essentials**，点击 **Install**，进入 **Install Alauda Container Platform Storage Essentials** 页面。

   配置参数：

参数	推荐配置
Channel	默认通道为 `stable`。
Installation Mode	`Cluster`：集群内所有命名空间共享单个 Operator 实例进行创建和管理，资源占用较低。
Installation Place	选择 `Recommended`，命名空间仅支持 **acp-storage**。
Upgrade Strategy	`Manual`：当 Operator Hub 有新版本时，需手动确认升级 Operator 至最新版本。

2 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 **Deploy Operator**。

   - 页面自动跳转下一步表示 Operator 部署成功。
   - 若部署失败，请根据界面提示执行 **Clean Up Deployed Information and Retry**，然后重新部署 Operator。

3 创建集群

1. 在 **Create Cluster** 向导页面，配置基础信息。

参数	描述
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 一起加密和签名请求。
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，识别发送者并防止请求篡改。

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

参数	描述
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	面向企业级应用，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。
Large scale	面向拥有 10,000,000 个对象存储需求和支持最多 500 个并发请求的集团用户，适用于高负载场景。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。

参数	描述
	• 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	描述
Instance Number	增加 MinIO 集群中的实例数量可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： • 资源消耗增加。 • 若节点承载多个实例，节点故障可能导致多个实例同时下线，降低整体集群可靠性。 注意： • 最小实例数为 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等信息，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	描述
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**。 注意： • 访问协议为 HTTP 时，集群内 pod 可直接通过获取的 IP 或域名访问 MinIO，无需配置 IP 与域名映射；集群内节点可直接通过获取的 IP 访问 MinIO，若需域名访问，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，方可通过域名正常访问。
Access Method	• **NodePort**：在每个计算节点主机上开放固定端口，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性。 • **LoadBalancer**：使用负载均衡器将流量转发至后端服务。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集群。

4 创建 Bucket

登录集群控制节点，使用命令创建 bucket。

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 获取的 IP 地址。
- `accessKey` 和 `secretKey` 填写集群创建时生成的 **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>
```

5 上传/下载文件

创建 bucket 后，可使用命令行上传文件至 bucket 或下载 bucket 中已有文件。

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. 上传文件至 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. 删除已上传文件。

```
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”的结果需向下取整。	
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。
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) 使用 MinIO 的对象存储是一种高性能、分布式的对象存储系统，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 pool 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。

Menu

核心概念

Menu

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。

Menu

操作指南

添加存储池

存储池是指用于存储数据的逻辑分区。在同一存储集群中，可以同时使用不同类型的底层存储，以满足各种业务需求。

除了在配置对象存储时创建的存储池外，还可以添加额外的存储池。

目录

注意事项

添加存储池会导致 MinIO 服务短暂中断，但之后会自动恢复到正常状态。

操作步骤

1. 进入 管理员。
2. 点击左侧导航栏中的 存储管理 > 对象存储。
3. 在 集群信息 标签页下，向下滚动至 存储池 区域，点击 添加存储池。
4. 按照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；平台会为每个选中的存储节点部署 1 个存储服务。 注意：当使用 3 个存储节点时，为保证可靠性，每个节点会部署 2 个存储服务。
单个存储卷容量	单个存储卷 PVC 的容量。每个存储服务管理 1 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 存储池概览 中查看。

5. 点击 确认。

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。它还支持可配置的通知策略，确保运维团队及时获知系统状态。实时监控数据有助于性能调优和运维决策，而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** 下查看实时监控数据。

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整体视图。如果存储状态异常，告警详情将指明根本原因，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

平台内置了预配置的告警策略，用于检测异常并在达到预设阈值时触发通知。这些内置规则涵盖组件健康、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** 中配置通知策略。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，细化您的设置。

Handling Alerts

- **Cluster in "Alert" state**：已触发警告，系统稳定性可能受到影响。请查看 **Live Alerts** 部分获取详情，查明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。

平台将告警按不同严重级别分类，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. 是否有某些告警反复出现？是否可以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，团队能够提升系统韧性，减少停机时间，确保存储业务的顺畅运行。

