Red Hat OpenShift Data Foundation 4.15

管理并分配存储资源

有关如何将存储分配到核心服务和托管应用程序的说明，包括快照和克隆。

Last Updated: 2024-03-07

Red Hat OpenShift Data Foundation 4.15 管理并分配存储资源

有关如何将存储分配到核心服务和托管应用程序的说明，包括快照和克隆。

法律通告

摘要

本文档介绍如何在 Red Hat OpenShift Data Foundation 为核心服务和托管应用程序分配存储。

目录

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 概述

阅读本文档，了解如何在 Red Hat OpenShift Data Foundation 中创建、配置和分配存储到核心服务或托管应用程序。

- 第 2 章 *存储类* 显示如何创建自定义存储类。
- 第 3 章 *块池* 为您提供有关如何创建、更新和删除块池的信息。
- 第 4 章 *为 OpenShift Container Platform 服务配置存储* 演示了如何将 OpenShift Data Foundation 用于 OpenShift Container Platform 核心服务。
- 第 6 章 *使用 OpenShift Data Foundation 支持 OpenShift Container Platform 应用程序* 提供有关如何配置 OpenShift 容器平台应用以使用 OpenShift Data Foundation 的信息。
- 将文件和对象存储添加到现有外部 OpenShift Data Foundation 集群
- 第 8 章 *如何在 Red Hat OpenShift Data Foundation 中使用专用 worker 节点* 提供有关如何为 Red Hat OpenShift Data Foundation 使用专用工作程序节点的信息。
- 第 9 章 *管理持久性卷声明* 提供有关管理持久卷声明请求和自动完成这些请求的信息。
- 第 10 章 *在目标卷中重新声明空间* 演示了如何回收实际可用的存储空间。
- 第 11 章 *卷快照* 向您演示如何创建、恢复和删除卷快照。
- 第 12 章 *卷克隆* 显示如何创建卷克隆。
- 第 13 章 *管理容器存储接口(CSI)组件放置* 提供有关设置容忍度以在节点上调出容器存储接口组件的信息。

第 2 章 存储类

OpenShift Data Foundation Operator 根据使用的平台安装默认存储类。这个默认存储类由 Operator 所有和控制，且无法删除或修改。但是，您可以创建客户存储类来使用其他存储资源或为应用提供不同的行为。

注意

外部模式 OpenShift Data Foundation 集群不支持自定义存储类。

2.1. 创建存储类和池

您可以使用现有池创建存储类，也可以在创建存储类时为存储类创建新池。

先决条件

- 确保您已登录到 OpenShift Container Platform Web 控制台，并且 OpenShift Data Foundation 集群处于 **Ready** 状态。

流程

1. 点 **Storage → StorageClasses**。
2. 点 **Create Storage Class**。
3. 输入存储类 **Name** 和 **Description**。
4. **Reclaim Policy** 设置为 **Delete** 作为默认选项。使用这个设置。
   如果您在存储类中将重新声明策略改为 **Retain**，则持久性卷(PV)会处于 **Released** 状态，即使在删除持久性卷声明(PVC)后也是如此。
5. **卷绑定模式**设置为 **WaitForConsumer** 作为默认选项。
   如果您选择 **Immediate** 选项，则创建 PVC 时会立即创建 PV。
6. 选择 **RBD** 或 **CephFS Provisioner**，作为调配持久卷的插件。
7. 为您的工作负载 **选择一个** 存储系统。
8. 从列表中选择现有**存储池**，或创建新池。

注意

双向复制数据保护策略仅支持非默认 RBD 池。通过创建额外的池，可以使用双向复制。要了解副本 2 池的 Data Availability 和 Integrity 注意事项，请参阅知识库文章。

创建新池

a. 单击 **Create New Pool**。

b. 输入 **池名称**。

c. 选择 **2-way-Replication** 或 **3-way-Replication**作为数据保护策略。

d. 如果需要压缩数据，选择**启用压缩**。
启用压缩可能会影响应用程序的性能，在已压缩或加密的数据时可能会证明无效。在启用压缩之前写入的数据不会压缩。

e. 单击 **Create** 以创建新存储池。

f. 创建池后，单击 **Finish**。

9. 可选：选择**启用加密**复选框。

10. 点 **Create** 创建存储类。

2.2. 用于持久性卷加密的存储类

持久性卷(PV)加密可确保租户（应用程序）之间的隔离和保密性。在使用 PV 加密前，您必须为 PV 加密创建一个存储类。持久卷加密仅可用于 RBD PV。

OpenShift Data Foundation 支持在 HashiCorp Vault 和 Thales CipherTrust Manager 中存储加密密码短语。您可以创建加密的存储类，使用外部密钥管理系统(KMS)进行持久性卷加密。在创建存储类前，您需要配置对 KMS 的访问。

注意

对于 PV 加密，您必须具有有效的 Red Hat OpenShift Data Foundation 高级订阅。如需更多信息，请参阅 OpenShift Data Foundation 订阅中的知识库文章。

2.2.1. 访问密钥管理系统 (KMS) 的配置

取决于您的具体用例，需要使用以下方法之一配置对 KMS 的访问：

- 使用 **vaulttoken** 允许用户使用令牌进行身份验证
- 使用 **Thales CipherTrust Manager** ：使用密钥管理互操作性协议 (KMIP)
- 使用 **vaulttenantsa** （技术预览）：允许用户使用 **serviceaccounts** 通过 **Vault** 进行身份验证

重要

使用 **vaulttenantsa** 访问 KMS 是一项技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需更多信息，请参阅技术预览功能支持范围。

2.2.1.1. 使用 vaulttoken 配置对 KMS 的访问

先决条件

- OpenShift 数据基础集群处于 **Ready** 状态。
- 在外部密钥管理系统 (KMS) 上，
  - 确保存在具有令牌的策略，并且启用了 **Vault** 中的键值后端路径。

- 确保您在 **Vault** 服务器上使用签名证书。

流程

在租户的命名空间中创建一个 secret。

1. 在 OpenShift Container Platform web 控制台中进入 **Workloads → Secrets**。
2. 点 **Create → Key/value secret**。
3. 输入 **Secret Name** 作为 **ceph-csi-kms-token**。
4. 输入 **Key** 作为 **token**。
5. 输入 **Value**。
   它是来自 Vault 的令牌。您可以单击 **Browse** 来选择并上传含有令牌的文件，或者直接在文本框中输入令牌。
6. 点 **Create**。

注意

只有在所有使用 **ceph-csi-kms-token** 的加密 PVC 已被删除后，才能删除令牌。

2.2.1.2. 使用 Thales CipherTrust Manager配置对 KMS 的访问

先决条件

1. 如果 KMIP 客户端不存在，请创建一个。在用户界面中，选择 **KMIP → Client Profile → Add Profile**。
   a. 在创建配置集的过程中，将 **CipherTrust** 用户名添加到 **Common Name** 字段中。
2. 通过进入到 **KMIP → Registration Token → New Registration Token** 来创建令牌。为下一步复制令牌。
3. 要注册客户端，请进入到 **KMIP → Registered Clients → Add Client**。指定**名称**。粘贴上一步中的**注册令牌**，然后点**保存**。
4. 点 **Save Private Key** 和 **Save Certificate** 下载私钥和客户端证书。
5. 要创建新的 KMIP 接口，请进入到 **Admin Settings → Interfaces → Add Interface**。
   a. 选择 **KMIP Key Management Interoperability Protocol** 并点 **Next**。
   b. 选择一个空闲**端口**。
   c. 为 **Network Interface** 选择 **all**。
   d. 为 **Interface Mode** 选择 **TLS, verify client cert, user name taken from client cert, auth request is optional**。
   e. （可选）您可以在密钥被删除时启用硬删除以删除元数据和材料。它默认是禁用的。
   f. 选择要使用的 CA，然后点 **Save**。

6. 要获取服务器 CA 证书，请点新创建的界面右侧的 Action 菜单(⋮)，然后点 **Download Certificate**。

流程

1. 要创建一个密钥以充当 storageclass 加密的 KEK（Key Encryption Key），请按照以下步骤执行：
   a. 进入到 **Keys → Add Key**。
   b. 输入 **Key Name**。
   c. 分别将 **Algorithm** 和 **Size** 设置为 **AES** 和 **256**。
   d. 启用 **Create a key in Pre-Active state**，并设置激活的日期和时间。
   e. 确保在 **Key Usage** 下启用了 **Encrypt** 和 **Decrypt**。
   f. 复制新创建的密钥的 ID，以在部署过程中用作唯一标识符。

2.2.1.3. 使用 vaulttenantsa 配置对 KMS 的访问

先决条件

- OpenShift 数据基础集群处于 **Ready** 状态。
- 在外部密钥管理系统 (KMS) 上，
  - 确保策略存在，并且已启用 Vault 中的键值后端路径。
  - 确保您在 Vault 服务器上使用签名的证书。
- 在租户命名空间中创建以下 serviceaccount，如下所示：

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: ServiceAccount
metadata:
   name: ceph-csi-vault-sa
EOF
```

流程

在 OpenShift Data Foundation 使用 **Vault** 进行身份验证之前，您需要配置 Kubernetes 身份验证方法。以下说明创建并配置 **serviceAccount**、**ClusterRole** 和 **ClusterRoleBinding**，以允许 OpenShift Data Foundation 使用 **Vault** 进行身份验证。

1. 将以下 YAML 应用到您的 Openshift 集群：

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: rbd-csi-vault-token-review
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
```

```
metadata:
  name: rbd-csi-vault-token-review
rules:
  - apiGroups: ["authentication.k8s.io"]
    resources: ["tokenreviews"]
    verbs: ["create", "get", "list"]

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rbd-csi-vault-token-review
subjects:
  - kind: ServiceAccount
    name: rbd-csi-vault-token-review
    namespace: openshift-storage
roleRef:
  kind: ClusterRole
  name: rbd-csi-vault-token-review
  apiGroup: rbac.authorization.k8s.io
```

2. 为 serviceaccount 令牌和 CA 证书创建 secret。

```
$ cat <<EOF | oc create -f -
apiVersion: v1
kind: Secret
metadata:
  name: rbd-csi-vault-token-review-token
  namespace: openshift-storage
  annotations:
    kubernetes.io/service-account.name: "rbd-csi-vault-token-review"
type: kubernetes.io/service-account-token
data: {}
EOF
```

3. 从 secret 获取令牌和 CA 证书。

```
$ SA_JWT_TOKEN=$(oc -n openshift-storage get secret rbd-csi-vault-token-review-token -o
jsonpath="{.data['token']}" | base64 --decode; echo)
$ SA_CA_CRT=$(oc -n openshift-storage get secret rbd-csi-vault-token-review-token -o
jsonpath="{.data['ca\.crt']}" | base64 --decode; echo)
```

4. 检索 OpenShift 集群端点。

```
$ OCP_HOST=$(oc config view --minify --flatten -o jsonpath="{.clusters[0].cluster.server}")
```

5. 使用前面步骤中收集的信息在 Vault 中设置 kubernetes 身份验证方法，如下所示：

```
$ vault auth enable kubernetes
$ vault write auth/kubernetes/config \
        token_reviewer_jwt="$SA_JWT_TOKEN" \
        kubernetes_host="$OCP_HOST" \
        kubernetes_ca_cert="$SA_CA_CRT"
```

6. 在 Vault 中为租户命名空间创建一个角色：

```
$ vault write "auth/kubernetes/role/csi-kubernetes" bound_service_account_names="ceph-csi-vault-sa" bound_service_account_namespaces=<tenant_namespace> policies=<policy_name_in_vault>
```

csi-kubernetes 是 OpenShift Data Foundation 在 Vault 中寻找的默认角色名称。OpenShift Data Foundation 集群中租户命名空间中的默认服务帐户名称为 **ceph-csi-vault-sa**。这些默认值可以通过在租户命名空间中创建 ConfigMap 来覆盖。

有关覆盖默认名称的更多信息，请参阅使用租户 ConfigMap 覆盖 Vault 连接详情 。

YAML 示例

- 要创建一个使用 **vaulttenantsa** 方法进行 PV 保护的存储类，您必须编辑现有的 ConfigMap 或创建名为 **csi-kms-connection-details** 的配置映射，它将保存建立与 Vault 连接所需的所有信息。以下提供的 yaml 示例可用于更新或创建 **csi-kms-connection-detail** ConfigMap：

```
apiVersion: v1
data:
  vault-tenant-sa: |-
    {
      "encryptionKMSType": "vaulttenantsa",
      "vaultAddress": "<https://hostname_or_ip_of_vault_server:port>",
      "vaultTLSServerName": "<vault TLS server name>",
      "vaultAuthPath": "/v1/auth/kubernetes/login",
      "vaultAuthNamespace": "<vault auth namespace name>"
      "vaultNamespace": "<vault namespace name>",
      "vaultBackendPath": "<vault backend path name>",
      "vaultCAFromSecret": "<secret containing CA cert>",
      "vaultClientCertFromSecret": "<secret containing client cert>",
      "vaultClientCertKeyFromSecret": "<secret containing client private key>",
      "tenantSAName": "<service account name in the tenant namespace>"
    }
metadata:
  name: csi-kms-connection-details
```

encryptionKMSType	设置为 **vaulttenantsa**，以使用服务帐户与 vault 进行身份验证。
vaultAddress	使用端口号的 vault 服务器的主机名或 IP 地址。
vaultTLSServerName	（可选） vault TLS 服务器名称
vaultAuthPath	（可选）在 Vault 中启用 kubernetes auth 方法的路径。默认路径为 **kubernetes**。如果在 **kubernetes** 之外的其他路径中启用了 auth 方法，则需要将此变量设置为 **"/v1/auth/<path>/login"**。

vaultAuthNamespace	（可选）启用 kubernetes auth 方法的 Vault 命名空间。
vaultNamespace	（可选）用于存储密钥的后端路径存在的 Vault 命名空间
vaultBackendPath	保存加密密钥的 Vault 中的后端路径
vaultCAFromSecret	包含来自 Vault 的 CA 证书的 OpenShift Data Foundation 集群中的 secret
vaultClientCertFromSecret	包含来自 Vault 的客户端证书的 OpenShift Data Foundation 集群中的 secret
vaultClientCertKeyFromSecret	包含来自 Vault 的客户端私钥的 OpenShift Data Foundation 集群中的 secret
tenantSAName	（可选）租户命名空间中的服务帐户名称。默认值为 **ceph-csi-vault-sa**。如果要使用其他名称，则必须相应地设置此变量。

2.2.2. 为持久性卷加密创建存储类

先决条件

根据您的用例，您必须确保为以下之一配置对 KMS 的访问：

- 使用 **vaulttokens** ：确保配置访问权限，如 使用 **vaulttoken**配置对 KMS 的访问所述
- 使用 **vaulttenantsa** （技术预览）：确保配置访问权限，如 使用 **vaulttenantsa**配置对 KMS 的访问所述
- 使用 Thales CipherTrust Manager (using KMIP)：确保配置访问权限，如 使用 Thales CipherTrust Manager 配置对 KMS 的访问所述

