Red Hat OpenShift Data Foundation 4.12

Red Hat OpenShift Data Foundation 架构

OpenShift Data Foundation 架构概述，以及组件和服务所执行的角色。

Last Updated: 2025-04-27

Red Hat OpenShift Data Foundation 4.12 Red Hat OpenShift Data Foundation 架构

OpenShift Data Foundation 架构概述，以及组件和服务所执行的角色。

法律通告

摘要

本文档概述 OpenShift Data Foundation 架构。

目录

前言

本文档概述 OpenShift Data Foundation 架构。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请告诉我们如何让它更好。

要提供反馈，请创建一个 Bugzilla ticket：

1. 进入 Bugzilla 网站。
2. 在 **Component** 部分中，选择 **文档**。
3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
4. 点 **Submit Bug**。

第 1 章 OPENSHIFT DATA FOUNDATION 介绍

Red Hat OpenShift Data Foundation 是 Red Hat OpenShift Container Platform 的云存储和数据服务的高度集成集合。它作为 Red Hat OpenShift Container Platform Service Catalog 的一部分提供，它作为一个 operator 提供，以便于简单部署和管理。

Red Hat OpenShift Data Foundation 服务主要通过代表以下组件的存储类提供给应用程序：

- 块存储设备，主要服务于数据库工作负载。示例包括 Red Hat OpenShift Container Platform 日志记录和监控，以及 PostgreSQL。
- 共享和分布式文件系统，主要服务于软件开发、消息传递和数据聚合工作负载。示例包括 Jenkins 构建源和工件、Wordpress 上传的内容、Red Hat OpenShift Container Platform registry，以及使用 JBoss AMQ 的消息传递。
- 多云对象存储，具有一个轻量级 S3 API 端点，可以从多个云对象存储中提取存储和检索数据。
- 在内部对象存储中，具有一个稳定的 S3 API 端点，可扩展到数十拍字节（PB）和数十亿个对象的环境，主要面向数据密集型应用。例如，使用 Spark、Pacesto、Red Hat AMQ Streams (Kafka) 等应用程序，以及 TensorFlow 和 Pytorch 等机器学习框架。

注意

不支持在 CephFS 持久性卷上运行 PostgresSQL 工作负载，建议使用 RADOS 块设备 (RBD) 卷。如需更多信息，请参阅知识库文章解决方案 ODF 数据库工作负载必须不使用 CephFS PV/PVC。

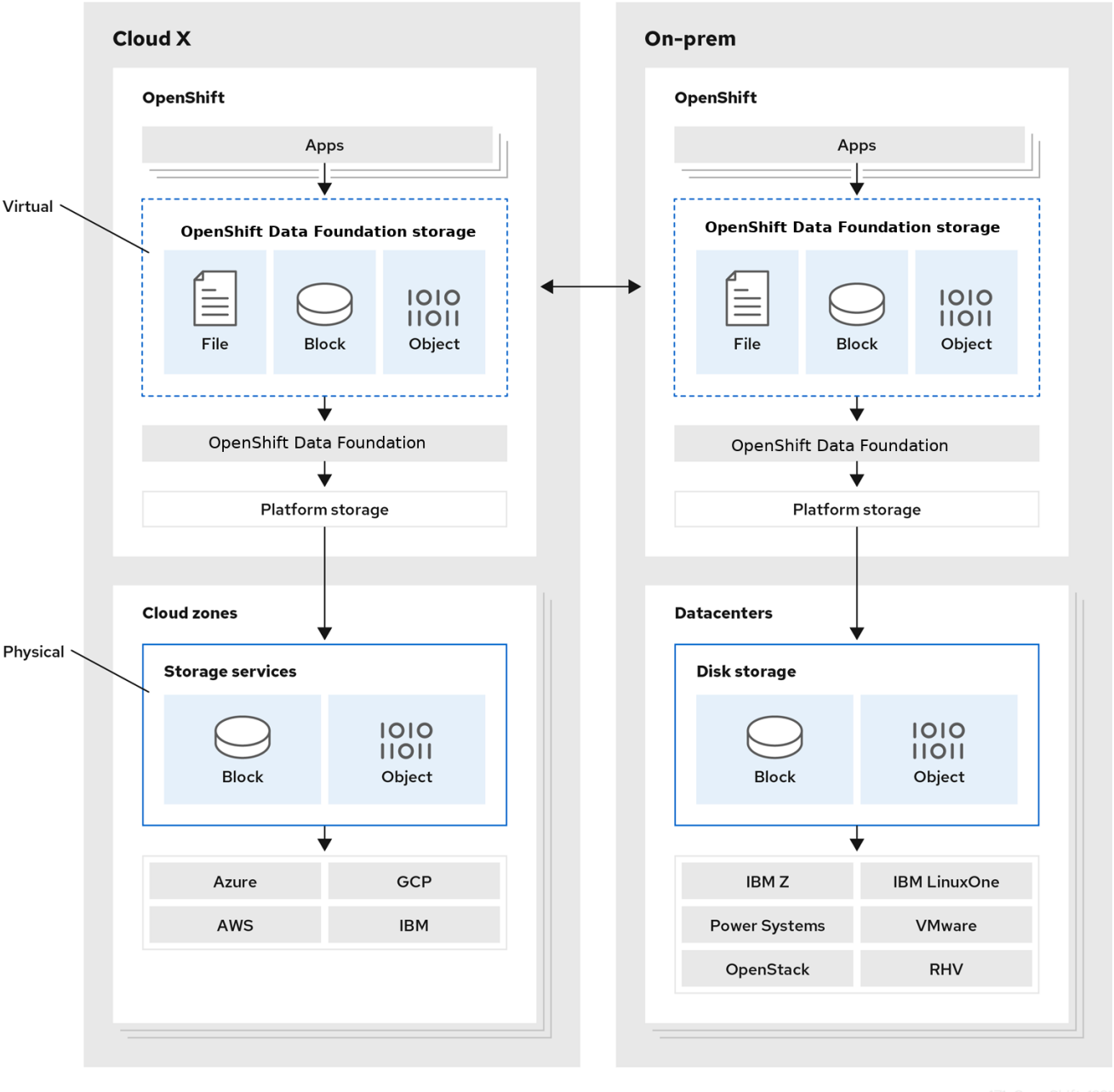
Red Hat OpenShift Data Foundation 版本 4.x 由一组软件项目组成，包括：

- Ceph，提供块存储、共享分布式文件系统以及内部对象存储
- Ceph CSI，用于管理持久性卷和声明的调配和生命周期
- NooBaa 提供多云对象网关
- OpenShift Data Foundation、Rook-Ceph 和 NooBaa 操作器，用于初始化和管理 OpenShift Data Foundation 服务。