Menu

实用指南

数据灾难恢复

MinIO 支持通过远程数据备份或主动-主动部署建立灾难恢复中心，以确保在灾难发生时原始数据不丢失或损坏，从而保障数据的安全性和可靠性。

目录

适用场景

- 热备份：在同一城市或不同地点有两个数据中心，一个为主用，一个为备份。数据实时从主集群复制到备份集群，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份集群，保证业务连续性。
- 城市级主动-主动：在城市级主动-主动（多集群）架构中，有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，可同时接收业务流量。当其中一个数据中心遇到灾难时，业务可以在另一个数据中心不间断运行。

术语

- 主集群：指当前处于活动状态并处理业务请求的集群。它是数据的来源或操作的发起方。在主集群中，数据被创建、修改或更新，业务流量首先发送到该集群进行处理。
- 目标集群：指接收数据复制、迁移或故障切换的集群。通常处于备份或待机状态，等待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。在主动-主动场景中，两个集群可以互为目标集群。

前提条件

- 主集群和目标集群均需开启外网访问。具体配置方法请参见 Create Object Storage。
- 主集群必须使用 **LoadBalancer** 访问方式，目标集群建议支持 负载均衡器 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，主集群和目标集群均需配置自身及对方的 DNS 解析。
- 使用 HTTPS 协议时，建议主集群和目标集群均使用 CA 签发的证书以确保通信安全可信；若使用自签名证书，双方必须导入并信任对方的自签名证书，才能成功建立安全的 HTTPS 连接。

操作步骤

1. 进入 管理员。
2. 在左侧导航栏点击 存储管理 > 对象存储。
3. 在 数据灾难恢复 标签页，点击 添加目标集群。
4. 按照以下说明配置目标集群的相关参数。

参数	说明
访问地址	目标集群的外部访问地址，以 http:// 或 https:// 开头。

参数	说明
Access Key	目标集群的 Access Key ID。与私有访问密钥关联的唯一标识符；用于配合私有访问密钥对请求进行加密。
Secret Key	与 Access Key ID 配合使用的私有访问密钥，用于加密请求、识别发送方及防止请求被篡改。

5. 点击 添加。

- 添加成功后，您可以查看目标集群的状态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。
Buckets	待同步和已同步的桶数量。 • 在热备份场景中，待同步指主集群需要同步到目标集群的桶数量。 • 在城市级主动-主动场景中，待同步指主集群和目标集群之间需要同步的桶总数。
Objects	桶中同步失败的对象数量。 注意：该数字仅供参考，因为 MinIO 在同步过程中会同步相关文件配置。
网络流量速率	主集群的网络进出速率。 • 在热备份场景中，网络入口速率始终为 0。 • 在城市级主动-主动场景中，入口和出口速率均有数据。

- 如果添加目标集群失败，可以点击 重新添加，清除集群信息并返回添加目标集群页面，重新添加目标集群。

相关操作

当不再需要灾难恢复时，可以点击 移除目标集群。移除目标集群不会删除已同步的数据；如果当前有数据正在同步，将会被中断。

Menu

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

实用指南

使用 **Velero** 备份和恢复 **TopoLVM** 文件系统 **PVC**

前提条件

限制

操作步骤

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

Local storage 是一种软件定义的服务器本地存储解决方案，提供简单、易维护且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 方式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必须安装 lvm2 包。如未安装，请在节点上执行 `yum install -y lvm2` 命令。
- 下载对应您平台架构的 **Alauda Container Platform Storage Essentials** 安装包。
- 通过 Upload Packages 机制上传 **Alauda Container Platform Storage Essentials** 安装包。
- 下载对应您平台架构的 **Alauda Build of TopoLVM** 安装包。
- 通过 Upload Packages 机制上传 **Alauda Build of TopoLVM** 安装包。

操作步骤

1 部署 Alauda Container Platform Storage Essentials

1. 登录，进入 **Administrator** 页面。
2. 点击 **Marketplace** > **OperatorHub**，进入 **OperatorHub** 页面。
3. 找到 **Alauda Container Platform Storage Essentials**，点击 **Install**，进入 **Install Alauda Container Platform Storage Essentials** 页面。

   配置参数：

参数	推荐配置
Channel	默认通道为 `stable`。
Installation Mode	`Cluster`：集群内所有命名空间共享单个 Operator 实例进行创建和管理，资源占用较低。
Installation Place	选择 `Recommended`，命名空间仅支持 **acp-storage**。
Upgrade Strategy	`Manual`：当 Operator Hub 有新版本时，需要手动确认升级 Operator 到最新版本。