流程

1. 在 OpenShift Web 控制台中，进入 **Storage → StorageClasses**。
2. 点 **Create Storage Class**。
3. 输入存储类 **Name** 和 **Description**。
4. 为 **Reclaim Policy** 选择 **Delete** 或 **Retain**。默认情况下，选择 **Delete**。
5. 选择 **Immediate** 或 **WaitForFirstConsumer** 作为**卷绑定模式**。**WaitForConsumer** 设置为默认选项。
6. 选择 **RBD Provisioner openshift-storage.rbd.csi.ceph.com**，这是用于调配持久卷的插件。

7. 选择 **存储池**，其中卷数据将存储到列表中或创建新池。

8. 选中 **Enable encryption** 复选框。有两个选项可用来设置 KMS 连接详情：

   - **Select existing KMS connection**: 从下拉列表中选择现有 KMS 连接。该列表填充自 **csi-kms-connection-details** ConfigMap 中的连接详情。

     a. 从下拉菜单中选择 **Provider**。

     b. 从列表中选择给定供应商的 **Key service**。

   - **创建新的 KMS 连接** ：这仅适用于 **vaulttoken** 和 **Thales CipherTrust Manager (using KMIP)**。

     a. 选择 **Key Management Service Provider**。

     b. 如果将 **Vault** 选为 **Key Management Service Provider**，请按照以下步骤执行：

        i. 输入唯一的**连接名称** , Vault 服务器的主机**地址**（'https://<hostname 或 ip>'），端口号和**令牌**。

        ii. 展开 **Advanced Settings**，以根据您的 **Vault** 配置输入其他设置和证书详情：

            A. 在 **后端路径**中输入为 OpenShift Data Foundation 专用且唯一的 Key Value secret 路径。

            B. （可选）输入 **TLS 服务器名称**和 **Vault Enterprise 命名空间**。

            C. 上传对应的 PEM 编码证书文件，以提供 **CA 证书**、**客户端证书**和**客户端私钥**。

            D. 点 **Save**。

     c. 如果选择了 **Thales CipherTrust Manager （using KMIP）** 作为 **Key Management Service Provider**，请按照以下步骤执行：

        i. 输入一个唯一的**连接名称**。

        ii. 在 **Address** 和 **Port** 部分中，输入 Thales CipherTrust Manager 的 IP 以及在其中启用了 KMIP 接口的端口。例如，**Address**: 123.34.3.2, **Port**: 5696。

        iii. 上传 **客户端证书**、**CA 证书**和 **客户端私钥**。

        iv. 输入在上面生成的用于加密和解密的密钥的**唯一标识符**。

        v. **TLS Server** 字段是可选的，并在没有 KMIP 端点的 DNS 条目时使用。例如，**kmip_all_<port>.ciphertrustmanager.local**。

     d. 点 **Save**。

     e. 点 **Create**。

9. 如果 HashiCorp Vault 设置不允许自动检测后端路径使用的 Key/Value(KV)secret 引擎 API 版本，编辑 ConfigMap 以添加 **vaultBackend** 参数。

注意

vaultBackend 是一个可选参数，添加到 configmap 中，以指定与后端路径关联的 KV secret 引擎 API 版本。确保值与为后端路径设置的 KV secret 引擎 API 版本匹配，否则可能会导致持久性卷声明(PVC)创建过程中失败。

a. 识别新创建的存储类使用的 encryptionKMSID。

   i. 在 OpenShift Web 控制台中，导航到 **Storage → Storage Classes**。

   ii. 点 **Storage class** name → **YAML** 标签页。

   iii. 捕获存储类使用的 **encryptionKMSID**。
   Example:

```
encryptionKMSID: 1-vault
```

b. 在 OpenShift Web 控制台中，导航到 **Workloads → ConfigMaps**。

c. 要查看 KMS 连接详情，请单击点击 **csi-kms-connection-details**。

d. 编辑 ConfigMap。

   i. 点击 Action 菜单 **(⋮) → Edit ConfigMap**。

   ii. 根据之前标识的 **encryptionKMSID** 配置的后端，添加 **vaultBackend** 参数。
   您可以为 KV secret engine API 版本 1 分配 **kv**，为 KV secret engine API 版本 2 分配 **kv-v2**。

   Example:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: csi-kms-connection-details
[...]
data:
  1-vault: |-
    {
      "encryptionKMSType": "vaulttokens",
      "kmsServiceName": "1-vault",
      [...]
      "vaultBackend": "kv-v2"
    }
  2-vault: |-
    {
      "encryptionKMSType": "vaulttenantsa",
      [...]
      "vaultBackend": "kv"
    }
```

   iii. 点 Save

后续步骤

- 存储类可用于创建加密的持久性卷。如需更多信息，请参阅管理持久性卷声明。

重要

红帽与技术合作伙伴合作，将本文档作为为客户提供服务。但是，红帽不为 HashiCorp 产品提供支持。有关此产品的技术协助，请联系 HashiCorp。

2.2.2.1. 使用租户 ConfigMap 覆盖 Vault 连接详情

可以通过在 Openshift 命名空间中创建 ConfigMap 来为每个租户重新配置 Vault 连接详情，其配置选项与 **openshift-storage** 命名空间中的 **csi-kms-connection-details** ConfigMap 中设置的值不同。ConfigMap 需要位于租户命名空间中。租户命名空间中的 ConfigMap 中的值将覆盖在该命名空间中创建的加密持久性卷的 **csi-kms-connection-details** ConfigMap 中设置的值。

流程

1. 确保您位于租户命名空间中。
2. 点 **Workloads → ConfigMaps**。
3. 点 **Create ConfigMap**。
4. 以下是 yaml 示例。给定租户命名空间要覆盖的值可以在 **data** 部分下指定，如下所示：

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: ceph-csi-kms-config
data:
  vaultAddress: "<vault_address:port>"
  vaultBackendPath: "<backend_path>"
  vaultTLSServerName: "<vault_tls_server_name>"
  vaultNamespace: "<vault_namespace>"
```

5. 编辑 yaml 后，点 **Create**。

2.3. 带有单一副本的存储类

您可以使用应用程序使用单个副本创建存储类。这可避免冗余数据副本，并允许在应用程序级别上进行弹性管理。

警告

启用这个功能会创建一个没有数据复制的副本池，如果应用程序本身没有复制，这会增加数据丢失、数据崩溃和潜在的系统不稳定的风险。如果有任何 OSD 丢失，此功能需要非常干扰的步骤才能恢复。所有请求都可能会丢失其数据，在出现故障 OSD 时必须重新创建数据。

流程

使用以下命令启用单个副本功能：

```
$ oc patch storagecluster ocs-storagecluster -n openshift-storage --type json --patch '[\{ "op": "replace", "path": "/spec/managedResources/cephNonResilientPools/enable", "value": true }]'
```

2.3.1. 在 OSD 从单个副本丢失后恢复

在使用副本 1 时，具有单个副本的存储类时，当 OSD 丢失时，可以保证数据丢失。

流程

按照以下步骤，使应用程序在数据丢失后再次运行。

1. 查找处于 **Error** 状态或 **CrashLoopBackoff** 状态的 OSD pod：

   ```
   $ oc get pods -nopenshift-storage -l app=rook-ceph-osd  | grep 'CrashLoopBackOff\|Error'
   ```

2. 识别具有故障 OSD 的 replica-1 池。

   a. 识别运行故障 OSD 的节点：

   ```
   failed_osd_id=0 #replace with the ID of the failed OSD
   ```

   b. 识别运行故障 OSD 的该节点的区域：

   ```
   failure_domain=$(oc get storageclass ocs-storagecluster-ceph-non-resilient-rbd -o yaml | grep domainLabel)
   ```

   ```
   domainLabel=$”(oc get pods rook-ceph-osd-$failed_osd_id -o yaml | grep topology-location-$failure_domain:)”
   ```

   输出显示池，例如：

   ```
   poolName= "ocs-storage cluster-ceph block pool-$domaiLabel”
   ```

   其中 **$domaiLabel** 是 zoneName。

3. 删除 replica-1 池。

   a. 连接到 toolbox pod:

   ```
   toolbox=$(kubectl get pod -l app=rook-ceph-tools -noperator-namespace -o jsonpath='{.items[*].metadata.name}')

   oc rsh $toolbox -noperator-namespace
   ```

   b. 删除 replica-1 池：

   ```
   ceph osd pool rm replica1-pool-name replica1-pool-name yes-I-really-really-mean-it
   ```

4. 缩减故障 OSD pod 的部署：

   -

```
failed_osd_id=0 #replace with the ID of the failed OSD
oc scale deployment -nopenshift-storage rook-ceph-osd-$failed_osd_id --replicas=0
```

5. 清除上面标识的 OSD。在 替换设备 指南中的根据您的平台，使用"替换操作或失败的存储设备"中的步骤。

6. 对于使用 LSO 的平台，Wipe 或 替换磁盘。

7. 重启 rook-ceph Operator：

```
$ oc delete pod -l rook-ceph-operator -nopenshift-storage
```

8. 在 avaialbity 区域中重新创建任何受影响的应用程序，以使用具有相同名称的新池。

第 3 章 块池

OpenShift Data Foundation Operator 根据使用的平台会安装一组默认存储类。这些默认存储池由操作器拥有和控制，且无法删除或修改。在 OpenShift Container Platform 中，您可以创建多个自定义存储池，它们映射到提供以下功能的存储类：

- 使具有自身高可用性的应用能够使用具有两个副本的持久卷，从而可能提高应用性能。
- 使用启用了压缩的存储类为持久性卷声明节省空间。

注意

外部模式 OpenShift Data Foundation 集群不支持多个块池。

3.1. 创建一个块池

先决条件

- 您需要以集群管理员身份登录 OpenShift Container Platform Web 控制台。

流程

1. 点 **Storage → Data Foundation**。
2. 在 **Storage Systems** 选项卡中，选择 storage 系统，然后单击 **BlockPools** 选项卡。
3. 点 **Create Block Pool**。
4. 输入 **池名称**。

注意

默认的池不支持使用双向复制数据保护策略。但是，如果您创建了额外的池，则可以使用双向复制。

5. 为**数据保护策略**选择**双向复制**或**三向复制**。
6. 可选：如果您需要压缩数据，请选择 **启用压缩** 复选框。
   启用压缩可能会影响应用程序的性能，在已压缩或加密的数据时可能会证明无效。在启用压缩前写入的数据不会压缩。
7. 点 **Create**。

3.2. 更新现有池

先决条件

- 您需要以集群管理员身份登录 OpenShift Container Platform Web 控制台。

流程

1. 点 **Storage → Data Foundation**。

2. 在 **Storage Systems** 选项卡中，选择 storage 系统，然后单击 **BlockPools**。
3. 点击您要更新的池末尾的 Action Menu(⋮)。
4. 单击 **Edit Block Pool**。
5. 修改表单详情，如下所示：

注意

默认的池不支持使用双向复制数据保护策略。但是，如果您创建了额外的池，则可以使用双向复制。

   a. 将**数据保护策略**更改为双向复制或三向复制。

   b. 启用或禁用压缩选项。
   启用压缩可能会影响应用程序的性能，在已压缩或加密的数据时可能会证明无效。在启用压缩前写入的数据不会压缩。

6. 点击 **Save**。

3.3. 删除池

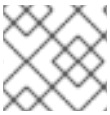
使用此流程删除 OpenShift Data Foundation 中的池。

先决条件

- 您需要以集群管理员身份登录 OpenShift Container Platform Web 控制台。

流程

1. .点 **Storage → Data Foundation**。
2. 在 **Storage Systems** 选项卡中，选择 storage 系统，然后单击 **BlockPools** 选项卡。
3. 点击您要删除的池末尾的 Action Menu(⋮)。
4. 单击 **Delete Block Pool**。
5. 单击 **Delete** 确认删除池。



注意

当池绑定到 PVC 时，无法删除它。您必须分离所有资源，然后才能执行此活动。

第 4 章 为 OPENSHIFT CONTAINER PLATFORM 服务配置存储

您可以使用 OpenShift Data Foundation 为 OpenShift Container Platform 服务提供存储，如下所示：

- OpenShift 镜像 registry
- OpenShift 监控
- OpenShift 日志记录(Loki)

为这些服务配置存储的过程取决于 OpenShift Data Foundation 部署中使用的基础架构。

警告

始终确保您配置的以下 OpenShift 服务有大量存储容量：

- OpenShift 镜像 registry
- OpenShift 监控
- OpenShift 日志记录(Loki)
- OpenShift tracing Platform (Tempo)

如果这些关键服务的存储空间不足，OpenShift 集群将变得不可用，很难恢复。

红帽建议为这些服务配置较短的策展和保留间隔。详情请参阅 *OpenShift* Container Platform 文档中的为 Prometheus 指标数据配置 Curator 调度和修改保留时间。

如果您确实耗尽这些服务的存储空间，请联系红帽客户支持。

4.1. 将镜像 REGISTRY 配置为使用 OPENSHIFT DATA FOUNDATION

OpenShift Container Platform 提供了一个内建的容器镜像 Registry，它作为一个标准的工作负载在集群中运行。registry 通常用作集群中构建的镜像的发布目标，以及在集群中运行的工作负载的镜像源。

按照本节中的说明，将 OpenShift Data Foundation 配置为 Container Image Registry 的存储。在 AWS 中，不需要更改 registry 的存储。但是，建议将存储更改为 vSphere 和裸机平台的 OpenShift Data Foundation 持久卷。

警告

此过程不会将数据从现有镜像 registry 迁移到新镜像 registry。如果您在现有 registry 中已有容器镜像，请在完成此过程前备份 registry，并在这个过程完成后重新注册您的镜像。

先决条件

- 具有 OpenShift Web 控制台的管理访问权限。
- OpenShift Data Foundation Operator 在 **openshift-storage** 命名空间上安装并运行。在 OpenShift Web 控制台中，点 **Operators → Installed Operators** 查看已安装的 Operator。
- Image Registry Operator 在 **openshift-image-registry** 命名空间中安装并运行。在 OpenShift Web 控制台中，点 **Administration → Cluster Settings → Cluster Operators** 查看集群操作器。
- 带有 provisioner **openshift-storage.cephfs.csi.ceph.com** 的存储类可用。在 OpenShift Web 控制台中，点 **Storage → StorageClasses** 查看可用的存储类。