第 2 章 OPENSHIFT DATA FOUNDATION 架构概述

Red Hat OpenShift Data Foundation 为 Red Hat OpenShift Container Platform 提供服务，也可以从 Red Hat OpenShift Container Platform 内部运行。

图 2.1. Red Hat OpenShift Data Foundation 架构



Red Hat OpenShift Data Foundation 支持部署到在 Installer Provisioned Infrastructure 或 User Provisioned Infrastructure 上部署的 Red Hat OpenShift Container Platform 集群。有关这两种方法的详情，请参阅 OpenShift Container Platform - 安装过程。要了解更多有关 Red Hat OpenShift Data Foundation 和 Red Hat OpenShift Container Platform 组件互操作性的信息，请参阅互操作性列表。

如需有关 OpenShift Container Platform 架构和生命周期的信息，请参阅 OpenShift Container Platform 架构。

第 3 章 OPENSHIFT DATA FOUNDATION 操作器（OPERATOR）

Red Hat OpenShift Data Foundation 由以下三个 Operator Liftcycle Manager(OLM) Operator 捆绑包组成，部署四个操作器来整合管理任务和自定义资源，以便轻松自动执行任务和资源特性：

- OpenShift Data Foundation
  - **odf-operator**
- OpenShift Container Storage
  - **ocs-operator**
  - **rook-ceph-operator**
- 多云对象网关
  - **mcg-operator**

管理员定义集群的所需最终状态，OpenShift Data Foundation 通过最少的管理员干预来确保集群处于该状态，或接近该状态。

3.1. OPENSHIFT DATA FOUNDATION OPERATOR（操作器）

odf-operator 可以被理解为 OpenShift Data Foundation 的 "meta" 操作器，它是一个旨在影响其他操作器的操作器。

odf-operator 有以下主要功能：

- 强制组成 OpenShift Data Foundation 的其他操作器的配置和版本控制。它通过使用两种主要机制来实现此目的：操作器依赖项和订阅管理。
  - **odf-operator** 捆绑包指定其他 OLM Operator 的依赖关系，以确保它们始终安装在特定版本中。
  - operator 本身为所有其他操作器管理订阅，以确保所需的 Operator 版本可供 OLM 安装。
- 为 OpenShift 控制台提供 OpenShift Data Foundation 外部插件。
- 提供一个 API，将存储解决方案与 OpenShift 控制台集成。

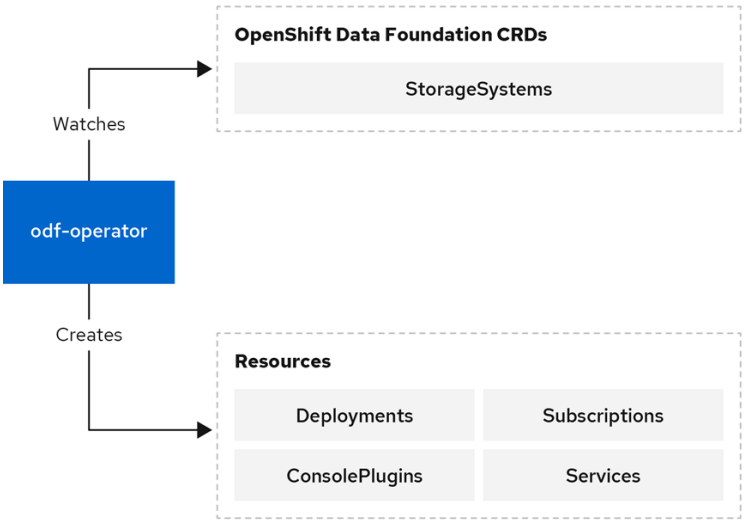
3.1.1. 组件

odf-operator 依赖于 **ocs-operator** 软件包。它还管理 stc **g-operator** 的订阅。此外，**odf-operator** 捆绑包为 OpenShift 控制台的 OpenShift Data Foundation 外部插件定义第二个部署。这会定义基于 **nginx** 的 Pod，它提供必要的文件来注册 OpenShift Data Foundation 仪表板并将其集成到 OpenShift Container Platform 控制台中。

3.1.2. 设计图

本图演示了 **odf-operator** 如何与 OpenShift Container Platform 集成。

图 3.1. OpenShift Data Foundation Operator



3.1.3. 响应能力

odf-operator 定义以下 CRD：

- **StorageSystem**

StorageSystem CRD 代表了一个底层存储系统，为 OpenShift Container Platform 提供数据存储和服务。它触发 Operator 以确保给定存储系统的 **Kind** 存在 订阅。

3.1.4. Resources

ocs-operator 会根据给定存储系统的 spec 创建以下 CR。

Operator Lifecycle Manager 资源

为操作器创建一个 **订阅**，用于定义和协调给定 **存储系统的** Kind。

3.1.5. 限制

odf-operator 本身不提供任何数据存储或服务。它作为其他存储系统的集成和管理层存在。

3.1.6. 高可用性

高可用性并不是 **odf-operator** Pod 的主要要求，与其他多数操作器类似。总体而言，并没有操作需要或受益于流程分布。在当前 Pod 不可用或被删除时，OpenShift Container Platform 会快速启动一个替换的 Pod。

3.1.7. 相关配置文件

odf-operator 附带一个变量 **ConfigMap**，可用于修改 Operator 的行为。

3.1.8. 相关日志文件

要了解 OpenShift Data Foundation 并排除问题，您可以查看以下内容：

- Operator Pod 日志

- **StorageSystem** 状态
- 底层存储系统 CRD 状态

Operator Pod 日志

每个操作器都提供标准的 Pod 日志，其中包括有关协调的信息以及遇到的错误的信息。这些日志通常具有有关成功协调的信息，可以过滤掉并忽略它们。

StorageSystem 状态和事件

StorageSystem CR 将协调详情存储在 CR 的状态中，并关联事件。**StorageSystem 的** spec 包含实际存储系统的 CRD 的名称、命名空间和 Kind，管理员可以用它来查找存储系统状态的更多信息。

3.1.9. 生命周期

需要存在 **odf-operator**，只要 OpenShift Data Foundation 捆绑包仍然被安装。这是作为 OLM 对 OpenShift Data Foundation CSV 协调的一部分进行管理。至少一个 pod 实例应该处于 **Ready** 状态。

CRD 等 operator 操作对象不应影响 Operator 的生命周期。创建和删除 **StorageSystems** 是在操作器控制之外的操作，必须由管理员发起，或者通过适当的应用程序编程接口(API)调用进行自动化。

3.2. OPENSHIFT CONTAINER STORAGE OPERATOR

ocs-operator 可以被理解为 OpenShift Data Foundation 的"meta"操作器，即操作器旨在影响其他操作器，并充当其他操作器提供的功能的配置网关。它不直接管理其他操作器。

ocs-operator 有以下主要功能：

- 创建自定义资源(CR)，以触发其他操作器进行协调。
- 将 Ceph 和多云对象网关配置抽象，并将其限制为由红帽验证和支持的已知最佳实践。
- 根据支持策略创建并协调部署容器化 Ceph 和 NooBaa 所需的资源。