2 部署存储

1. 进入 **Administrator**。
2. 在左侧导航栏点击 **Storage Management** > **Local Storage**。
3. 点击 **Configure Now**。
4. 在 **Install Operator** 向导页面，点击 **Start Deployment**。
   - 页面自动跳转到下一步表示 Operator 部署成功。

- 若部署失败，请根据界面提示排查解决，然后点击 **Clean Up** 并重新部署 Operator。

5. 在 **Create Cluster** 向导页面，添加设备。

参数	说明
Select Node	至少有 1 块裸盘的节点。
Device Class	每个设备类对应一组具有相同特性的存储设备，建议根据磁盘性质填写名称，如 *hdd*、*ssd*。
Device Type	仅支持磁盘类型。
Storage Device	例如 */dev/sda*。若有多块磁盘，可逐一添加。
Snapshot	启用后支持创建 PVC 快照，并使用快照配置新的 PVC，实现业务数据的快速备份与恢复。 若创建存储时未启用快照，后续可在存储集群详情页的 **Operations** 部分按需启用。 注意：使用前请确保当前集群已部署Volume Snapshot Plugin。

- 点击下一步，页面自动跳转表示集群部署成功。
- 若创建失败，请根据界面提示及时清理资源。

6. 在 **Create Storage Class** 向导页面，配置相关参数。

参数	说明
Name	存储类名称，必须在当前集群内唯一。
Display Name	便于识别或筛选的名称，如存储类的中文描述。
Device Class	设备类是 TopoLVM 中对存储设备的分类方式，每个设备类对应一组具有相同特性的存储设备。无特殊需求时，可使用集群中的 **Auto-Allocated** 设备类。

参数	说明
File System	- **XFS** 是一种高性能的日志文件系统，擅长处理并行 I/O 负载，支持大文件处理并提供流畅的数据传输。 - **EXT4** 是 Linux 中的日志文件系统，采用 extent 文件存储方式，支持大文件处理。文件系统容量可达 1 EiB，最大支持文件大小为 16 TiB。
Recycling Policy	持久卷的回收策略。 - Delete：删除持久卷声明时，绑定的持久卷也会被删除。 - Retain：即使删除持久卷声明，绑定的持久卷仍会保留。
Access Mode	ReadWriteOnce (RWO)：可被单个节点以读写模式挂载。
Allocation Project	该类型的持久卷声明只能在特定项目中创建。 若暂未分配项目，也可后续进行 更新。

7. 点击 **Next**，等待资源创建完成。

Menu

操作指南

设备管理

前提条件

添加设备

监控与告警

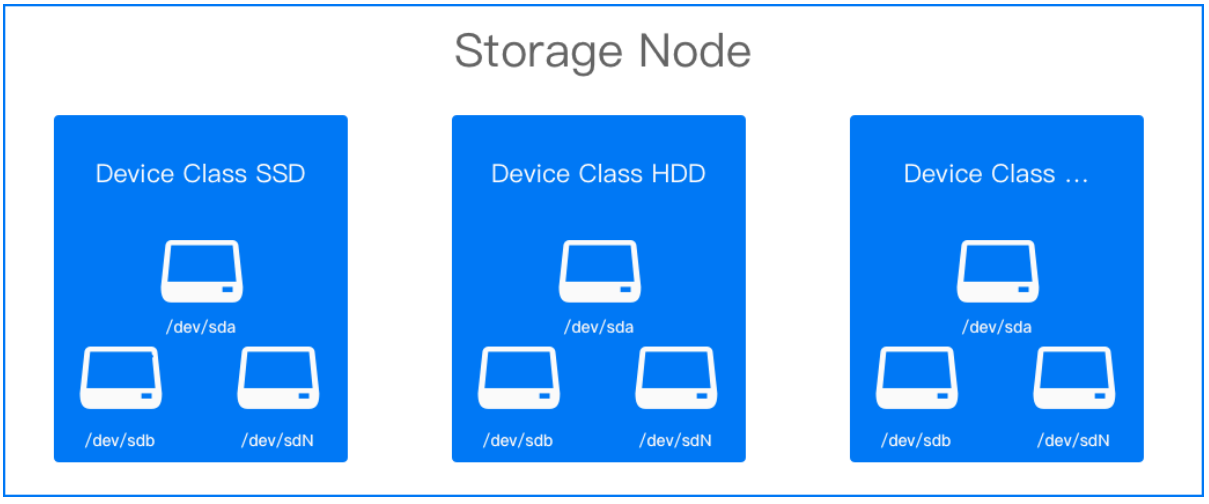
监控功能

告警功能

设备管理

无论是初次部署还是资源扩容，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，必须至少添加了 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 管理员。