流程

1. 为镜像 Registry 创建一个持久性卷声明。
   a. 在 OpenShift Web 控制台中，点击 **Storage → Persistent Volume Claims**。
   b. 将 **Project** 设置为 **openshift-image-registry**。
   c. 单击 **Create Persistent Volume Claim**。
      i. 在上方检索的可用存储类列表中，使用置备程序 **openshift-storage.cephfs.csi.ceph.com** 指定存储类。
      ii. 指定持久性卷声明 **名称**，如 **ocs4registry**。
      iii. 指定 **Shared Access (RWX)** 访问模式。
      iv. 将 **Size** 指定为最少 100 GB。
      v. 点 **Create**。
      等待新持久卷声明的状态变为 **Bound**。
2. 将集群的 Image Registry 配置为使用新的持久卷声明。
   a. 点 **Administration → Custom Resource Definitions**。
   b. 点与 **imageregistry.operator.openshift.io** 组关联的 **Config** 自定义资源定义。
   c. 点 **实例** 选项卡。
   d. 在集群实例外，点 **Action Menu(⋮) → Edit Config**。
   e. 添加新的持久性卷声明作为镜像 Registry 的持久性存储。
      i. 在 **spec:** 下添加以下内容，并替换现有的 **storage:** 部分（如有必要）。

```
storage:
  pvc:
    claim: <new-pvc-name>
```

例如：

```
storage:
  pvc:
    claim: ocs4registry
```

ii. 点 **Save**。

3. **验证新配置是否正在使用。**

   a. 点击 **Workloads → Pods**。

   b. 将 **Project** 设置为 **openshift-image-registry**。

   c. 验证新的 **image-registry-*** pod 的状态是否为 **Running**，并且以前的 **image-registry-*** pod 已终止。

   d. 点击新 **image-registry-*** Pod 查看 pod 详情。

   e. 向下滚动到 **Volumes**，再验证 **registry-storage** 卷是否具有与您的新持久性卷声明匹配的 **Type**，如 **ocs4registry**。

4.2. 使用 MULTICLOUD 对象网关作为 OPENSHIFT IMAGE REGISTRY 后端存储

您可以在 on-prem OpenShift 部署中使用 Multicloud Object Gateway (MCG)作为 OpenShift Container Platform (OCP) Image Registry 后端存储。

要将 MCG 配置为 OCP 镜像 registry 的后端存储，请按照流程中介绍的步骤操作。

先决条件

- OCP Web 控制台的管理访问权限.
- 使用 MCG 运行 OpenShift Data Foundation 集群。

流程

1. 按照 Creating Object Bucket Claim 中的步骤创建 **ObjectBucketClaim**。
2. 创建 **image-registry-private-configuration-user** secret。

   a. 进入 OpenShift web-console。

   b. 点 **ObjectBucketClaim** -→ **ObjectBucketClaim Data**。

   c. 在 **ObjectBucketClaim 数据** 中，在 **openshift-image-registry namespace** 中查找 **MCG access key** 和 **MCG secret key**。

   d. 使用以下命令创建 secret：

```
$ oc create secret generic image-registry-private-configuration-user --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=<MCG Accesskey> --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=<MCG Secretkey> --namespace openshift-image-registry
```

3. 将 Image Registry Operator 的 **managementState** 改为 **Managed**。

```
$ oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec":
{"managementState": "Managed"}}'
```

4. 编辑 Image Registry Operator 配置文件的 **spec.storage** 部分：

   a. 从 Web 控制台获取 **Object Bucket Claim Data** 部分下的 **unique-bucket-name** 和 **regionEndpoint**，**或者**也可以使用以下命令获取 regionEndpoint 和 unique-bucket-name 的信息：

   ```
   $ oc describe  noobaa
   ```

   b. 将 **regionEndpoint** 添加为 http://<Endpoint-name>:<port> 如果

   - StorageClass 是 **ceph-rgw** 存储类，
   - 端点指向来自 openshift-storage 命名空间中的内部 SVC。

   c. 在对 Operator registry 配置文件进行更改后，**image-registry** pod 会生成。

   ```
   $ oc edit configs.imageregistry.operator.openshift.io -n openshift-image-registry
   apiVersion: imageregistry.operator.openshift.io/v1
   kind: Config
   metadata:
   [..]
   name: cluster
   spec:
   [..]
   storage:
   s3:

     bucket: <Unique-bucket-name>

     region: us-east-1 (Use this region as default)

     regionEndpoint: https://<Endpoint-name>:<port>

     virtualHostedStyle: false
   ```

5. 将镜像 registry 设置重置为默认值。

   ```
   $ oc get pods -n openshift-image-registry
   ```

验证步骤

- 运行以下命令检查是否成功将 MCG 配置为 OpenShift Image Registry 后端存储。

  ```
  $ oc get pods -n openshift-image-registry
  ```

  输出示例

  ```
  $ oc get pods -n openshift-image-registry

  NAME                               READY   STATUS    RESTARTS   AGE
  ```

```
cluster-image-registry-operator-56d78bc5fb-bxcgv   2/2     Running     0        44d
image-pruner-1605830400-29r7k                      0/1     Completed   0        10h
image-registry-b6c8f4596-ln88h                     1/1     Running     0        17d
node-ca-2nxvz                                      1/1     Running     0        44d
node-ca-dtwjd                                      1/1     Running     0        44d
node-ca-h92rj                                      1/1     Running     0        44d
node-ca-k9bkd                                      1/1     Running     0        44d
node-ca-stkzc                                      1/1     Running     0        44d
node-ca-xn8h4                                      1/1     Running     0        44d
```

- （可选）您也可以运行以下命令来验证是否将 MCG 配置为 OpenShift Image Registry 后端存储。

```
$ oc describe pod <image-registry-name>
```

输出示例

```
$ oc describe pod image-registry-b6c8f4596-ln88h

Environment:

    REGISTRY_STORAGE_S3_REGIONENDPOINT:      http://s3.openshift-storage.svc

    REGISTRY_STORAGE:                        s3

    REGISTRY_STORAGE_S3_BUCKET:              bucket-registry-mcg

    REGISTRY_STORAGE_S3_REGION:              us-east-1

    REGISTRY_STORAGE_S3_ENCRYPT:             true

    REGISTRY_STORAGE_S3_VIRTUALHOSTEDSTYLE:  false

    REGISTRY_STORAGE_S3_USEDUALSTACK:        true

    REGISTRY_STORAGE_S3_ACCESSKEY:           <set to the key
'REGISTRY_STORAGE_S3_ACCESSKEY' in secret 'image-registry-private-configuration'>
Optional: false

    REGISTRY_STORAGE_S3_SECRETKEY:           <set to the key
'REGISTRY_STORAGE_S3_SECRETKEY' in secret 'image-registry-private-configuration'>
Optional: false

    REGISTRY_HTTP_ADDR:                      :5000

    REGISTRY_HTTP_NET:                       tcp

    REGISTRY_HTTP_SECRET:
57b943f691c878e342bac34e657b702bd6ca5488d51f839fecafa918a79a5fc6ed70184cab04760
1403c1f383e54d458744062dcaaa483816d82408bb56e686f

    REGISTRY_LOG_LEVEL:                      info

    REGISTRY_OPENSHIFT_QUOTA_ENABLED:        true
```

```
    REGISTRY_STORAGE_CACHE_BLOBDESCRIPTOR:   inmemory

    REGISTRY_STORAGE_DELETE_ENABLED:         true

    REGISTRY_OPENSHIFT_METRICS_ENABLED:      true

    REGISTRY_OPENSHIFT_SERVER_ADDR:          image-registry.openshift-image-
registry.svc:5000

    REGISTRY_HTTP_TLS_CERTIFICATE:           /etc/secrets/tls.crt

    REGISTRY_HTTP_TLS_KEY:                   /etc/secrets/tls.key
```

4.3. 配置监控以使用 OPENSHIFT 数据基础

OpenShift 数据基础提供由 Prometheus 和 Alert Manager 组成的监控堆栈。

按照本节中的说明，将 OpenShift 数据基础配置为监控堆栈的存储。

重要

如果存储空间不足，则监控将无法正常工作。始终确保您拥有大量用于监控的存储容量。

红帽建议为此服务配置简短的保留间隔。详情请参阅 OpenShift Container Platform 文档中的 Prometheus 指标数据的修改保留时间。

先决条件

- 具有 OpenShift Web 控制台的管理访问权限。
- OpenShift Data Foundation Operator 在 **openshift-storage** 命名空间上安装并运行。在 OpenShift Web 控制台中，点 **Operators → Installed Operators** 查看已安装的 Operator。
- 监控 Operator 在 **openshift-monitoring** 命名空间内安装并运行。在 OpenShift Web 控制台中，点 **Administration → Cluster Settings → Cluster Operators** 查看集群操作器。
- 带有 provisioner **openshift-storage.rbd.csi.ceph.com** 的存储类可用。在 OpenShift Web 控制台中，点 **Storage → StorageClasses** 查看可用的存储类。

流程

1. 在 OpenShift Web 控制台中，前往 **Workloads → Config Maps**。
2. 将 **Project** 下拉菜单设置为 **openshift-monitoring**。
3. 单击 **Create Config Map**。
4. 使用以下命令定义一个新的 **cluster-monitoring-config** Config Map。
   将尖括号 (**<**, **>**) 中的内容替换为您自己的值，如 **retention: 24h** 或 **storage: 40Gi**。

   将 **storageClassName** 替换为使用 provisioner **openshift-storage.rbd.csi.ceph.com** 的 **storageclass**。在下例中，storageclass 的名称为 **ocs-storagecluster-ceph-rbd**。

   cluster-monitoring-config Config Map 示例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      retention: <time to retain monitoring files, e.g. 24h>
      volumeClaimTemplate:
        metadata:
          name: ocs-prometheus-claim
        spec:
          storageClassName: ocs-storagecluster-ceph-rbd
          resources:
            requests:
              storage: <size of claim, e.g. 40Gi>
    alertmanagerMain:
      volumeClaimTemplate:
        metadata:
          name: ocs-alertmanager-claim
        spec:
          storageClassName: ocs-storagecluster-ceph-rbd
          resources:
            requests:
              storage: <size of claim, e.g. 40Gi>
```

5. 单击 **Create** 以保存并创建 Config Map。

验证步骤

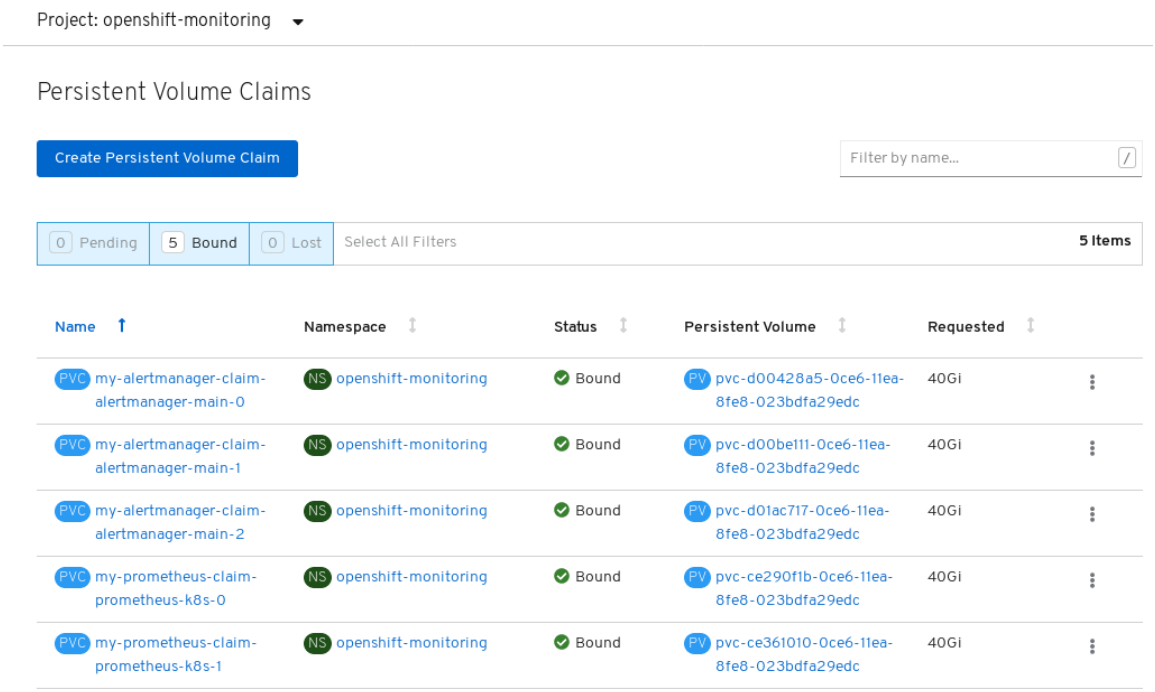
1. 验证持久卷声明是否已绑定到 pod。

   a. 进入 **Storage → Persistent Volume Claims**。

   b. 将 **Project** 下拉菜单设置为 **openshift-monitoring**。

   c. 验证 5 持久性卷声明是否可见，状态为 **Bound**，附加到三个 **alertmanager-main-*** pod，以及两个 **prometheus-k8s-*** pod。

图 4.1. 监控创建和绑定的存储



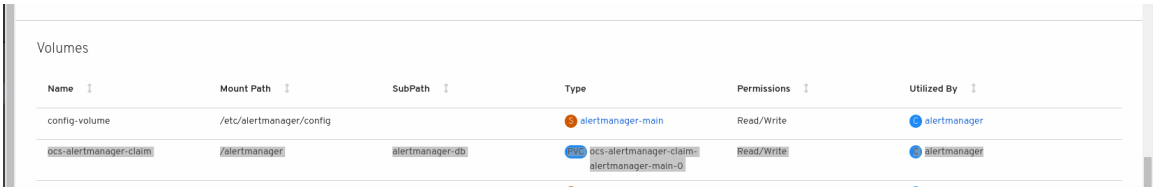
2. 验证新 **alertmanager-main-*** pod 的状态是否显示为 **Running**。

   a. 进入 **Workloads → Pods**。

   b. 点击新 **alertmanager-main-*** pod 查看 pod 详情。

   c. 向下滚动到 **Volumes**，再验证卷是否具有 **Type** (**ocs-alertmanager-claim**)，它与您的新持久性卷声明匹配，如 **ocs-alertmanager-claim-alertmanager-main-0**。

   图 4.2. 附加到 alertmanager-main-* pod 的持久性卷声明

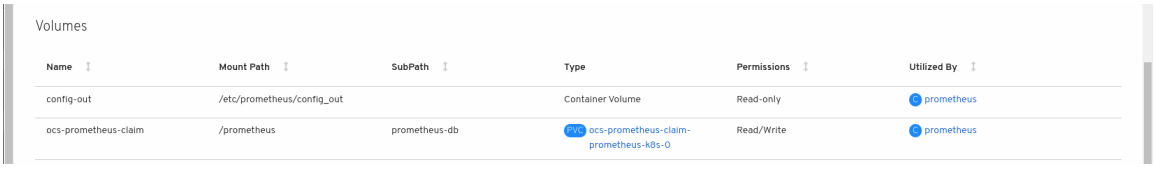


3. 验证新的 **prometheus-k8s-*** pod 的状态是否为 **Running**。

   a. 点新的 **prometheus-k8s-*** Pod 查看 pod 详情。

   b. 向下滚动到 **Volumes**，再验证卷是否具有 **Type** (**ocs-prometheus-claim**)，它与您的新持久性卷声明匹配，如 **ocs-prometheus-claim-prometheus-k8s-0**。

   图 4.3. 附加到 prometheus-k8s-* pod 的持久性卷声明

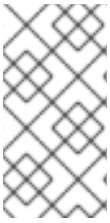


4.4. OVERPROVISION 级别策略控制 [技术预览]

Overprovision (超额置备) 控制是一种机制，它可让您根据特定的应用程序命名空间，定义从存储集群中使用的持久性卷声明(PVC)的配额。

当您启用 overprovision 控制机制时，它会阻止置备存储集群消耗的 PVC。OpenShift 提供了定义约束的灵活性，可以利用 **ClusterResourceQuota** 来限制集群范围内聚合的资源消耗。如需更多信息，请参阅 OpenShift ClusterResourceQuota。

通过超额置备控制，会启动 **ClusteResourceQuota**，您可以为每个存储类设置存储容量限制。当消耗了 80% 的容量限制时，警报将触发。



注意

Overprovision 级别策略控制是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。如需更多信息，请参阅技术预览功能支持范围。

如需有关 OpenShift Data Foundation 部署的更多信息，请参阅 产品文档 并根据平台选择部署过程。

先决条件

- 确保创建了 OpenShift Data Foundation 集群。

流程

1. 通过命令行界面或用户界面部署 **storagecluster**。
2. 标记应用程序命名空间。