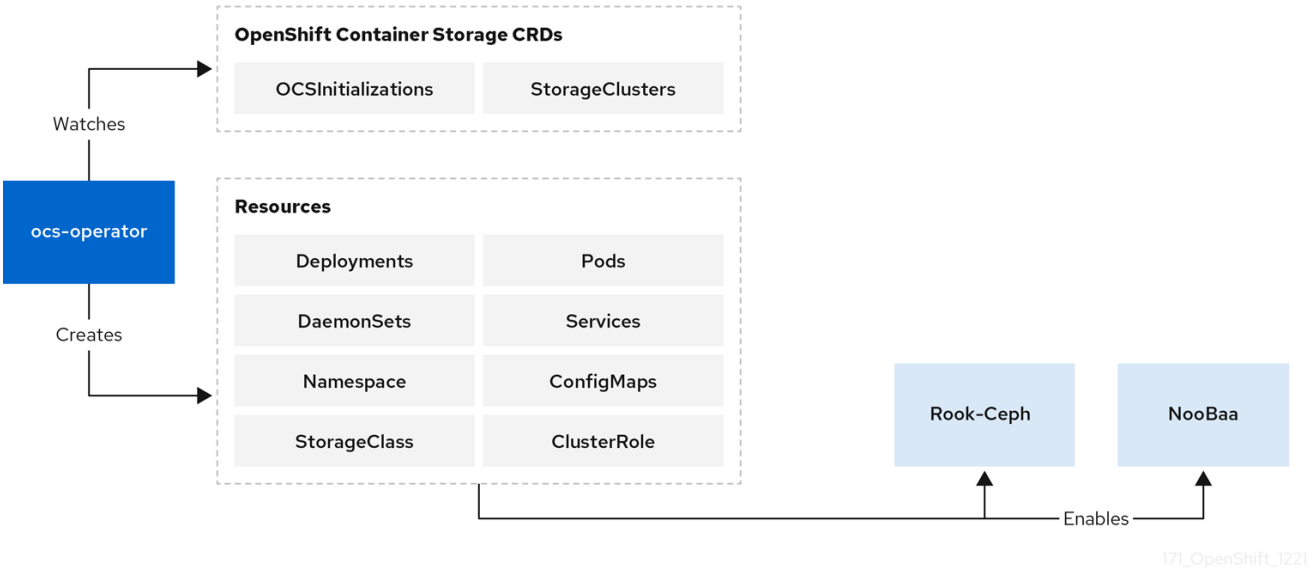
3.2.1. 组件

ocs-operator 没有任何依赖组件。但是，Operator 依赖于是否存在来自其他 Operator 的所有自定义资源定义(CRD)，这些定义在 **ClusterServiceVersion** (CSV)中定义。

3.2.2. 设计图

本图演示了 OpenShift Container Storage 如何与 OpenShift Container Platform 集成。

图 3.2. OpenShift Container Storage Operator



3.2.3. 职责

这两个 **ocs-operator** CRD 是：

- **OCSInitialization**
- **StorageCluster**

OCSInitialization 是一个单例 CRD，用于封装应用到 operator 级别的操作。操作器负责确保一个实例始终存在。CR 会触发以下内容：

- 执行 OpenShift Container Storage 所需的初始化任务。如果需要，可以通过删除 **OCSInitialization** CRD 来触发这些任务来再次运行。
  - 确保存在 OpenShift Container Storage 所需的安全性上下文约束(SCC)。
- 管理 Ceph toolbox Pod 的部署，用于执行高级故障排除和恢复操作。

StorageCluster CRD 代表提供 OpenShift Container Storage 完整功能的系统。它触发操作器，以确保 **Rook-Ceph** 和 **NooBaa** CRD 的生成与协调。**ocs-operator** 算法根据 **StorageCluster** spec 中的配置生成 **CephCluster** 和 **NooBaa** CRD。Operator 还会创建额外的 CR，如 **CephBlockPools** 和 **Routes** 等。启用 OpenShift Container Storage 的不同功能需要这些资源。目前，每个 OpenShift Container Platform 集群只支持一个 StorageCluster CR。

3.2.4. Resources

ocs-operator 会根据定义的 CRD 的 spec 创建以下 CR。可以覆盖其中部分资源的配置，允许更改生成的 spec 或不能完全创建这些资源。

常规资源

事件

在需要时创建各种事件以响应协调。

持久性卷(PV)

PV 不会被 Operator 直接创建。但是，Operator 会跟踪 Ceph CSI 驱动程序创建的所有 PV，并确保 PV 具有支持功能的适当注解。

Quickstarts

为 OpenShift Container Platform 控制台部署各种 Quickstart CR。

Rook-Ceph 资源

CephBlockPool

定义默认的 Ceph 块池。Ceph 对象存储的 **CephFilesysPrometheusRulesoute**。

StorageClass

定义默认存储类。例如，Ceph **BlockPool** 和 **CephFilesystem**。

VolumeSnapshotClass

为对应存储类定义默认卷快照类。

多云对象网关资源

NooBaa

定义默认的 Multicloud 对象网关系统。

监控资源

- 指标导出器服务
- 指标导出器服务监控器
- PrometheusRules

3.2.5. 限制

ocs-operator 都部署或协调 OpenShift Data Foundation 的其他 Pod。 **ocs-operator** CSV 定义顶层组件，如 operator Deployments，Operator Lifecycle Manager(OLM)协调指定的组件。

3.2.6. 高可用性

高可用性并不是 **ocs-operator** Pod 的主要要求，与其他多数操作器类似。总体而言，并没有操作需要或受益于流程分布。在当前 Pod 不可用或被删除时，OpenShift Container Platform 会快速启动一个替换的 Pod。

3.2.7. 相关配置文件

ocs-operator 配置完全由 CSV 指定，在没有 CSV 自定义构建的情况下不可修改。

3.2.8. 相关日志文件

要了解 OpenShift Container Storage 并排除问题，您可以参阅以下内容：

- Operator Pod 日志
- Storagecluster 状态和事件
- OCSInitialization 状态

Operator Pod 日志

每个操作器都提供标准的 Pod 日志，其中包括有关协调的信息以及遇到的错误的信息。这些日志通常具有有关成功协调的信息，可以过滤掉并忽略它们。

Storagecluster 状态和事件

StorageCluster CR 将协调详情存储在 CR 的状态中，并关联事件。status 包含预期容器镜像的一个部分。它显示了应当存在于其他操作器的容器集中的容器镜像，以及它当前检测到的镜像。这有助于确定 OpenShift Container Storage 升级是否完成。