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。 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` ，可以继续进行以下操作。

6. 切换到 存储设备 标签页，点击 添加存储设备。

7. 按照界面提示添加设备。

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？

Menu

实用指南

使用 Velero 备份和恢复 TopoLVM 文件系统 PVC

Velero 支持对 TopoLVM 文件系统的 Persistent Volume Claims（PVC）和 Persistent Volumes（PV）进行备份和恢复。此功能已集成到平台中。

本指南专门针对 TopoLVM 文件系统的 PVC。

目录

前提条件

1. 通过 Marketplace/Cluster Plugins 部署“Alauda Container Platform Data Backup for Velero”。
2. 为 Velero 的 `BackupStorageLocation` 配置一个兼容 S3 的存储。可使用平台提供的 Ceph 或 MinIO 对象存储。

限制

1. S3 存储必须有足够的剩余空间以存储目标集群中所有 PV 的数据。
2. 恢复时，命名空间配额和存储类必须支持所有 PVC 的总容量。

操作步骤

第 1 步：配置备份仓库

1. 确保已有兼容 S3 的存储可用。
2. 进入 管理员 > 集群管理 > 备份与恢复 > 备份仓库。
3. 使用对象存储的凭据创建备份仓库。

第 2 步：执行备份

1. 给需要备份的 PVC 和相关 Pod 添加标签：

   Velero 需要一个 Pod 来恢复文件系统 PVC。该 Pod 挂载 PVC，Velero 通过它导入数据；如果没有 Pod，PVC 会一直处于 Pending 状态。对于复杂应用，建议暂停应用，将 PVC 挂载到轻量级 Pod（如 Nginx）上进行备份/恢复，恢复完成后再恢复原应用配置。

   ```
   kubectl label pvc -n <namespace> <pvc-name> velero-backup=true
   kubectl label pod -n <namespace> <pod-name> velero-backup=true
   ```

2. 进入 备份与恢复，创建新的备份任务：

   - 选择 备份 **Kubernetes** 资源和 **PVC** 数据卷。
   - 选择包含需要备份数据的命名空间。
   - 按照以下配置设置备份：

```yaml
apiVersion: velero.io/v1
kind: Schedule
metadata:
  name: <backup-name>
  namespace: cpaas-system
  annotations:
    cpaas.io/description: ''
spec:
  template:
    includedNamespaces:
      - <namespace>
    includedResources:
      - '*'
    labelSelector:
      matchLabels:
        velero-backup: 'true'
    excludedNamespaces: []
    excludedResources: []
    defaultVolumesToFsBackup: true
    storageLocation: default
    ttl: 720h
  schedule: '@every 876000h'
  skipImmediately: false
status:
  phase: Enabled
```

3. 备份完成后，验证 S3 存储桶（如 MinIO）中的数据：

```
mc ls <minio-alias>/<bucket-name>/<backup-path>/<namespace>/
```

示例输出：

```
[2025-03-14 00:18:33 CST]  155B STANDARD config
[2025-03-14 09:04:56 CST]   0B data/
[2025-03-14 09:04:56 CST]   0B index/
[2025-03-14 09:04:56 CST]   0B keys/
[2025-03-14 09:04:56 CST]   0B snapshots/
```

第 3 步：恢复集群

1. 在目标集群中配置与备份时相同的 S3 存储桶，Velero 会自动识别已有备份。
2. 进入 备份与恢复，创建恢复任务：
   - 选择需要恢复的命名空间。
   - 在高级配置中，如有需要，将原命名空间映射到目标命名空间。
3. 执行恢复操作。
4. 恢复完成后，确认：
   - PVC 名称与原集群一致。
   - PVC 中的应用数据完整且可访问。