```
apiVersion: v1
kind: Namespace
metadata:
    name: <desired_name>
    labels:
        storagequota: <desired_label>
```

<desired_name>

为 application 命名空间指定一个名称，如 **quota-rbd**。

<desired_label>

为存储配额指定一个标签，如 **storagequota1**。

3. 编辑 **storagecluster**，以在存储类上设置配额限制。

```
$ oc edit storagecluster -n openshift-storage <ocs_storagecluster_name>
```

<ocs_storagecluster_name>

指定存储集群的名称。

4. 在 **StorageCluster.Spec** 中为 Overprovision Control 添加一个条目：

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
spec:
```

```
[...]
   overprovisionControl:
   - capacity: <desired_quota_limit>
       storageClassName: <storage_class_name>
       quotaName: <desired_quota_name>
       selector:
         labels:
               matchLabels:
                storagequota: <desired_label>
[...]
```

<desired_quota_limit>

为存储类指定所需的配额限制，例如 **27Ti**。

<storage_class_name>

指定要设置配额限制的存储类的名称，如 **ocs-storagecluster-ceph-rbd**。

<desired_quota_name>

为存储配额指定一个名称，如 **quota1**。

<desired_label>

为存储配额指定一个标签，如 **storagequota1**。

5. 保存修改后的 **storagecluster**。

6. 验证是否定义了 **clusterresourcequota**。

注意

期望 **clusterresourcequota** 带有您在上一步中定义的 **quotaName**，例如 **quota1**。

```
$ oc get clusterresourcequota -A

$ oc describe clusterresourcequota -A
```

4.5. OPENSHIFT 数据基础的集群日志记录

您可以部署集群日志记录来聚合一系列 OpenShift Container Platform 服务的日志。有关如何部署集群日志记录的详情，请参考部署集群日志记录。

在初始 OpenShift Container Platform 部署时，默认情况下不配置 OpenShift Data Foundation，OpenShift Container Platform 集群将依赖于节点提供的默认存储。您可以编辑 OpenShift 日志记录(ElasticSearch)的默认配置，使其由 OpenShift Data Foundation 支持，使 OpenShift Data Foundation 支持日志(Elasticsearch)。

重要

始终确保您具有适用于这些服务的大量存储容量。如果您对这些关键服务的存储空间不足，日志记录应用将变得不可用，很难恢复。

红帽建议为这些服务配置较短的策展和保留间隔。详情请参阅 OpenShift Container Platform 文档中的 集群日志记录 Curator。

如果您缺少这些服务的存储空间，请联系红帽客户支持。

4.5.1. 配置持久性存储

您可以使用存储类名称和大小参数为 Elasticsearch 集群配置持久性存储类和大小。Cluster Logging Operator 根据这些参数为 Elasticsearch 集群中的每个数据节点创建一个持久性卷声明。例如：

```
spec:
   logStore:
     type: "elasticsearch"
     elasticsearch:
       nodeCount: 3
       storage:
         storageClassName: "ocs-storagecluster-ceph-rbd”
         size: "200G"
```

本例指定，集群中的每个数据节点将绑定到请求 **200GiB** 的 **ocs-storagecluster-ceph-rbd** 存储的持久性卷声明。每个主分片将由单个副本支持。分片的副本会在所有节点之间复制，并且始终可用；如果因为单一冗余策略至少存在两个节点，则可以恢复副本。如需有关 Elasticsearch 复制策略的信息，请参阅 关于部署和配置集群日志记录 中的 *Elasticsearch 复制策略*。

注意

缺少存储块将导致默认存储支持部署。例如：

```
spec:
   logStore:
     type: "elasticsearch"
     elasticsearch:
       nodeCount: 3
       storage: {}
```

如需更多信息，请参阅 配置集群日志记录。

4.5.2. 配置集群日志记录以使用 OpenShift Data Foundation

按照本节中的说明，将 OpenShift Data Foundation 配置为 OpenShift 集群日志记录的存储。

注意

当您首次在 OpenShift 数据基础中配置日志记录时，您可以获取所有日志。但是，在卸载和重新安装日志记录后，会删除旧日志并只处理新日志。

先决条件

- 具有 OpenShift Web 控制台的管理访问权限。
- OpenShift Data Foundation Operator 在 **openshift-storage** 命名空间上安装并运行。
- Cluster logging Operator 已安装并在 **openshift-logging** 命名空间中运行。

流程

1. 从 OpenShift Web 控制台左侧窗格中，点击 **Administration → Custom Resource Definitions**。
2. 在 Custom Resource Definitions 页面中点 **ClusterLogging**。
3. 在 Custom Resource Definition Overview 页面上，从 Actions 菜单中选择 **View Instances**，或者点击 **Instances** 选项卡。
4. 在 Cluster Logging 页面上，点击 **Create Cluster Logging**。
   您可能需要刷新页面来加载数据。
5. 在 YAML 中，将 **storageClassName** 替换为使用 provisioner **openshift-storage.rbd.csi.ceph.com** 的 **storageclass**。在下例中，storageclass 的名称为 **ocs-storagecluster-ceph-rbd** ：

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: ocs-storagecluster-ceph-rbd
        size: 200G # Change as per your requirement
      redundancyPolicy: "SingleRedundancy"
  visualization:
    type: "kibana"
    kibana:
      replicas: 1
  curation:
    type: "curator"
    curator:
      schedule: "30 3 * * *"
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
```

如果 OpenShift Data Foundation 节点带有污点，您必须添加容限，以启用为日志调度 daemonset pod。

```
spec:
[...]
  collection:
```

```
    logs:
      fluentd:
        tolerations:
        - effect: NoSchedule
          key: node.ocs.openshift.io/storage
          value: 'true'
      type: fluentd
```

6. 点 **Save**。

验证步骤

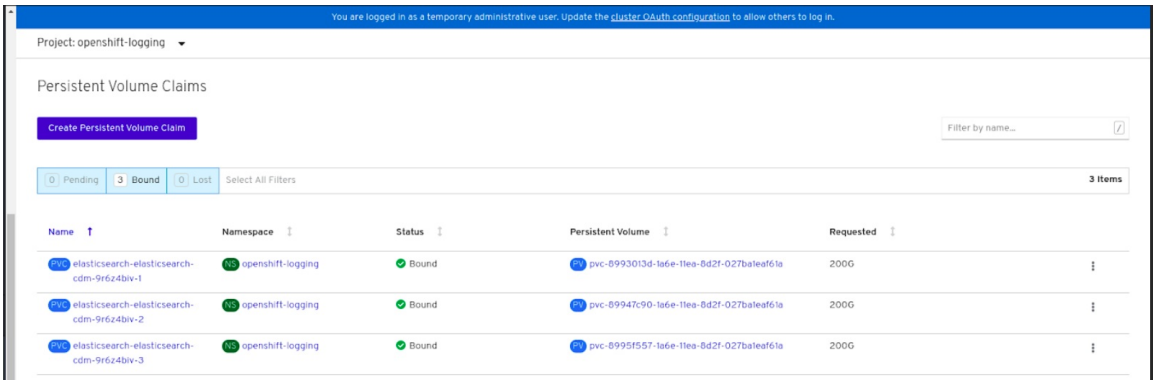
1. 验证持久卷声明是否已绑定到 **elasticsearch** Pod。

   a. 进入 **Storage → Persistent Volume Claims**。

   b. 将 **Project** 下拉菜单设置为 **openshift-logging**。

   c. 验证持久卷声明是否可见，状态为 **Bound**，附加到 **elasticsearch-*** pod。

   图 4.4. 创建并绑定集群日志记录



2. 验证是否在使用新集群日志记录。

   a. 点 **Workload → Pods**。

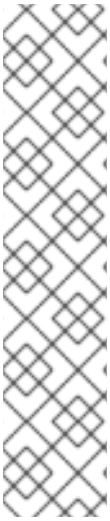
   b. 将项目设置为 **openshift-logging**。

   c. 验证新的 **elasticsearch-*** Pod 的状态是否为 **Running**。

   d. 点新的 **elasticsearch-*** Pod 查看 pod 详情。

   e. 向下滚动到 **Volumes**，再验证 elasticsearch 卷是否具有与新持久性卷声明匹配的 **Type**，如 **elasticsearch-elasticsearch-cdm-9r624biv-3**。

   f. 点 Persistent Volume Claim 名称，然后在 PersistentVolumeClaim Overview 页面中验证存储类名称。



注意

确保使用较短的 Curator 时间，以避免在附加到 Elasticsearch Pod 的 PV 上 PV 完整场景。

您可以配置 Curator，以根据保留设置删除 Elasticsearch 数据。建议您将以下默认索引数据保留 5 天设为默认值。

```
config.yaml: |
    openshift-storage:
      delete:
        days: 5
```

如需了解更多详细信息，请参阅 Elasticsearch 数据。

注意

要卸载由持久性卷声明支持的集群日志记录，请使用相应部署指南的卸载章节中从 OpenShift Data Foundation 中删除集群日志记录 Operator 的步骤。

第 5 章 创建 MULTUS 网络

OpenShift Container Platform 使用 Multus CNI 插件来实现对 CNI 插件的链接。您可以在集群安装过程中配置默认 pod 网络。默认网络处理集群中的所有一般网络流量。

您可以基于可用的 CNI 插件定义额外网络，并将一个或多个此类网络附加到 pod。要将额外网络接口附加到 pod，您必须创建配置来定义接口的附加方式。

您可以使用 NetworkAttachmentDefinition（NAD）自定义资源 (CR) 来指定每个接口。每个 NetworkAttachmentDefinition 中的 CNI 配置定义如何创建该接口。

OpenShift Data Foundation 使用名为 macvlan 的 CNI 插件。创建基于 macvlan 的额外网络可让主机上的 pod 通过使用物理网络接口与其他主机和那些主机上的 pod 通信。附加到基于 macvlan 的额外网络的每个 pod 都会获得一个唯一的 MAC 地址。

5.1. 创建网络附加定义

要使用 Multus，需要一个已具有正确网络配置的集群，请参阅 Multus 配置的要求。新创建的 NetworkAttachmentDefinition(NAD)可以在 Storage Cluster 安装过程中选择。这就是必须在存储集群之前创建它们的原因。

注意

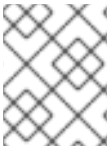
网络附加定义只能使用 **whereabouts** IP 地址管理 (IPAM) 的位置，它必须指定 **range** 字段。不支持 **ipRanges** 和插件链。

您可以在存储集群安装过程中选择新创建的 **NetworkAttachmentDefinition** (NAD)。这是您在创建存储集群前必须创建 NAD 的原因。

如规划指南中所述，您创建的 Multus 网络取决于您用于 OpenShift Data Foundation 流量的可用网络接口数量。可以将所有存储流量分隔到两个接口中的一个接口（一个用于默认 OpenShift SDN），或者将存储流量进一步分隔到客户端存储流量（公共）和存储复制流量（私有或集群）。

以下是同一接口上所有存储流量（公共和集群）的 **NetworkAttachmentDefinition** 示例。它要求所有可调度节点上有一个额外的接口（OpenShift 默认 SDN 在单独的网络接口上）：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ocs-public-cluster
  namespace: openshift-storage
spec:
  config: '{
   "cniVersion": "0.3.1",
   "type": "macvlan",
   "master": "ens2",
   "mode": "bridge",
   "ipam": {
        "type": "whereabouts",
        "range": "192.168.1.0/24"
   }
  }'
```



注意

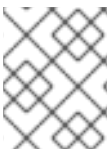
所有网络接口名称必须在附加至 Multus 网络的所有节点上相同（即 **ocs-public-cluster**的 **ens2** ）。

以下是用于单独 Multus 网络上存储流量的 **NetworkAttachmentDefinition** 示例，适用于客户端存储流量，以及用于复制流量的集群。它需要在 OpenShift 节点上有两个额外的接口，托管对象存储设备(OSD) pod，在所有其他可调度节点上有一个额外的接口(OpenShift 默认 SDN 在单独的网络接口上)：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ocs-public
  namespace: openshift-storage
spec:
  config: '{
   "cniVersion": "0.3.1",
   "type": "macvlan",
   "master": "ens2",
   "mode": "bridge",
   "ipam": {
        "type": "whereabouts",
        "range": "192.168.1.0/24"
   }
  }'
```

NetworkAttachmentDefinition 示例：

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: ocs-cluster
  namespace: openshift-storage
spec:
  config: '{
   "cniVersion": "0.3.1",
   "type": "macvlan",
   "master": "ens3",
   "mode": "bridge",
   "ipam": {
        "type": "whereabouts",
        "range": "192.168.2.0/24"
   }
  }'
```



注意

所有网络接口名称必须在附加至 Multus 网络的所有节点上相同（即，**ens2** 用于 **ocs-public**，**ens3** 用于 **ocs-cluster**）。

第 6 章 使用 OPENSHIFT DATA FOUNDATION 支持 OPENSHIFT CONTAINER PLATFORM 应用程序

您无法在 OpenShift Container Platform 安装过程中直接安装 OpenShift Data Foundation。但是，您可以使用 Operator Hub 在现有 OpenShift Container Platform 上安装 OpenShift Data Foundation，然后将 OpenShift Container Platform 应用程序配置为由 OpenShift Data Foundation 支持。