OCSInitialization 状态

此状态显示初始化任务是否已成功完成。

3.2.9. 生命周期

只要仍然安装了 OpenShift Container Storage 捆绑包，就需要 **ocs-operator**。这是作为 OLM 对 OpenShift Container Storage CSV 协调的一部分进行管理。至少一个 pod 实例应该处于 **Ready** 状态。

CRD 等 operator 操作对象不应影响 Operator 的生命周期。**OCSInitialization** CR 应始终存在。如果不存在，Operator 会创建一个。创建和删除 StorageClusters 是在 Operator 控制之外的操作，必须由管理员发起，或者使用适当的 API 调用自动执行。

3.3. ROOK-CEPH OPERATOR

Rook-Ceph 操作器是 OpenShift Data Foundation中 Ceph 的 Rook 操作器。Rook 使 Ceph 存储系统能够在 OpenShift Container Platform 上运行。

Rook-Ceph 操作器是一个简单的容器，它会自动引导存储集群并监控存储守护进程，以确保存储集群正常运行。

3.3.1. 组件

Rook-Ceph 操作器管理许多组件，作为 OpenShift Data Foundation 部署的一部分。

Ceph-CSI 驱动程序

操作器创建和更新 CSI 驱动程序，包括每个驱动程序的调配器、RADOS 块设备(RBD)和 Ceph 文件系统(CephFS)，以及每个驱动程序的卷插件**守护进程集**。

Ceph 守护进程

Mons

监视器(mons)为 Ceph 提供核心元数据存储。

OSD

对象存储守护进程(OSD)将数据存储在底层设备上。

Mgr

管理器(mgr)收集指标并为 Ceph 提供其他内部功能。

RGW

RADOS 网关(RGW)为对象存储提供 S3 端点。

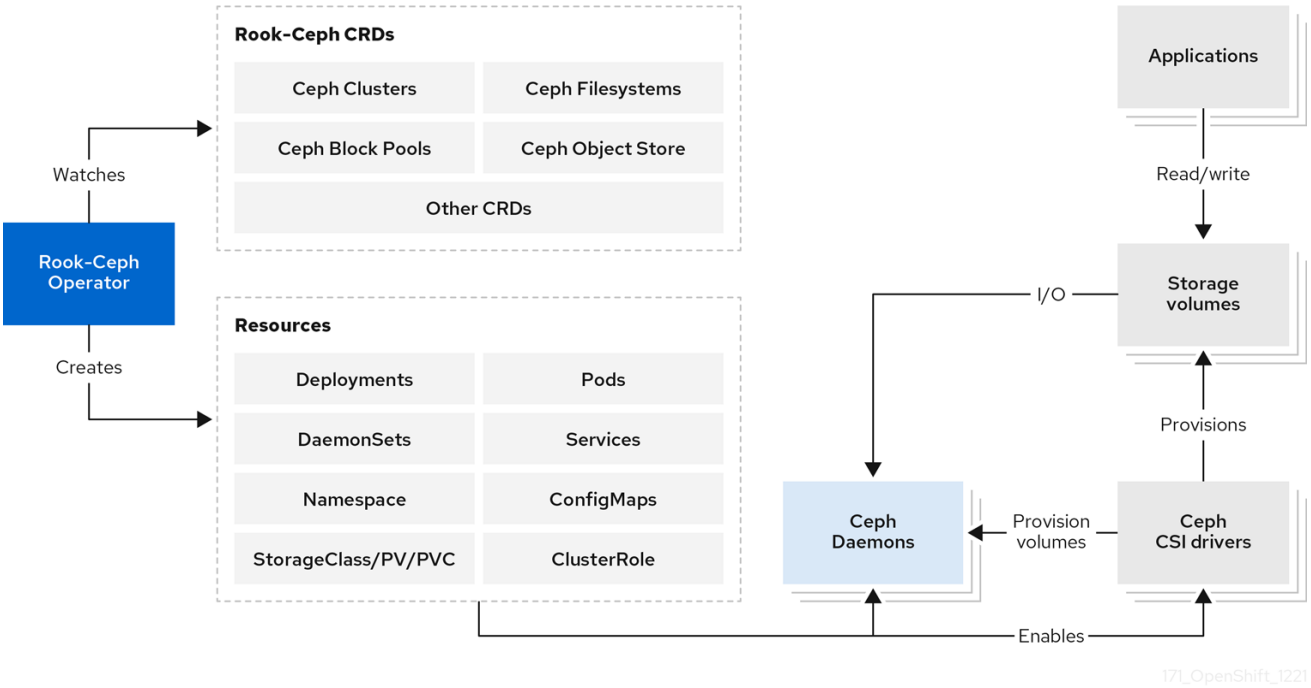
MDS

元数据服务器(MDS)提供 CephFS 共享卷。

3.3.2. 设计图

下图说明了 Ceph Rook 如何与 OpenShift Container Platform 集成。

图 3.3. Rook-Ceph Operator



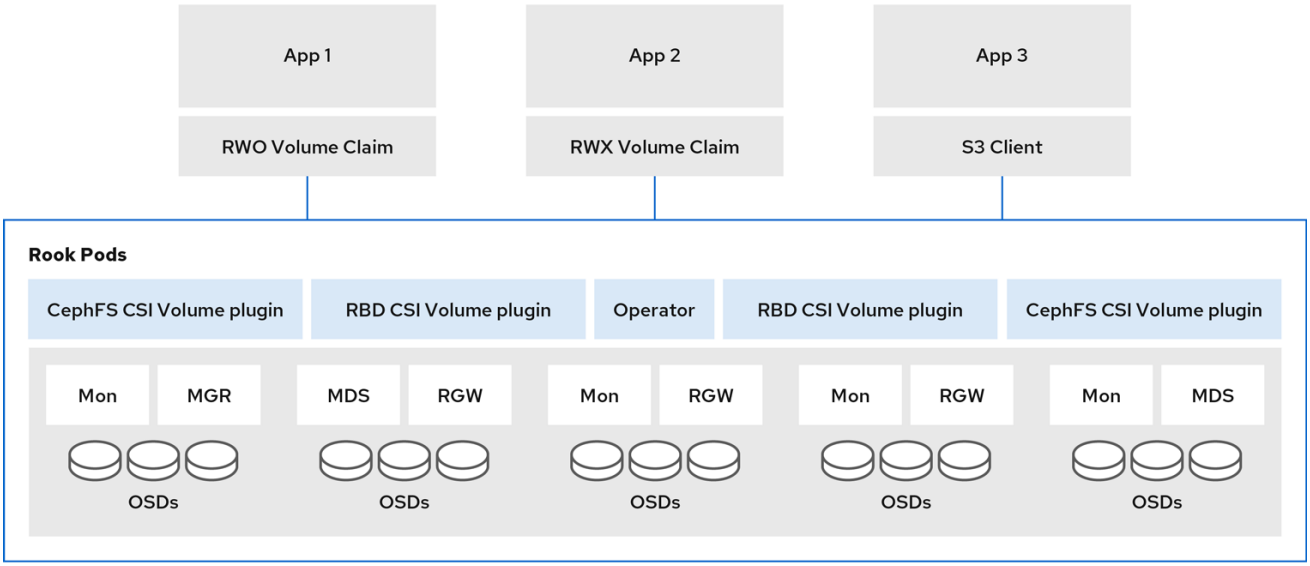
当 Ceph 在 OpenShift Container Platform 集群中运行时，OpenShift Container Platform 应用程序可以挂载由 Rook-Ceph 管理的块设备和文件系统，或者使用 S3/Swift API 进行对象存储。