先决条件

- 已安装 OpenShift Container Platform，您还可管理 OpenShift Web 控制台。
- OpenShift Data Foundation 在 **openshift-storage** 命名空间上安装并运行。

流程

1. 在 OpenShift Web 控制台中执行以下任一操作：
   - 点 **Workloads → Deployments**。
     在 Deployments 页面中，您可以执行以下操作之一：
     - 从 **Action** 菜单中选择任何现有部署并点击 **Add Storage** 选项。
     - 创建新部署，然后添加存储。
       i. 单击 **Create Deployment** 以创建新部署。
       ii. 根据您的要求编辑 **YAML** 以创建部署。
       iii. 点 **Create**。
       iv. 从页面右上角的 **Actions** 下拉菜单中选择 **Add Storage**。
   - 点 **Workloads → Deployment Configs**。
     在 Deployment Configs 页面中，您可以执行以下操作之一：
     - 从 **Action** 菜单中选择任何现有部署并点击 **Add Storage** 选项。
     - 创建新部署，然后添加存储。
       i. 单击 **Create Deployment Config** 以创建新部署。
       ii. 根据您的要求编辑 **YAML** 以创建部署。
       iii. 点 **Create**。
       iv. 从页面右上角的 **Actions** 下拉菜单中选择 **Add Storage**。
2. 在 Add Storage 页面中，您可以选择以下选项之一：
   - 点击 **Use existing claim** 选项，然后从下拉菜单中选择适合的 PVC。
   - 单击 **Create new claim** 选项。
     a. 从 **Storage Class** 下拉列表中，选择适当的 **CephFS** 或 **RBD** 存储类。
     b. 为持久性卷声明提供名称。

c. 选择 ReadWriteOnce (RWO) 或 ReadWriteMany (RWX) 访问模式。

注意

ReadOnlyMany (ROX) 已被取消激活，因为它不受支持。

d. 选择所需存储容量的大小。

注意

您可以扩展块 PV，但无法在创建持久性卷声明后减少存储容量。

3. 指定容器内挂载路径卷的挂载路径和子路径（如果需要）。
4. 点 **Save**。

验证步骤

1. 根据您的配置，执行以下任一操作：
   - 点 **Workloads → Deployments**。
   - 点 **Workloads → Deployment Configs**。
2. 根据需要设置项目。
3. 单击您添加存储的部署，以显示部署详情。
4. 向下滚动到 **Volumes**，再验证您的部署带有一个与您分配的持久性卷声明相匹配的类型。
5. 点 Persistent Volume Claim 名称，然后在 Persistent Volume Claim Overview 页面中验证存储类名称。

第 7 章 将文件和对象存储添加到现有外部 OPENSHIFT DATA FOUNDATION 集群

当 OpenShift Data Foundation 配置为外部模式时，可以通过多种方式为持久性卷声明和对象存储桶声明提供存储。

- 块存储的持久卷声明直接从外部 Red Hat Ceph Storage 集群提供。
- 文件存储的持久卷声明可以通过向外部 Red Hat Ceph Storage 添加元数据服务器(MDS)提供。
- 对象存储的对象 bucket 声明可以通过使用 Multicloud 对象网关或将 Ceph 对象网关添加到外部 Red Hat Ceph Storage 集群来提供。

使用以下流程，将文件存储（使用元数据服务器）或对象存储（使用 Ceph 对象网关）添加到最初部署为仅提供块存储的外部 OpenShift Data Foundation 集群。

先决条件

- OpenShift Data Foundation 4.14 已安装并在 OpenShift Container Platform 版本 4.14 或更高版本中运行。另外，处于外部模式的 OpenShift Data Foundation 集群处于 **Ready** 状态。
- 您的外部 Red Hat Ceph Storage 集群被配置为以下一个或多个集群：
  - 一个 Ceph 对象网关(RGW)端点，可由 OpenShift Container Platform 集群访问以用于对象存储
  - 用于文件存储的元数据服务器(MDS)池
- 确保您知道在外部 OpenShift Data Foundation 集群部署期间用于 **ceph-external-cluster-details-exporter.py** 脚本的参数。

流程

1. 使用以下命令下载 OpenShift Data Foundation 的 **ceph-external-cluster-details-exporter.py** python 脚本：

```
oc get csv $(oc get csv -n openshift-storage | grep ocs-operator | awk '{print $1}') -n openshift-storage -o jsonpath='{.metadata.annotations.external\.features\.ocs\.openshift\.io/export-script}' | base64 --decode > ceph-external-cluster-details-exporter.py
```

2. 通过在外部 Red Hat Ceph Storage 集群中的任何客户端节点上运行 **ceph-external-cluster-details-exporter.py**，更新外部 Red Hat Ceph Storage 存储集群的权限上限。您可能需要要求您的 Red Hat Ceph Storage 管理员来执行此操作。

```
# python3 ceph-external-cluster-details-exporter.py --upgrade \
--run-as-user=ocs-client-name \
--rgw-pool-prefix rgw-pool-prefix
```

--run-as-user

在 OpenShift Data Foundation 集群部署期间使用的客户端名称。如果未设置其他客户端名称，请使用默认的客户端名称 **client.healthchecker**。

--rgw-pool-prefix

用于 Ceph 对象网关池的前缀。如果使用默认前缀，可以省略它。

3. 从外部 Red Hat Ceph Storage 生成并保存配置详情。

   a. 通过在外部 Red Hat Ceph Storage 集群中的任何客户端节点上运行 **ceph-external-cluster-details-exporter.py** 生成配置详情。

   ```
   # python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name rbd-block-pool-name --monitoring-endpoint ceph-mgr-prometheus-exporter-endpoint --monitoring-endpoint-port ceph-mgr-prometheus-exporter-port --run-as-user ocs-client-name --rgw-endpoint rgw-endpoint --rgw-pool-prefix rgw-pool-prefix
   ```

   --monitoring-endpoint

   是可选的。它接受可从 OpenShift Container Platform 集群访问的活跃和待机 mgrs 的以逗号分隔的 IP 地址列表。如果没有提供，则会自动填充该值。

   --monitoring-endpoint-port

   是可选的。它是与 **--monitoring-endpoint** 指定的 ceph-mgr Prometheus exporter 关联的端口。如果没有提供，则会自动填充该值。

   --run-as-user

   在 OpenShift Data Foundation 集群部署期间使用的客户端名称。如果未设置其他客户端名称，请使用默认的客户端名称 client.healthchecker。

   --rgw-endpoint

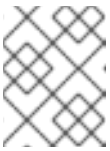
   提供此参数，以通过 Ceph 对象网关为 OpenShift Data Foundation 置备对象存储（可选参数）

   --rgw-pool-prefix

   用于 Ceph 对象网关池的前缀。如果使用默认前缀，可以省略它。

   用户权限已更新，如下所示：

   ```
   caps: [mgr] allow command config
   caps: [mon] allow r, allow command quorum_status, allow command version
   caps: [osd] allow rwx pool=default.rgw.meta, allow r pool=.rgw.root, allow rw pool=default.rgw.control, allow rx pool=default.rgw.log, allow x pool=default.rgw.buckets.index
   ```

   注意

   确保除 Ceph 对象网关详细信息（如果提供）之外的所有参数（包括可选参数）与在外部模式中部署 OpenShift Data Foundation 期间所用的相同。

   b. 将脚本的输出保存到 **external-cluster-config.json** 文件中。
   以下示例输出以粗体文本显示生成的配置更改。

   ```
   [{"name": "rook-ceph-mon-endpoints", "kind": "ConfigMap", "data": {"data": "xxx.xxx.xxx.xxx:xxxx", "maxMonId": "0", "mapping": "{}"}}, {"name": "rook-ceph-mon", "kind": "Secret", "data": {"admin-secret": "admin-secret", "fsid": "<fs-id>", "mon-secret": "mon-secret"}}, {"name": "rook-ceph-operator-creds", "kind": "Secret", "data": {"userID": "<user-id>", "userKey": "<user-key>"}}, {"name": "rook-csi-rbd-node", "kind": "Secret", "data": {"userID": "csi-rbd-node", "userKey": "<user-key>"}}, {"name": "ceph-rbd", "kind": "StorageClass", "data": {"pool": "<pool>"}}, {"name": "monitoring-endpoint", "kind": "CephCluster", "data": {"MonitoringEndpoint": "xxx.xxx.xxx.xxx", "MonitoringPort": "xxxx"}}, {"name": "rook-ceph-dashboard-link", "kind": "Secret", "data": {"userID": "ceph-dashboard-link", "userKey": "<user-key>"}}, {"name": "rook-csi-rbd-provisioner", "kind":
   ```

"Secret", "data": {"userID": "csi-rbd-provisioner", "userKey": "<user-key>"}}, {"name": "rook-csi-cephfs-provisioner", "kind": "Secret", "data": {"adminID": "csi-cephfs-provisioner", "adminKey": "<admin-key>"}}, {"name": "rook-csi-cephfs-node", "kind": "Secret", "data": {"adminID": "csi-cephfs-node", "adminKey": "<admin-key>"}}, {"name": "cephfs", "kind": "StorageClass", "data": {"fsName": "cephfs", "pool": "cephfs_data"}}, {"name": "ceph-rgw", "kind": "StorageClass", "data": {"endpoint": "xxx.xxx.xxx.xxx:xxxx", "poolPrefix": "default"}}, {"name": "rgw-admin-ops-user", "kind": "Secret", "data": {"accessKey": "<access-key>", "secretKey": "<secret-key>"}}]

4. 上传生成的 JSON 文件。

   a. 登录 OpenShift Web 控制台。

   b. 点 **Workloads → Secrets**。

   c. 将 **project** 设置为 **openshift-storage**。

   d. 点 **rook-ceph-external-cluster-details**。

   e. 点 **Actions** (⋮) → **Edit Secret**

   f. 点 **Browse** 并上传 **external-cluster-config.json** 文件。

   g. 点 **Save**。

验证步骤

- 要验证 OpenShift Data Foundation 集群是否健康，且数据具有弹性，请导航到 **Storage → Data foundation → Storage Systems** 选项卡，然后点击存储系统名称。
  - 在 **Overview → Block and File** 选项卡中，检查 Status 卡以确认 *存储集群* 有一个绿色勾号表示其健康。
- 如果您为文件存储添加了元数据服务器：
  a. 点 **Workloads → Pods**，验证 **csi-cephfsplugin-*** pod 是否已创建新并处于 **Running** 状态。
  b. 点 **Storage → Storage Classes** 并验证是否已创建 **ocs-external-storagecluster-cephfs** 存储类。
- 如果您为对象存储添加了 Ceph 对象网关：
  a. 点 **Storage → Storage Classes** 并验证是否已创建 **ocs-external-storagecluster-ceph-rgw** 存储类。
  b. 要验证 OpenShift Data Foundation 集群是否健康，且数据具有弹性，请导航到 **Storage → Data foundation → Storage Systems** 选项卡，然后点击存储系统名称。
  c. 单击 **Object** 选项卡，并确认 *Object Service* 和 *数据弹性* 具有绿色勾号，指示其运行正常。

第 8 章 如何在 RED HAT OPENSHIFT DATA FOUNDATION 中使用专用 WORKER 节点

任何 Red Hat OpenShift Container Platform 订阅都需要一个 OpenShift Data Foundation 订阅。但是，如果您使用基础架构节点调度 OpenShift 数据基础资源，您可以在 OpenShift Container Platform 订阅上保存。

务必要在不同环境中维持 Machine API 支持的一致性。因此，强烈建议在所有情形中都有特殊类别的节点标记为 worker 或 infra，或者同时具有这两个角色。如需更多信息，请参阅 第 8.3 节 “手动创建基础架构节点” 部分。

8.1. 基础架构节点分析

用于 OpenShift Data Foundation 的基础架构节点有几个属性。需要 **infra** node-role 标签，以确保节点不使用 RHOCP 权利。**infra** node-role 标签负责确保运行 OpenShift Data Foundation 的节点仅需要 OpenShift Data Foundation 权利。

- 标记了 **node-role.kubernetes.io/infra**

还需要添加具有 **NoSchedule effect** 的 OpenShift Data Foundation 污点，以便 **infra** 节点只调度 OpenShift Data Foundation 资源。

- 使用 **node.ocs.openshift.io/storage="true"** 污点

该标签将 RHOCP 节点识别为 **infra** 节点，以便不应用 RHOCP 订阅成本。该污点可防止将非 OpenShift Data Foundation 资源调度到污点节点上。

注意

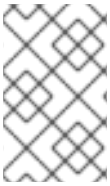
在节点上添加存储污点可能需要对其他 **daemonset** pod（如 **openshift-dns daemonset**）进行容限处理。有关如何管理容限的详情，请参考知识库文章 https://access.redhat.com/solutions/6592171。

用于运行 OpenShift Data Foundation 服务的基础架构节点上的污点和标签示例：

```
spec:
  taints:
  - effect: NoSchedule
    key: node.ocs.openshift.io/storage
    value: "true"
  metadata:
    creationTimestamp: null
    labels:
      node-role.kubernetes.io/worker: ""
      node-role.kubernetes.io/infra: ""
      cluster.ocs.openshift.io/openshift-storage: ""
```

8.2. 用于创建基础架构节点的机器集

如果环境中支持 Machine API，则应将标签添加到要调配基础架构节点的 Machine Sets 的模板中。避免将标签手动添加到机器 API 创建的节点的反模式。这样做类似于向部署创建的 pod 添加标签。在这两种情况下，pod/节点失败时，替代的 pod/节点都将没有适当的标签。

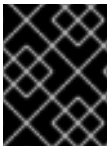


注意

在 EC2 环境中，您将需要三个计算机集，各自配置为在不同的可用区（如 us-east-2a, us-east-2b, us-east-2c）中调配基础架构节点。目前，OpenShift Data Foundation 不支持在超过三个可用区部署。

以下 Machine Set 模板示例创建具有基础架构节点所需的适当污点和标签的节点。这将用于运行 OpenShift Data Foundation 服务。

```
template:
  metadata:
    creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: kb-s25vf
      machine.openshift.io/cluster-api-machine-role: worker
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: kb-s25vf-infra-us-west-2a
  spec:
    taints:
    - effect: NoSchedule
      key: node.ocs.openshift.io/storage
      value: "true"
    metadata:
      creationTimestamp: null
      labels:
        node-role.kubernetes.io/infra: ""
        cluster.ocs.openshift.io/openshift-storage: ""
```



重要

如果向基础架构节点添加污点，您还需要为其他工作负载的污点添加容限，如 fluentd pod。如需更多信息，请参阅 OpenShift 4 中的基础架构节点红帽知识库解决方案。