3.3.3. 职责

Rook-Ceph 操作器是一个引导和监控存储集群的容器。它执行以下功能：

- 自动配置存储组件
- 启动、监控和管理 Ceph 监控容器集和 Ceph OSD 守护进程，以提供 RADOS 存储集群
- 初始化 pod 和其他工件以运行要管理的服务：
  - 池的 CRD
  - 对象存储(S3/Swift)
  - 文件系统
- 监控 Ceph mons 和 OSD，以确保存储仍然可用且正常运行
- 在根据集群大小调整 mon 配置时部署和管理 Cephmonmon 放置
- 监视 API 服务请求的状态更改，并应用更改
- 初始化使用存储所需的 Ceph-CSI 驱动程序
- 自动配置 Ceph-CSI 驱动程序，将存储挂载到 pod

rook-Ceph Operator 架构



171_OpenShift_1221

Rook-Ceph 操作器镜像包含管理集群所需的所有工具。数据路径没有改变。但是，操作器不会公开所有 Ceph 配置。PG 和 crush map 等许多 Ceph 功能都隐藏给用户，在物理资源、池、卷、文件系统和存储桶方面提供更好的用户体验。

3.3.4. Resources

rook-Ceph operator 添加了对在 **openshift-storage** 命名空间中创建的所有资源的所有者引用。卸载集群时，所有者会引用确保资源都已清理。这包括 OpenShift Container Platform 资源，如 **configmaps**, **secrets**, **services**, **deployments**, **daemonsets** 等。

Rook-Ceph 操作器监视 CR，以配置由 OpenShift Data Foundation 决定的设置，其中包括 **CephCluster**、**CephObjectStore**、**CephFilesystem** 和 **CephBlockPool**。

3.3.5. 生命周期

rook-Ceph 操作器管理 Ceph 集群中以下 pod 的生命周期：

rook operator

拥有集群协调的单一 pod。

RBD CSI 驱动程序

- 两个调配器容器集，由一个部署进行管理。
- 每个节点有一个插件 pod，由 **daemonset** 管理。

CephFS CSI 驱动程序

- 两个调配器容器集，由一个部署进行管理。
- 每个节点有一个插件 pod，由 **daemonset** 管理。

Monitors (mons)

三个 mon pod，各自具有自己的部署。

扩展集群

包含五个 mon pod，一个位于仲裁区域中，另一个位于另外两个数据区域中。

Manager (mgr)

集群有一个 mgr pod。

扩展集群

有两个 mgr 容器集（从 OpenShift Data Foundation 4.8 开始），各自位于两个非仲裁区域中。

对象存储守护进程 (OSD)

集群中最初创建至少三个 OSD。扩展集群时，会添加更多 OSD。

元数据服务器(MDS)

CephFS 元数据服务器只有一个 pod。

RADOS 网关(RGW)

Ceph RGW 守护进程只有一个 pod。

3.4. MCG OPERATOR

Multicloud 对象网关(MCG)操作器是 OpenShift Data Foundation 的操作器，以及 OpenShift Data Foundation 操作器和 Rook-Ceph 操作器。MCG operator 上游作为独立操作器提供。

MCG 操作器执行以下主要功能：

- 控制并协调 OpenShift Data Foundation 中的多云对象网关(MCG)组件。
- 管理新的用户资源，如对象 bucket 声明、存储桶类和后备存储。
- 创建默认的"开箱即用"资源。

些配置和信息通过 OpenShift Data Foundation 操作器传递给 MCG 操作器。

3.4.1. 组件

MCG 操作器没有子组件。但是，它由一个协调循环组成，用于由它控制的不同资源。

MCG 操作器具有命令行界面(CLI)，并作为 OpenShift Data Foundation 的一部分提供。它支持创建、删除和查询各种资源。此 CLI 在应用配置前添加了一个输入和状态验证层，这与直接应用 YAML 文件不同。

3.4.2. 职责和资源

MCG Operator 协调并负责自定义资源定义(CRD)和 OpenShift Container Platform 实体。

- 后备存储
- 命名空间存储
- bucket 类
- 对象存储桶声明(OBC)
- NooBaa, pod 有状态设置 CRD
- Prometheus 规则和服务监控

- Pod 横向自动扩展(HPA)

后备存储

客户连接到 MCG 组件的资源。此资源为 MCG 提供在其上保存置备存储桶数据的功能。

根据 OpenShift Container Platform 运行的平台，作为部署的一部分创建默认后备存储。例如，当 OpenShift Container Platform 或 OpenShift Data Foundation 部署在 Amazon Web Services(AWS)上时，它会生成一个默认的后备存储，即 AWS::S3 存储桶。同样，对于 Microsoft Azure，默认后备存储是一个 blob 容器，以此类推。

默认后备存储使用云凭证操作器的 CRD 创建，该 CRD 附带了 OpenShift Container Platform。可添加到 MCG 的后备存储的数量没有限制。bucket 类 CRD 中使用后备存储来定义存储桶的不同策略。参考特定 OpenShift Data Foundation 版本的文档，以确定作为后备存储支持的服务或资源的类型。

命名空间存储

命名空间存储桶中使用的资源。部署期间不会创建默认值。

BucketClass

新调配存储桶的默认或初始策略。在存储桶类中设置以下策略：

放置策略（Placement policy）

指示要附加到存储桶的后备存储并用于写入存储桶的数据。此策略用于数据存储桶和缓存策略，以指示本地缓存放置。放置策略有两种模式：

- 分散。在定义的后备存储间条状数据
- 镜像。在每个后备存储上创建完整的副本

命名空间策略

命名空间存储桶的策略，用于定义用于聚合的资源以及用于写入目标的资源。

缓存策略

这是存储桶的策略，并设置缓存项目的 hub（数据源）和生存时间(TTL)。

部署期间会创建一个默认的 bucket 类，它使用使用默认后备存储的放置策略进行设置。对可添加的 bucket 类数量没有限制。

请参阅特定 OpenShift Data Foundation 版本的文档，以确定受支持的策略类型。

对象存储桶声明(OBC)

启用置备 S3 存储桶的 CRD。使用 MCG 时，OBC 接收一个可选存储桶类来记录存储桶的初始配置。如果没有提供 bucket 类，则使用默认的 bucket 类。

NooBaa, pod 有状态设置 CRD

控制 NooBaa 部署的不同 pod 的内部 CRD，如 DB pod、核心 pod 和端点。此 CRD 不可更改，因为它是内部的。此操作器协调以下实体：

- DB Pod SCC
- Role Binding 和 Service Account，允许 OpenShift Container Platform 和 NooBaa 用户界面间的 SSO 单点登录
- S3 访问的路由

- 由 OpenShift Container Platform 获取并签名的证书，它们是在 S3 路由上设置的

Prometheus 规则和服务监控

这些 CRD 为 Prometheus 和 MCG 支持的警报规则设置提取点。

Pod 横向自动扩展(HPA)

它与 MCG 端点集成。端点容器集根据 CPU 压力（S3 流量的数量）扩展和缩减。

3.4.3. 高可用性

作为操作器，唯一提供的高可用性是 OpenShift Container Platform 重新调度失败的 pod。

3.4.4. 相关日志文件

要排除 NooBaa Operator 的问题，您可以查看以下内容：

- Operator pod 日志，它们也可通过 must-gather 获得。
- 可以通过 must-gather 获得的不同 CRD 或实体及其状态。

3.4.5. 生命周期

在部署 OpenShift Data Foundation 并卸载前，MCG 操作器会运行和协调。

第 4 章 OPENSHIFT DATA FOUNDATION 安装概述

OpenShift Data Foundation 由多个操作器管理的多个组件组成。

4.1. 已安装的 OPERATOR

从 Operator Hub 安装 OpenShift Data Foundation 时，会创建以下四个单独的 Deployment：

- **odf-operator** ：定义 **odf-operator** Pod
- **ocs-operator**：定义 **ocs-operator** Pod，该 Pod 为 **ocs-operator** 和在相同容器中的它的 **metrics-exporter** 运行进程。
- **rook-ceph-operator**: 定义 **rook-ceph-operator** Pod。
- **mcg-operator**: 定义 **mcg-operator** Pod。

这些操作器通过创建由其他操作器监视的客户资源(CR)来独立运行并相互交互。**ocs-operator** 主要负责创建 CR 来配置 Ceph 存储和多云对象网关。**mcg-operator** 有时会创建供其组件使用的 Ceph 卷。

4.2. OPENSHIFT CONTAINER STORAGE 初始化

OpenShift Data Foundation 捆绑包还定义了 OpenShift Container Platform 控制台的外部插件，添加新的屏幕和功能，即控制台中无法提供的新屏幕和功能。此插件在 **odf-console-plugin** Pod 中作为 Web 服务器运行，该插件由安装时由 OLM 创建的 Deployment 管理。

ocs-operator 在创建后会自动创建 **OCSInitialization** CR。在任何时间点上只有一个 **OCSInitialization** CR。它控制的 **ocs-operator** 行为不仅限于单个 **StorageCluster** 的范围，而是只执行一次。当您删除 **OCSInitialization CR** 时，**ocs-operator** 会再次创建它，这可让您重新触发其初始化操作。

OCSInitialization CR 控制以下行为：

SecurityContextConstraints (SCCs)

创建 **OCSInitialization** CR 后，**ocs-operator** 会创建各种 SCC 供组件 Pod 使用。

Ceph Toolbox 部署

您可以使用 **OCSInitialization** 部署用于高级 Ceph 操作的 Ceph 工具箱 Pod。

rook-Ceph Operator 配置

此配置会创建 **rook-ceph-operator-config ConfigMap**，用于监管 **rook-ceph-operator** 行为的整体配置。

4.3. 存储集群创建

OpenShift Data Foundation 操作器本身不提供存储功能，而且必须定义所需的存储配置。

安装 Operator 后，可使用 OpenShift Container Platform 控制台向导或 CLI 创建新的 **StorageCluster**，并使用 **ocs-operator** 协调这个 **StorageCluster**。OpenShift Data Foundation 在每次安装中支持单个 **StorageCluster**。**ocs-operator** 协调忽略了在第一个 CR 之后创建的 **StorageCluster** CR。

OpenShift Data Foundation 允许以下 StorageCluster 配置：

内部

在 Internal 模式下，所有组件都在 OpenShift Container Platform 集群中容器化，并使用在安装向导中根据管理员指定的 **StorageClass** 创建的动态置备持久性卷(PV)。

内部附加

此模式与内部模式类似，但管理员需要定义直接附加到 Ceph 用于其后备存储的群集节点的本地存储设备。另外，管理员需要创建与本地存储 operator 协调的 CR 以提供 **StorageClass**。**ocs-operator** 使用这个 **StorageClass** 作为 Ceph 的后备存储。

外部

在这种模式下，Ceph 组件不会在 OpenShift Container Platform 集群内运行，而是为应用程序可以为其创建 PV 的外部 OpenShift Container Storage 安装提供连接。其他组件根据需要在集群中运行。

MCG 独立

此模式有助于在没有附带的 CephCluster 的情况下安装多云对象网关系统。

找到 **StorageCluster** CR 后，**ocs-operator** 会对其进行验证，并开始创建后续资源来定义存储组件。

4.3.1. 内部模式存储集群

内部和外部附加存储集群的设置过程相同，如下所示：

StorageClasses	创建集群应用用于创建 Ceph 卷的存储类。
SnapshotClasses	创建集群应用用于创建 Ceph 卷快照的卷快照类。
Ceph RGW 配置	创建各种 Ceph 对象 CR，以启用和提供对 Ceph RGW 对象存储端点的访问。
Ceph RBD 配置	创建 **CephBlockPool** CR 以启用 RBD 存储。
CephFS 配置	创建 **CephFilesystem** CR 以启用 CephFS 存储。
Rook-Ceph 配置	创建 **rook-config-override ConfigMap**，以管理底层 Ceph 集群的整体行为。
CephCluster	创建 **CephCluster** CR，以从 **rook-ceph-operator** 触发 Ceph 协调。如需更多信息，请参阅 Rook-Ceph 操作器。
NoobaaSystem	创建 **NooBaa** CR，以触发从 **mcg-operator** 的协调。如需更多信息，请参阅 MCG operator。
作业模板	创建 **OpenShift Template** CR，以定义作业，以运行 OpenShift Container Storage 的管理操作。
Quickstarts	创建 **QuickStart** CR，在 Web 控制台中显示 Quickstart 指南。

4.3.1.1. 集群创建

在 **ocs-operator** 创建 **CephCluster** CR 后，**rook-operator** 根据所需的配置创建 Ceph 集群。**rook-operator** 配置以下组件：