8.3. 手动创建基础架构节点

只有环境中不支持 Machine API 时，标签才应直接应用到节点。手动创建要求至少可使用 3 个 RHOCP worker 节点来调度 OpenShift Data Foundation 服务，并且这些节点有足够的 CPU 和内存资源。要避免 RHOCP 订阅成本，需要以下内容：

```
oc label node <node> node-role.kubernetes.io/infra=""
oc label node <node> cluster.ocs.openshift.io/openshift-storage=""
```

还需要添加一个 **NoSchedule** OpenShift Data Foundation 污点，以便 **infra** 节点只调度 OpenShift Data Foundation 资源并代表任何其他非 OpenShift Data Foundation 工作负载。

```
oc adm taint node <node> node.ocs.openshift.io/storage="true":NoSchedule
```

警告

不要删除 node-role node-role.kubernetes.io/worker=""

除非对 OpenShift 调度程序和 MachineConfig 资源进行了更改，否则删除 **node-role.kubernetes.io/worker=""** 可能会导致问题。

如果已删除，则应将其重新添加到每个 **infra** 节点。添加 node-role **node-role.kubernetes.io/infra=""** 和 OpenShift Data Foundation 污点足以满足权利的要求。

8.4. 从用户界面污点一个节点

本节解释了在 OpenShift Data Foundation 部署后污点节点的步骤。

流程

1. 在 OpenShift Web 控制台中，点击 **Compute → Nodes**，然后选择必须污点的节点。
2. 在 **Details** 页面中，点 **Edit taint**。
3. 在 **Key** <nodes.openshift.ocs.io/storage>, **Value** <true> and in the **Effect**<Noschedule> 项中输入值。
4. 点 Save。

验证步骤

- 按照以下步骤验证节点是否已成功污点：
  - 进入 **Compute → Nodes**。
  - 选择节点以验证其状态，然后单击 **YAML** 选项卡。
  - 在 **specs** 部分中检查以下参数值：

```
Taints:
  Key: node.openshift.ocs.io/storage
  Value: true
  Effect: Noschedule
```

其他资源

如需更多信息，请参阅在 VMware vSphere 上创建 OpenShift Data Foundation 集群 。

第 9 章 管理持久性卷声明

9.1. 配置应用程序 POD 以使用 OPENSHIFT DATA FOUNDATION

按照本节中的说明，将 OpenShift Data Foundation 配置为应用 pod 的存储。

先决条件

- 具有 OpenShift Web 控制台的管理访问权限。
- OpenShift Data Foundation Operator 在 **openshift-storage** 命名空间上安装并运行。在 OpenShift Web 控制台中，点 **Operators → Installed Operators** 查看已安装的 Operator。
- OpenShift Data Foundation 提供的默认存储类可用。在 OpenShift Web 控制台中，点 **Storage → Storage Classes** 查看默认存储类。

流程

1. **为要使用的应用创建持久性卷声明 (PVC)。**
   a. 在 OpenShift Web 控制台中，点击 **Storage → Persistent Volume Claims**。
   b. 为应用程序 pod 设置 **Project**。
   c. 单击 **Create Persistent Volume Claim**。
      i. 指定由 OpenShift Data Foundation 提供的**存储类**。
      ii. 指定 PVC **Name**，如 **myclaim**。
      iii. 选择所需的 **Access Mode**。

      注意

      IBM FlashSystem 不支持 **Access Mode**, **Shared access (RWX)**。

      iv. 对于 Rados 块设备(RBD)，如果 **Access 模式** 为 ReadWriteOnce(**RWO**)，请选择所需的**卷模式**。默认卷模式是 **Filesystem**。
      v. 根据应用程序要求指定一个 **大小**。
      vi. 点 **Create** 并等待 PVC 处于 **Bound** 状态。
2. **配置新的或现有应用容器集以使用新 PVC。**
   - 对于新应用程序 pod，执行以下步骤：
      i. 点 **Workloads →Pods**。
      ii. 创建新的应用 pod。
      iii. 在 **spec:** 部分下，添加 **volume:** 部分，将新 PVC 添加为应用 Pod 的卷。

      ```
      volumes:
        - name: <volume_name>
      ```

```
    persistentVolumeClaim:
      claimName: <pvc_name>
```

例如：

```
volumes:
  - name: mypd
    persistentVolumeClaim:
      claimName: myclaim
```

- 对于现有应用程序 pod，执行以下步骤：

  i. 点 **Workloads →Deployment Configs**。

  ii. 搜索与应用程序 pod 关联的所需部署配置。

  iii. 点击其 **Action 菜单(⋮) → Edit Deployment Config**。

  iv. 在 **spec:** 部分下，添加 **volume:** 部分，将新 PVC 添加为应用程序 Pod 的卷，然后点 **Save**。

```
volumes:
  - name: <volume_name>
    persistentVolumeClaim:
      claimName: <pvc_name>
```

例如：

```
volumes:
  - name: mypd
    persistentVolumeClaim:
      claimName: myclaim
```

3. **验证新配置是否正在使用。**

   a. 点击 **Workloads → Pods**。

   b. 为应用程序 pod 设置 **Project**。

   c. 验证应用容器集的状态是否为 **Running**。

   d. 单击应用容器集名称，以查看容器集详细信息。

   e. 向下滚动到 **Volumes** 部分，再验证卷的 **Type** 与您的新持久卷声明匹配，如 **myclaim**。

9.2. 查看持久性卷声明请求状态

使用这个流程查看 PVC 请求的状态。

先决条件

- OpenShift Data Foundation 的管理员访问权限。

流程

1. 登录 OpenShift Web 控制台。
2. 点 **Storage → Persistent Volume Claims**
3. 使用 **Filter** 文本框搜索所需的 PVC 名称。您还可以按 Name 或 Label 过滤 PVC 列表来缩小列表范围
4. 检查与所需 PVC 对应的 **Status** 列。
5. 点所需的 **Name** 查看 PVC 详情。

9.3. 查看持久性卷声明请求事件

使用这个流程来查看和解决持久性卷声明 (PVC) 请求事件。

先决条件

- 管理员对 OpenShift Web 控制台的访问权限。

流程

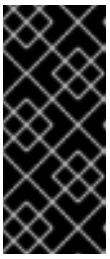
1. 在 OpenShift Web 控制台中，点 **Storage → Data Foundation**。
2. 在 **Storage systems** 选项卡中，选择存储系统，然后点 **Overview → Block and File**。
3. 找到 **Inventory** 卡，查看 PVC 数量并显示错误。
4. 点 **Storage → Persistent Volume Claims**
5. 使用 **Filter** 文本框搜索所需的 PVC。
6. 点 PVC 名称并导航到 **Events**
7. 根据需要或按指示处理事件。

9.4. 扩展持久性卷声明

自 OpenShift Data Foundation 4.6 开始，能够扩展持久性卷声明，在管理持久性存储资源方面提供更多灵活性。

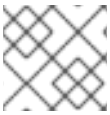
以下持久性卷支持扩展：

- 具有 ReadWriteOnce (RWO) 和 ReadWriteMany (RWX) 访问权限的 PVC，这些访问基于 Ceph 文件系统 (CephFS)，用于卷模式 **Filesystem**。
- 具有 ReadWriteOnce (RWO) 访问的 PVC，它基于卷模式 **Filesystem** 的 Ceph RADOS 块设备 (RBD)。
- 具有 ReadWriteOnce (RWO) 访问的 PVC，它基于卷模式 **Block** 的 Ceph RADOS 块设备 (RBD)。
- 具有 ReadWriteOncePod (RWOP)的 PVC，它基于 Ceph 文件系统(CephFS)或网络文件系统 (NFS)用于卷模式 **Filesystem**。
- 具有 ReadWriteOncePod (RWOP)访问的 PVC，它基于卷模式 **Filesystem** 的 Ceph RADOS 块设备(RBD)。使用 RWOP 访问模式，您可以将单个 pod 挂载到单个节点上以读写模式挂载。



重要

ReadWriteOncePod (RWOP)访问模式是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。



注意

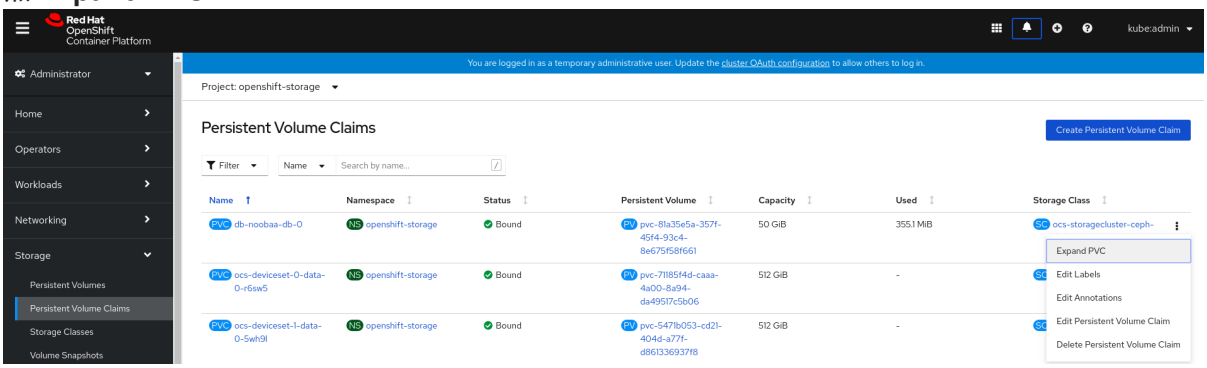
OSD、MON 和加密 PVC 不支持 PVC 扩展。

先决条件

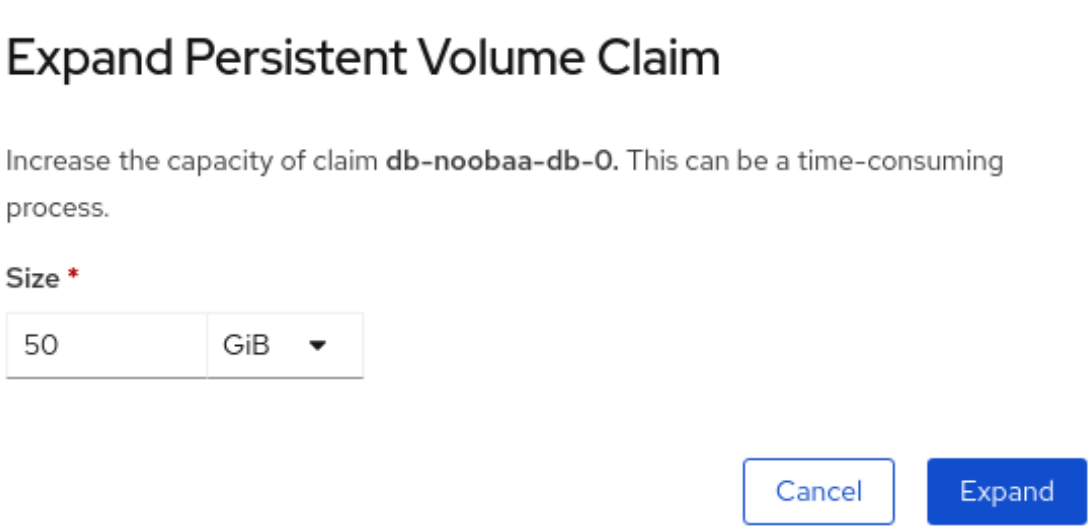
- 管理员对 OpenShift Web 控制台的访问权限。

流程

1. 在 OpenShift Web 控制台中，导航到 **Storage → Persistent Volume Claims**。
2. 点击您要扩展的持久性卷声明旁边的 Action Menu(⋮)。
3. 点 **Expand PVC** ：



4. 选择持久性卷声明的新大小，然后点 **Expand**：



5. 要验证扩展，请导航到 PVC 的详情页面，并验证 **Capacity** 字段是否具有请求的正确大小。

注意

当基于 Ceph RADOS 块设备 (RBD) 扩展 PVC 时，如果 PVC 尚未附加到 pod，**Condition type** 在 PVC 详情页面中为 **FileSystemResizePending**。挂载卷后，文件系统大小调整成功，并在 **Capacity** 字段中反映新大小。

9.5. 动态置备

9.5.1. 关于动态置备

StorageClass 资源对象描述并分类了可请求的存储，并提供了根据需要为动态置备存储传递参数的方法。StorageClass 也可以作为控制不同级别的存储和访问存储的管理机制。集群管理员（**cluster-admin**）或者存储管理员（**storage-admin**）可以在无需了解底层存储卷资源的情况下，定义并创建用户可以请求的 StorageClass 对象。

OpenShift Container Platform 的持久性卷框架启用了这个功能，并允许管理员为集群提供持久性存储。该框架还可让用户在不了解底层存储架构的情况下请求这些资源。

很多存储类型都可用于 OpenShift Container Platform 中的持久性卷。存储插件可能支持静态置备、动态置备或两种置备类型。

9.5.2. OpenShift Data Foundation 中的动态置备

Red Hat OpenShift Data Foundation 是软件定义的存储，针对容器环境优化。它在 OpenShift Container Platform 上作为操作器运行，为容器提供高度集成和简化的持久性存储管理。

OpenShift Data Foundation 支持各种存储类型，包括：

- 数据库的块存储
- 共享文件存储，用于持续集成、消息传递和数据聚合
- 归档、备份和介质存储的对象存储

第 4 版使用 Red Hat Ceph Storage 来提供支持持久卷的文件、块和对象存储，以及 Rook.io 来管理和编排持久卷和声明的调配。NooBaa 提供对象存储，其多云网关允许在多个云环境中联合对象（作为技术预览使用）。

在 OpenShift Data Foundation 4 中，RADOS 块设备 (RBD) 和 Ceph 文件系统 (CephFS) 的 Red Hat Ceph Storage Container Storage Interface (CSI) 驱动程序处理动态置备请求。当 PVC 请求动态进入时，CSI 驱动程序有以下选项：

- 创建一个具有 ReadWriteOnce (RWO)和 ReadWriteMany (RWX)访问的 PVC，它基于卷模式 **Block** 的 Ceph RBD。
- 创建一个具有 ReadWriteOnce (RWO)访问的 PVC，它基于卷模式 **Filesystem** 的 Ceph RBD。
- 创建一个具有 ReadWriteOnce (RWO)和 ReadWriteMany (RWX)访问的 PVC，该 PVC 基于 CephFS 用于卷模式 **Filesystem**。
- 创建基于 CephFS、NFS 和 RBD 的 ReadWriteOncePod (RWOP)访问的 PVC。使用 RWOP 访问模式，您可以将单个 pod 挂载到单个节点上以读写模式挂载。

重要

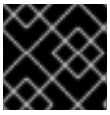
ReadWriteOncePod (RWOP)访问模式是一个技术预览功能。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的更多信息，请参阅技术预览功能支持范围。