Ceph **mon** 守护进程	在集群中的不同节点上启动三个 Ceph **mon** 守护进程。它们管理 Ceph 集群的核心元数据，它们必须组成大多数仲裁。如果每个 **mon** 的元数据位于云环境中，则由 PV 支持，如果它位于本地存储设备环境中，则由本地主机上的路径支持。
Ceph **mgr** 守护进程	此守护进程已启动，它会为集群收集指标并将其报告到 Prometheus。
Ceph OSD	这些 OSD 根据 **storageClassDeviceSets** 的配置创建。每一 OSD 使用一个存储用户数据的 PV。默认情况下，Ceph 使用 **CRUSH** 算法在不同的 OSD 之间维护三个应用数据副本，以实现高持久性和可用性。
CSI 置备程序	为 RBD 和 **CephFS** 启动这些调配器。当为 OpenShift Container Storage 的存储类请求卷时，请求将定向到 **Ceph-CSI** 驱动程序来置备 Ceph 中的卷。
CSI 卷插件和 **CephFS**	RBD 和 **CephFS** 的 CSI 卷插件在集群中的每一节点上启动。卷插件需要处于运行状态，只要应用需要挂载 Ceph 卷的位置。

配置 **CephCluster** CR 后，Rook 会协调剩余的 Ceph CR 以完成设置：

CephBlockPool	**CephBlockPool** CR 提供 Rook operator 的配置，以便为 RWO 卷创建 Ceph 池。
CephFilesystem	**CephFilesystem** CR 指示 Rook 操作器使用 CephFS 配置共享文件系统，通常用于 RWX 卷。CephFS 元数据服务器(MDS)已启动，以管理共享卷。
CephObjectStore	**CephObjectStore** CR 指示 Rook 操作器使用 RGW 服务配置对象存储
CephObjectStoreUser CR	**CephObjectStoreUser** CR 指示 Rook 操作器将 NooBaa 的对象存储用户配置为使用，发布 **access/private** 密钥和 **CephObjectStore** 端点。

操作器监控 Ceph 健康状况，以确保存储平台健康。如果某个 **mon** 守护进程长时间停机（10 分钟），Rook 会在其位置启动一个新的 **mon**，以便能够完全恢复仲裁。

当 **ocs-operator** 更新 **CephCluster** CR 时，Rook 会立即响应请求的更改，以更新集群配置。

4.3.1.2. NooBaa 系统创建

当创建 NooBaa 系统时，**mcg-operator** 会协调以下内容：

默认 BackingStore

根据部署 OpenShift Container Platform 和 OpenShift Data Foundation 的平台，会创建一个默认的后备存储资源，以便存储桶可以将其用于放置策略。不同的选项如下：

Amazon Web Services(AWS)部署	**mcg-operator** 使用 **CloudCredentialsOperator** (CCO)来 mint 凭证，以便创建新的 AWS::S3 存储桶并在该存储桶上创建 **BackingStore**。

Microsoft Azure 部署	**mcg-operator** 使用 CCO 来 mint 凭证，以便创建新的 Azure Blob 并在该存储桶上创建一个 **BackingStore**。
Google Cloud Platform(GCP)部署	**mcg-operator** 使用 CCO 来 mint 凭证以创建新的 GCP 存储桶，并将在该存储桶上创建一个 BackingStore。
On-prem 部署	如果 RGW 存在，**scg-operator** 会在 RGW 上创建新的**CephUser** 和新存储桶，并在该存储桶基础上创建 **BackingStore**。
以上所述部署都不适用	**mcg-operator** 基于默认存储类创建一个 **pv-pool**，并在该存储桶之上创建一个 **BackingStore**。

默认 BucketClass

创建带有放置策略到默认 **BackingStore** 的 **BucketClass**。

NooBaa pod

以下 NooBaa pod 已创建并启动：

数据库(DB)	一个 Postgres DB 用来保存元数据、统计信息和事件等。然而，它不会保存存储的实际数据。
Core	这是处理配置、后台进程、元数据管理、统计信息等的 pod。
Endpoints	这些 pod 执行与 I/O 相关的实际工作，如重复数据删除和压缩、与不同服务通信以写入和读取数据等。端点与 **HorizonalPodAutoscaler** 集成，并根据现有端点 pod 上观察到的 CPU 使用量增加和减少。

Route

为使用 S3 的应用创建 NooBaa S3 接口的路由。

服务

为 NooBaa S3 接口创建一个服务，供使用 S3 的应用创建。

4.3.2. 外部模式存储集群

对于外部存储集群，**ocs-operator** 的设置过程略有不同。**ocs-operator** 会查找是否存在 **rook-ceph-external-cluster-details ConfigMap**，该 ConfigMap 必须由其他人（管理员或控制台）创建。有关如何创建 **ConfigMap** 的详情，请参考为外部模式创建 OpenShift Data Foundation。然后，**ocs-operator** 会创建部分或全部以下资源，如 **ConfigMap** 中指定的：

外部 Ceph 配置	指定外部 **mon** 的端点的 ConfigMap。
外部 Ceph 凭据 Secret	包含用于连接外部 Ceph 实例的凭据的 Secret。

外部 Ceph StorageClasses	一个或多个 StorageClasses，用于为 RBD、CephFS 和/或 RGW 创建卷。
启用 CephFS CSI 驱动程序	如果指定了 **CephFS** StorageClass，请配置 **rook-ceph-operator** 以部署 **CephFS** CSI Pod。
Ceph RGW 配置	如果指定了 RGW StorageClass，请创建各种 Ceph 对象 CR 来启用和提供对 Ceph RGW 对象存储端点的访问。

创建 **ConfigMap** 中指定的资源后，StorageCluster 创建过程会进行如下：

CephCluster	创建 **CephCluster** CR，从 **rook-ceph-operator** 触发 Ceph 协调（请参阅后续小节）。
SnapshotClasses	创建应用程序用于创建 Ceph 卷快照的 **SnapshotClasses**。
NoobaaSystem	创建 **NooBaa** CR，以从 noobaa-operator 触发协调（请参阅后续部分）。
QuickStarts	创建 **Quickstart** CR，在控制台中显示 Quickstart 指南。

4.3.2.1. 集群创建

当以外部模式创建 **CephCluster** CR 时，Rook operator 会执行以下操作：

- 操作器验证连接可供远程 Ceph 集群使用。连接需要将 **mon** 端点和 secret 导入到本地集群中。
- CSI 驱动程序通过与 Ceph 的远程连接进行配置。在内部模式中配置时，RBD 和 **CephFS** 调配器和卷插件的启动方式与 CSI 驱动程序相似，与 Ceph 的连接正好与 OpenShift 集群外部。
- 定期监控地址更改，并相应地更新 Ceph-CSI 配置。

4.3.2.2. NooBaa 系统创建

当创建 NooBaa 系统时，**mcg-operator** 会协调以下内容：

默认 BackingStore

根据部署 OpenShift Container Platform 和 OpenShift Data Foundation 的平台，会创建一个默认的后备存储资源，以便存储桶可以将其用于放置策略。不同的选项如下：

Amazon Web Services(AWS)部署	**mcg-operator** 使用 **CloudCredentialsOperator** (CCO)来 mint 凭证，以便创建新的 AWS::S3 存储桶并在该存储桶上创建 **BackingStore**。
Microsoft Azure 部署	**mcg-operator** 使用 CCO 来 mint 凭证，以便创建新的 Azure Blob 并在该存储桶上创建一个 **BackingStore**。

Google Cloud Platform(GCP)部署	**mcg-operator** 使用 CCO 来 mint 凭证以创建新的 GCP 存储桶，并将在该存储桶上创建一个 BackingStore。
On-prem 部署	如果 RGW 存在，**scg-operator** 会在 RGW 上创建新的**CephUser** 和新存储桶，并在该存储桶基础上创建 **BackingStore**。
以上所述部署都不适用	**mcg-operator** 基于默认存储类创建一个 **pv-pool**，并在该存储桶之上创建一个 **BackingStore**。

默认 BucketClass

创建带有放置策略到默认 **BackingStore** 的 **BucketClass**。

NooBaa pod

以下 NooBaa pod 已创建并启动：

数据库(DB)	一个 Postgres DB 用来保存元数据、统计信息和事件等。然而，它不会保存存储的实际数据。
Core	这是处理配置、后台进程、元数据管理、统计信息等的 pod。
Endpoints	这些 pod 执行与 I/O 相关的实际工作，如重复数据删除和压缩、与不同服务通信以写入和读取数据等。端点与 **HorizonalPodAutoscaler** 集成，并根据现有端点 pod 上观察到的 CPU 使用量增加和减少。

Route

为使用 S3 的应用创建 NooBaa S3 接口的路由。

服务

为 NooBaa S3 接口创建一个服务，供使用 S3 的应用创建。

4.3.3. MCG 独立存储集群

在这种模式中，不会创建 CephCluster。相反，会使用默认值创建一个 NooBaa 系统 CR，以利用 OpenShift Container Platform 中的预先存在的 StorageClasses。

4.3.3.1. NooBaa 系统创建

当创建 NooBaa 系统时，**mcg-operator** 会协调以下内容：

默认 BackingStore

根据部署 OpenShift Container Platform 和 OpenShift Data Foundation 的平台，会创建一个默认的后备存储资源，以便存储桶可以将其用于放置策略。不同的选项如下：

Amazon Web Services(AWS)部署	**mcg-operator** 使用 **CloudCredentialsOperator** (CCO)来 mint 凭证，以便创建新的 AWS::S3 存储桶并在该存储桶上创建 **BackingStore**。
Microsoft Azure 部署	**mcg-operator** 使用 CCO 来 mint 凭证，以便创建新的 Azure Blob 并在该存储桶上创建一个 **BackingStore**。
Google Cloud Platform(GCP)部署	**mcg-operator** 使用 CCO 来 mint 凭证以创建新的 GCP 存储桶，并将在该存储桶上创建一个 BackingStore。
On-prem 部署	如果 RGW 存在，**scg-operator** 会在 RGW 上创建新的**CephUser** 和新存储桶，并在该存储桶基础上创建 **BackingStore**。
以上所述部署都不适用	**mcg-operator** 基于默认存储类创建一个 **pv-pool**，并在该存储桶之上创建一个 **BackingStore**。

默认 BucketClass

创建带有放置策略到默认 **BackingStore** 的 **BucketClass**。

NooBaa pod

以下 NooBaa pod 已创建并启动：

数据库(DB)	一个 Postgres DB 用来保存元数据、统计信息和事件等。然而，它不会保存存储的实际数据。
Core	这是处理配置、后台进程、元数据管理、统计信息等的 pod。
Endpoints	这些 pod 执行与 I/O 相关的实际工作，如重复数据删除和压缩、与不同服务通信以写入和读取数据等。端点与 **HorizonalPodAutoscaler** 集成，并根据现有端点 pod 上观察到的 CPU 使用量增加和减少。

Route

为使用 S3 的应用创建 NooBaa S3 接口的路由。

服务

为 NooBaa S3 接口创建一个服务，供使用 S3 的应用创建。

4.3.3.2. StorageSystem 创建

作为 StorageCluster 创建的一部分，**odf-operator** 会自动创建一个对应的 **StorageSystem** CR，它将 StorageCluster 公开给 OpenShift Data Foundation。

第 5 章 OPENSHIFT DATA FOUNDATION 升级概述

作为由 Operator Lifecycle Manager(OLM)管理的操作器捆绑包，OpenShift Data Foundation 会利用其操作器来执行通过 **ClusterServiceVersion** (CSV)CR 安装和升级产品的高级任务。

5.1. 升级工作流

OpenShift Data Foundation 可识别两种类型的升级：Z-stream 发行升级和次版本升级。虽然这两个升级路径的用户界面工作流并不完全相同，但生成的行为相当相似。其区别如下：

对于 Z-stream 发行版本，OCS 将在 **redhat-operators CatalogSource** 中发布一个新捆绑包。OLM 将检测到这一点，并为新 CSV 创建 **InstallPlan** 来替换现有 CSV。订阅批准策略（无论是 Automatic 或 Manual）将决定 OLM 是否继续协调，或等待管理员批准。

对于次版本发行版本，OpenShift Container Storage 也会在 **redhat-operators CatalogSource** 中发布一个新的捆绑包。不同之处在于，这个捆绑包将成为新频道的一部分，频道升级不是自动的。管理员必须显式选择新发行频道。完成后，OLM 将检测到这一点，并为新 CSV 创建 **InstallPlan** 来替换现有的 CSV。由于频道切换是一个手动操作，因此 OLM 将自动启动协调。

从现在起，升级过程是相同的。

5.2. CLUSTERSERVICEVERSION 协调

当 OLM 检测到批准的 **InstallPlan** 时，它开始协调 CSV 的过程。一般来说，它根据新 spec 更新 operator 资源，验证新 CSV 安装正确，然后删除旧的 CSV。升级过程将把更新推送到 Operator Deployments，这将触发 Operator Pod 重启使用新 CSV 中指定的镜像。

注意

虽然可以对给定 CSV 进行更改并将这些更改传播到相关资源，但在升级到新 CSV 时，所有自定义更改都将丢失，因为新 CSV 将根据其未更改的 spec 创建。

5.3. OPERATOR 协调

此时，OpenShift 数据基础操作对象的协调继续进行，如 OpenShift 数据基础安装概述中所述。Operator 将确保其预期配置中存在所有相关资源，如面向用户的资源（如 **StorageCluster**）中指定的。