判断要使用的驱动程序（RBD 或 CephFS）取决于 **storageclass.yaml** 文件中的条目。

9.5.3. 可用的动态部署插件

OpenShift Container Platform 提供了以下置备程序插件，用于使用集群配置的供应商 API 创建新存储资源的动态部署：

存储类型	provisioner 插件名称	备注
OpenStack Cinder	**kubernetes.io/cinder**	
AWS Elastic Block Store (EBS)	**kubernetes.io/aws-ebs**	当在不同的区中使用多个集群进行动态置备时，使用 **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>**（每个集群的**<cluster_name>** 和 **<cluster_id>** 是唯一的）来标记（tag）每个节点。
AWS Elastic File System (EFS)		动态置备通过 EFS provisioner pod 实现，而不是通过置备程序插件实现。
Azure Disk	**kubernetes.io/azure-disk**	
Azure File	**kubernetes.io/azure-file**	**persistent-volume-binder** ServiceAccount 需要相应的权限，以创建并获取 Secret 来存储 Azure 存储帐户和密钥。
GCE 持久性磁盘 (gcePD)	**kubernetes.io/gce-pd**	在多区（multi-zone）配置中，建议在每个 GCE 项目中运行一个 OpenShift Container Platform 集群，以避免在当前集群没有节点的区域中创建 PV。
VMware vSphere	**kubernetes.io/vsphere-volume**	
Red Hat Virtualization	**csi.ovirt.org**	



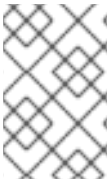
重要

任何选择的置备程序插件还需要根据相关文档为相关的云、主机或者第三方供应商配置。

第 10 章 在目标卷中重新声明空间

已删除的文件或零数据区块有时会在 Ceph 集群上占用存储空间，从而导致报告可用的存储空间不准确。重新声明空间操作通过对目标卷执行以下操作来删除此类差异：

- **fstrim** - 此操作在处于 **Filesystem** 模式的卷上执行，且仅在执行重新声明空间操作时将其挂载到 pod 时。
- **RBD sparsify** - 当卷没有附加到任何 pod 时，会执行此操作，并回收由 4M 分散的零数据占用的空间。



注意

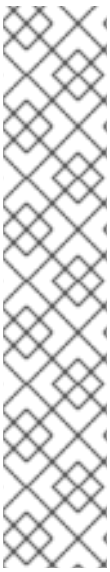
- 重新声明空间操作仅由 Ceph RBD 卷支持。
- 回收空间操作涉及执行时的性能损失。

您可以使用以下方法之一重新声明空间：

- 使用注解 PersistentVolumeClaims 启用重新声明空间操作（推荐使用方法启用重新声明空间操作）
- 使用 ReclaimSpaceJob 启用重新声明空间操作
- 使用 ReclaimSpaceCronJob 启用重新声明空间操作

10.1. 使用注解 PERSISTENTVOLUMECLAIM 启用重新声明空间操作

使用此流程来注解 **PersistentVolumeClaim**，以便它可以根据给定的调度自动调用重新声明空间操作。



注意

- schedule 值的格式与 Kubernetes CronJob 相同，用于设置重复操作请求的和/或间隔。
- 推荐的调度间隔为 **@weekly**。如果调度间隔值为空或无效格式，则默认调度值设置为 **@weekly**。
- 每个调度的操作之间支持的最小间隔至少为 24 小时。例如，**@daily** （每天的 00:00）或 **0 3 *** （每天为 3:00）。
- 在非高峰、维护窗口或工作负载输入/输出应低时调度 **ReclaimSpace** 操作。
- **当修改调度时**，ReclaimSpaceCronJob 会被重新创建。**当注解被删除时，它会被自动删除。**

流程

1. **获取持久性卷声明(PVC)详情。**

```
$ oc get pvc data-pvc
```

```
NAME      STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS                  AGE
data-pvc  Bound    pvc-f37b8582-4b04-4676-88dd-e1b95c6abf74   1Gi        RWO            ocs-
storagecluster-ceph-rbd       20h
```

2. **将注解 reclaimspace.csiaddons.openshift.io/schedule=@monthly 添加到 PVC 以创建 reclaimspacecronjob。**

```
$ oc annotate pvc data-pvc "reclaimspace.csiaddons.openshift.io/schedule=@monthly"
```

```
persistentvolumeclaim/data-pvc annotated
```

3. 验证 **reclaimspacecronjob 是否已创建，格式为 "< pvc-name>-xxxxxxx"。**

```
$ oc get reclaimspacecronjobs.csiaddons.openshift.io
```

```
NAME                 SCHEDULE    SUSPEND   ACTIVE   LASTSCHEDULE   AGE
data-pvc-1642663516  @monthly                       3s
```

4. **修改计划，使其自动运行此任务。**

```
$ oc annotate pvc data-pvc "reclaimspace.csiaddons.openshift.io/schedule=@weekly" --
overwrite=true
```

```
persistentvolumeclaim/data-pvc annotated
```

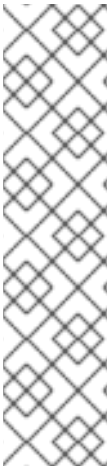
5. 验证 **reclaimspacecronjob 的调度是否已修改。**

```
$ oc get reclaimspacecronjobs.csiaddons.openshift.io
```

```
NAME                 SCHEDULE    SUSPEND   ACTIVE   LASTSCHEDULE   AGE
data-pvc-1642664617  @weekly                        3s
```

10.2. 使用 RECLAIMSPACEJOB 启用重新声明空间操作

ReclaimSpaceJob 是一个命名空间自定义资源(CR)，用于在目标卷上调用重新声明空间操作。这是一个时间方法，用于立即启动回收空间操作。您必须重复创建 ReclaimSpaceJob CR，以便在需要时重复回收空间操作。



注意

- **回收空间操作之间的推荐间隔为 每周。**
- **确保每个操作之间的最小间隔至少为 24 小时。**
- **在非高峰、维护窗口或工作负载输入/输出预期较低时，调度回收空间操作。**

流程

1. **创建并应用以下自定义资源以回收空间操作：**

```
apiVersion: csiaddons.openshift.io/v1alpha1
kind: ReclaimSpaceJob
metadata:
  name: sample-1
spec:
  target:
    persistentVolumeClaim: pvc-1
  timeout: 360
```

其中，

target

指明执行操作的卷目标。

persistentVolumeClaim

PersistentVolumeClaim 的名称。

backOfflimit

指定在将重新声明空间操作 标记为失败 前的最大重试次数。默认值为 6。允许的最大值和最小值分别为 60 和 0。

retryDeadlineSeconds

指定操作可能会停用的时间（以秒为单位），它相对于开始时间。该值必须是正整数。默认值为 600 秒，允许的最大值为 1800 秒。

timeout

指定发送到 CSI 驱动程序的 grpc 请求的超时时间（以秒为单位）。如果没有指定超时值，则默认为全局重新声明空间超时值。timeout 允许的最小值为 60。

2. 在完成操作后删除自定义资源。

10.3. 使用 RECLAIMSPACECRONJOB 启用重新声明空间操作

ReclaimSpaceCronJob 根据给定的调度（如每天、每周等）调用重新声明空间操作。您必须只为持久性卷声明创建 ReclaimSpaceCronJob 一次。CSI-addons 控制器在请求的时间和带有 schedule 属性的间隔创建一个 ReclaimSpaceJob。

注意

- 推荐的调度间隔为 @weekly。
- 每个调度的操作之间的最小间隔应至少为 24 小时。例如，@daily (At 00:00) 或 "0 3 * *" (At 3:00 每天)。
- 在非高峰、维护窗口或工作负载输入/输出应低时调度 ReclaimSpace 操作。

流程

1. 创建并应用以下自定义资源以回收空间操作

```
apiVersion: csiaddons.openshift.io/v1alpha1
kind: ReclaimSpaceCronJob
metadata:
  name: reclaimspacecronjob-sample
spec:
  jobTemplate:
    spec:
      target:
        persistentVolumeClaim: data-pvc
      timeout: 360
  schedule: '@weekly'
  concurrencyPolicy: Forbid
```

其中，

concurrencyPolicy

描述当 ReclaimSpaceJob 调度到 ReclaimSpaceCronJob 时的更改，而之前的 ReclaimSpaceJob 仍然在运行。默认 Forbid 可防止启动新的作业，而 replace 可用于删除可能处于故障状态的正在运行的作业，并创建一个新的作业。

failedJobsHistoryLimit

指定为故障排除保留失败的 ReclaimSpaceJobs 数量。

jobTemplate

指定 ReclaimSpaceJob.spec 结构，用于描述请求的 ReclaimSpaceJob 操作的详情。

successfulJobsHistoryLimit

指定成功 ReclaimSpaceJob 操作的数量。

调度

指定重复操作请求的 和/或间隔，其格式与 Kubernetes CronJob 相同。

2. 在执行重新声明空间操作时，删除 ReclaimSpaceCronJob 自定义资源，或者删除目标 PVC。

10.4. RECLAIM SPACE 操作所需的自定义超时

根据 RBD 卷大小及其数据模式，Reclaim Space Operation 可能会因为 上下文截止时间超过错误而失败。您可以通过增加超时值来避免这种情况。

以下示例通过检查对应 ReclaimSpaceJob 的 -o yaml 来显示失败的状态：

Example

```
Status:
  Completion Time:  2023-03-08T18:56:18Z
  Conditions:
    Last Transition Time:  2023-03-08T18:56:18Z
    Message:               Failed to make controller request: context deadline exceeded
    Observed Generation:   1
    Reason:                failed
    Status:                True
    Type:                  Failed
  Message:                 Maximum retry limit reached
```

```
  Result:              Failed
  Retries:             6
  Start Time:            2023-03-08T18:33:55Z
```

您还可以通过创建以下 configmap，在全局级别上设置自定义超时：

Example

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: csi-addons-config
  namespace: openshift-storage
data:
  "reclaim-space-timeout": "6m"
```

重启 csi-addons operator pod。

```
oc delete po -n openshift-storage -l "app.kubernetes.io/name=csi-addons"
```

所有 Reclaim Space Operations 在上述 configmap 创建后启动，都使用自定义超时。

第 11 章 卷快照

卷快照是集群中特定时间点的存储卷的状态。这些快照有助于更有效地使用存储，不必每次都制作完整的副本，也可用作应用程序开发的构建块。

卷快照类允许管理员指定属于卷快照对象的不同属性。OpenShift Data Foundation 操作器根据使用的平台安装默认卷快照类。Operator 拥有并控制这些默认卷快照类，且无法删除或修改它们。

您可以创建同一持久性卷声明(PVC)的许多快照，但无法调度定期创建快照。

- 对于 CephFS，您可以为每个 PVC 创建最多 100 个快照。
- 对于 RADOS 块设备 (RBD)，您可以为每个 PVC 创建最多 512 个快照。

注意

持久性卷加密现在支持卷快照。

11.1. 创建卷快照

您可以从持久性卷声明 (PVC) 页面或 Volume Snapshots 页面创建卷快照。

先决条件

- 对于一致的快照，PVC 应该处于 Bound 状态，且不使用。确保先停止所有 IO，然后再执行快照。

注意

只有 pod 使用时，OpenShift Data Foundation 才会为 PVC 的卷快照提供崩溃一致性。若要确保应用一致性，请务必先停止正在运行的容器集，以确保快照的一致性，或使用应用提供的任何静默机制来确保快照的一致性。

流程

在持久性卷声明页中显示

1. 从 OpenShift Web 控制台点 Storage → Persistent Volume Claims。
2. 要创建卷快照，请执行以下操作之一：
   - 在所需 PVC 旁边，点 Action 菜单（⋮）→ Create Snapshot。
   - 点击您要创建快照的 PVC，然后点击 Actions → Create Snapshot。
3. 输入卷快照的名称。
4. 从下拉列表中选择 Snapshot Class。
5. 点 Create。您将被重定向到所创建的卷快照的 Details 页面。

从 Volume Snapshots 页面中

1. 从 OpenShift Web 控制台点 Storage → Volume Snapshots。
2. 在 Volume Snapshots 页面中，单击 Create Volume Snapshot。
3. 从下拉列表中选择所需的 项目。
4. 从下拉列表中选择持久性卷声明。
5. 输入快照的名称。
6. 从下拉列表中选择 Snapshot Class。
7. 点 Create。您将被重定向到所创建的卷快照的 Details 页面。

验证步骤

- 进入 PVC 的 Details 页面，然后点击 Volume Snapshots 选项卡查看卷快照列表。验证是否列出了新卷快照。
- 从 OpenShift Web 控制台点 Storage → Volume Snapshots。验证是否列出了新卷快照。
- 等待卷快照处于 Ready 状态。

11.2. 恢复卷快照

恢复卷快照时，会创建一个新的持久性卷声明 (PVC)。恢复的 PVC 独立于卷快照和父 PVC。

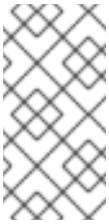
您可以从 PVC 页面或 Volume Snapshots 页面恢复卷快照。

流程

在持久性卷声明页中显示

只有在存在父 PVC 时，才可以从持久性卷声明页面恢复卷快照。

1. 从 OpenShift Web 控制台点 Storage → Persistent Volume Claims。
2. 点击 PVC 名称及卷快照将卷快照恢复为新 PVC。
3. 在 Volume Snapshots 选项卡中，点您要恢复的卷快照旁的 Action 菜单(⋮)。
4. 点 Restore 作为新 PVC。
5. 输入新 PVC 的名称。
6. 选择 Storage Class 名。



注意

对于 Rados 块设备(RBD)，您必须选择一个与父 PVC 池相同的存储类。使用未启用加密的存储类恢复加密 PVC 的快照，反之亦然。

7. 选择您选择的 Access Mode。

重要

ReadOnlyMany(ROX)访问模式是一个开发者预览功能，它受到开发人员预览支持的限制。开发人员预览版本不应在生产环境中运行，且不受红帽客户门户网站问题单管理系统的支持。如果您需要 ReadOnlyMany 功能的帮助，请联络 ocs-devpreview@redhat.com 邮件列表和红帽开发团队成员将根据可用性和工作计划尽快为您提供协助。请参阅 使用新的只读访问模式创建克隆或恢复快照以使用 ROX 访问模式。

8. 可选：对于 RBD，选择 卷模式。

9. 单击 Restore。您将被重定向到新的 PVC 详情页面。

从 Volume Snapshots 页面中

1. 从 OpenShift Web 控制台点 Storage → Volume Snapshots。

2. 在 Volume Snapshots 选项卡中，点您要恢复的卷快照旁的 Action 菜单(⋮)。

3. 点 Restore 作为新 PVC。

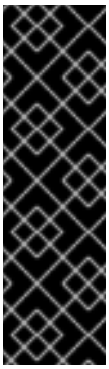
4. 输入新 PVC 的名称。

5. 选择 Storage Class 名。

注意

对于 Rados 块设备(RBD)，您必须选择一个与父 PVC 池相同的存储类。使用未启用加密的存储类恢复加密 PVC 的快照，反之亦然。

6. 选择您选择的 Access Mode。



重要

ReadOnlyMany(ROX)访问模式是一个开发者预览功能，它受到开发人员预览支持的限制。开发人员预览版本不应在生产环境中运行，且不受红帽客户门户网站问题单管理系统的支持。如果您需要 ReadOnlyMany 功能的帮助，请联络 ocs-devpreview@redhat.com 邮件列表和红帽开发团队成员将根据可用性和工作计划尽快为您提供协助。请参阅 使用新的只读访问模式创建克隆或恢复快照以使用 ROX 访问模式。

7. 可选：对于 RBD，选择 卷模式。

8. 单击 Restore。您将被重定向到新的 PVC 详情页面。

验证步骤

- 从 OpenShift Web 控制台点 Storage → Persistent Volume Claims，并确认新 PVC 在 Persistent Volume Claims 页面中列出。
- 等待新 PVC 进入 Bound 状态。

11.3. 删除卷快照

先决条件

- 要删除卷快照，应存在该特定卷快照中使用的卷快照类。

流程

从持久性卷声明页面

1. 从 OpenShift Web 控制台点 Storage → Persistent Volume Claims。

2. 点击具有需要删除卷快照的 PVC 名称。

3. 在 Volume Snapshots 选项卡中，点击所需卷快照旁的 Action 菜单 (⋮) → Delete Volume Snapshot。

从卷快照页面

1. 从 OpenShift Web 控制台点 Storage → Volume Snapshots。

2. 在 Volume Snapshots 页面中，点击所需卷快照菜单 (⋮) → Delete Volume Snapshot 旁。

验证步骤

- 确保 PVC 详情页面的 Volume Snapshots 选项卡中没有删除的卷快照。

- 点 Storage → Volume Snapshots 并确保不会列出删除的卷快照。

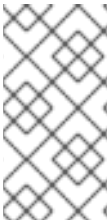
第 12 章 卷克隆

克隆是现有存储卷的副本，用作任何标准卷。您可以创建一个卷克隆，以达到数据的时间副本。持久性卷声明 (PVC) 不能使用不同的大小克隆。您可以为每个 PVC 为 CephFS 和 RADOS 块设备 (RBD) 创建最多 512 个克隆。

12.1. 创建克隆

先决条件

- 源 PVC 必须处于 Bound 状态，且不得处于使用状态。

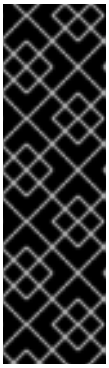


注意

如果 Pod 正在使用 PVC，则不要创建 PVC 克隆。这样做可能会导致数据崩溃，因为 PVC 没有被静默（暂停）。

流程

1. 从 OpenShift Web 控制台点 Storage → Persistent Volume Claims。
2. 要创建克隆，请执行以下操作之一：
   - 在所需的 PVC 旁边，点 Action 菜单 (⋮) → Clone PVC。
   - 点击您要克隆的 PVC，然后点击 Actions → Clone PVC。
3. 输入克隆的名称。
4. 选择您选择的访问模式。



重要

ReadOnlyMany(ROX)访问模式是一个开发者预览功能，它受到开发人员预览支持的限制。开发人员预览版本不应在生产环境中运行，且不受红帽客户门户网站问题单管理系统的支持。如果您需要 ReadOnlyMany 功能的帮助，请联络 ocs-devpreview@redhat.com 邮件列表和红帽开发团队成员将根据可用性和工作计划尽快为您提供协助。请参阅 使用新的只读访问模式创建克隆或恢复快照以使用 ROX 访问模式。

5. 单击 **Clone。您将被重定向到新的 PVC 详情页面。**

6. **等待克隆的 PVC 状态变为 Bound。**

克隆的 PVC 现在可以被 pod 使用。这个克隆的 PVC 独立于其 dataSource PVC。

第 13 章 管理容器存储接口(CSI)组件放置

每个集群由多个专用节点组成，如 infra 和 storage 节点。但是，具有自定义污点的 infra 节点将无法在该节点上使用 OpenShift Data Foundation 持久性卷声明(PVC)。因此，如果要使用这样的节点，可以设置容限以在节点上调出 csi-plugins。

流程

1. 编辑 configmap，为自定义污点添加容限。记住在退出编辑器之前进行保存。

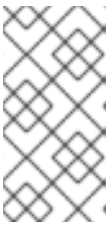
```
$ oc edit configmap rook-ceph-operator-config -n openshift-storage
```

2. 显示 configmap 以检查添加的容限。

```
$ oc get configmap rook-ceph-operator-config -n openshift-storage -o yaml
```

为污点添加的容限的输出示例 nodetype=infra:NoSchedule：

```
apiVersion: v1
data:
[...]
  CSI_PLUGIN_TOLERATIONS: |
    - key: nodetype
      operator: Equal
      value: infra
      effect: NoSchedule
    - key: node.ocs.openshift.io/storage
      operator: Equal
      value: "true"
      effect: NoSchedule
[...]
kind: ConfigMap
metadata:
[...]
```



注意

确保 Tolerations value 字段中的所有非字符串值都带有双引号。例如，值是 true（类型为布尔值）和 1 （类型为 int），则需要输入 "true" 和 "1"。

3. 如果 csi-cephfsplugin- * 和 csi-rbdplugin-* pod 无法自行在 infra 节点上找到，则重启 rook-ceph-operator。

```
$ oc delete -n openshift-storage pod <name of the rook_ceph_operator pod>
```

示例：

```
$ oc delete -n openshift-storage pod rook-ceph-operator-5446f9b95b-jrn2j

pod "rook-ceph-operator-5446f9b95b-jrn2j" deleted
```

验证步骤

验证 **csi-cephfsplugin-* 和 csi-rbdplugin-* pod 正在 infra 节点上运行。**

第 14 章 使用 NFS 创建导出

这部分论述了如何使用 NFS 创建导出，然后可以从 OpenShift 集群外部访问。

按照下面的说明创建导出并从 OpenShift 集群外部访问它们：

- 第 14.1 节 “启用 NFS 功能”
- 第 14.2 节 “创建 NFS 导出”
- 第 14.3 节 “在集群中消耗 NFS 导出”
- 第 14.4 节 “从 OpenShift 集群外部使用 NFS 导出”

14.1. 启用 NFS 功能

要使用 NFS 功能，您需要在创建集群后使用命令行界面(CLI)在存储集群中启用它。您还可以使用用户界面创建存储集群时启用 NFS 功能。

先决条件

- OpenShift Data Foundation 在 openshift-storage 命名空间上安装并运行。
- OpenShift Data Foundation 安装包含一个 CephFilesystem。

流程

- 运行以下命令通过 CLI 启用 NFS 功能：

```
$ oc --namespace openshift-storage patch storageclusters.ocs.openshift.io ocs-storagecluster --type merge --patch '{"spec": {"nfs":{"enable": true}}}'
```

验证步骤

满足以下条件时，NFS 安装和配置已完成：

- 名为 ocs-storagecluster-cephnfs 的 CephNFS 资源的状态为 Ready。
- 检查所有 csi-nfspluginGalaxy pod 是否正在运行：

```
oc -n openshift-storage describe cephnfs ocs-storagecluster-cephnfs
```

```
oc -n openshift-storage get pod | grep csi-nfsplugin
```

输出具有多个 pod。例如：

```
csi-nfsplugin-47qwq                                2/2     Running  0  10s
csi-nfsplugin-77947                                2/2     Running  0  10s
csi-nfsplugin-ct2pm                                2/2     Running  0  10s
csi-nfsplugin-provisioner-f85b75fbb-2rm2w          2/2     Running  0  10s
csi-nfsplugin-provisioner-f85b75fbb-8nj5h          2/2     Running  0  10s
```

14.2. 创建 NFS 导出

NFS 导出通过针对 ocs-storagecluster-ceph-nfs StorageClass 创建持久性卷声明(PVC)。

您可以通过两种方式创建 NFS PVC：

使用 yaml 创建 NFS PVC。

以下是一个 PVC 示例。

注意

对于 NFS 卷，volumeMode: Block 将无法正常工作。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
 name: <desired_name>
spec:
 accessModes:
  - ReadWriteOnce
 resources:
  requests:
   storage: 1Gi
 storageClassName: ocs-storagecluster-ceph-nfs
```

<desired_name>

为 **PVC** 指定名称，如 **my-nfs-export**。

当 PVC 达到 **Bound** 状态后，会创建导出。

从 OpenShift Container Platform Web 控制台创建 **NFS PVC**。

先决条件

- **确保已登录到 OpenShift Container Platform web** 控制台，并且为存储集群启用了 **NFS** 功能。

流程

1. **在 OpenShift Web** 控制台中，点 **Storage → Persistent Volume Claims**
2. **将 Project** 设置为 **openshift-storage**。
3. **点 Create PersistentVolumeClaim**。
   a. **指定 Storage Class,ocs-storagecluster-ceph-nfs**。
   b. **指定 PVC** 名称，例如 **my-nfs-export**。
   c. 选择所需的 **Access Mode**。

d. 根据应用程序要求指定一个 大小。

e. 将卷模式选择为文件系统。

注： NFS PVC 不支持 Block 模式

f. 点 Create 并等待 PVC 处于 Bound 状态。

14.3. 在集群中消耗 NFS 导出

Kubernetes 应用程序 pod 可以通过挂载之前创建的 PVC 来消耗创建的 NFS 导出。

您可以通过两种方式挂载 PVC 之一：

使用 YAML：

以下是使用 第 14.2 节 “创建 NFS 导出” 中创建的示例 PVC 的 pod 示例：

```
apiVersion: v1
kind: Pod
metadata:
 name: nfs-export-example
spec:
 containers:
   - name: web-server
     image: nginx
     volumeMounts:
       - name: nfs-export-pvc
         mountPath: /var/lib/www/html
 volumes:
   - name: nfs-export-pvc
     persistentVolumeClaim:
       claimName: <pvc_name>
       readOnly: false
```

<pvc_name>

指定之前创建的 PVC，例如 my-nfs-export。

使用 OpenShift Container Platform Web 控制台。

流程

1. 在 OpenShift Container Platform web 控制台中进入到 Workloads → Pods。

2. 点 Create Pod 以创建新的应用 pod。

3. 在 metadata 部分下添加一个名称。例如： nfs-export-example，其 namespace 为 openshift-storage。

4. 在 spec: 部分中，使用 image 和 volumeMounts 部分添加 containers: 部分：

```
apiVersion: v1
kind: Pod
metadata:
 name: nfs-export-example
 namespace: openshift-storage
spec:
 containers:
   - name: web-server
     image: nginx
     volumeMounts:
       - name: <volume_name>
         mountPath: /var/lib/www/html
```

例如：

```
apiVersion: v1
kind: Pod
metadata:
 name: nfs-export-example
 namespace: openshift-storage
spec:
 containers:
   - name: web-server
     image: nginx
     volumeMounts:
       - name: nfs-export-pvc
         mountPath: /var/lib/www/html
```

5. 在 spec: 部分，添加 volumes: 部分将 NFS PVC 添加为应用程序 pod 的卷：

```
volumes:
  - name: <volume_name>
    persistentVolumeClaim:
      claimName: <pvc_name>
```

例如：

```
volumes:
  - name: nfs-export-pvc
    persistentVolumeClaim:
      claimName: my-nfs-export
```

14.4. 从 OPENSHIFT 集群外部使用 NFS 导出

OpenShift 集群外部的 NFS 客户端可以挂载由之前创建的 PVC 创建的 NFS 导出。

流程

1. 启用 nfs 标志后，单服务器 CephNFS 由 Rook 部署。您需要获取要在下一步中使用的 nfs-ganesha 服务器的 ceph_nfs 字段的值：

```
$ oc get pods -n openshift-storage | grep rook-ceph-nfs
```

```
$ oc describe pod  <name of the rook-ceph-nfs pod> | grep ceph_nfs
```

例如：

```
$ oc describe pod rook-ceph-nfs-ocs-storagecluster-cephnfs-a-7bb484b4bf-bbdhs | grep
ceph_nfs
  ceph_nfs=my-nfs
```

2. 通过创建 Kubernetes LoadBalancer 服务，在 OpenShift 集群外公开 NFS 服务器。以下示例创建了一个 LoadBalancer 服务，并引用 OpenShift Data Foundation 创建的 NFS 服务器。

```
apiVersion: v1
kind: Service
metadata:
 name: rook-ceph-nfs-ocs-storagecluster-cephnfs-load-balancer
 namespace: openshift-storage
```

```
spec:
 ports:
   - name: nfs
     port: 2049
 type: LoadBalancer
 externalTrafficPolicy: Local
 selector:
   app: rook-ceph-nfs
   ceph_nfs: <my-nfs>
   instance: a
```

将 <my-nfs> 替换为在第 1 步中获取的值。

3. **收集连接信息。外部客户端需要连接到导出的信息来自为 PVC 创建的持久性卷(PV)以及上一步中创建的 LoadBalancer 服务的状态。**

   a. **从 PV 获取共享路径。**

      i. **获取与 NFS 导出的 PVC 关联的 PV 名称：**

```
$ oc get pvc <pvc_name> --output jsonpath='{.spec.volumeName}'
pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d
```

将 <pvc_name> 替换为您自己的 PVC 名称。例如：

```
oc get pvc pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d --output
jsonpath='{.spec.volumeName}'
pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d
```

      ii. **使用前面获取的 PV 名称获取 NFS 导出的共享路径：**

```
$ oc get pv pvc-39c5c467-d9d3-4898-84f7-936ea52fd99d --output
jsonpath='{.spec.csi.volumeAttributes.share}'
/0001-0011-openshift-storage-0000000000000001-ba9426ab-d61b-11ec-9ffd-
0a580a800215
```

   b. **获取 NFS 服务器的入口地址。服务的入口状态可以有多个地址。选择外部客户端所需的使用方法。在以下示例中，只有一个地址：主机名 ingress-id.somedomain.com。**

```
$ oc -n openshift-storage get service rook-ceph-nfs-ocs-storagecluster-cephnfs-load-
balancer --output jsonpath='{.status.loadBalancer.ingress}'
[{"hostname":"ingress-id.somedomain.com"}]
```

4. **使用上一步中的共享路径和入口地址连接外部客户端。以下示例将导出挂载到客户端的目录路径 /export/mount/path ：**

```
$ mount -t nfs4 -o proto=tcp ingress-id.somedomain.com:/0001-0011-openshift-storage-
0000000000000001-ba9426ab-d61b-11ec-9ffd-0a580a800215 /export/mount/path
```

如果这仍然无法正常工作，则可能是 Kubernetes 环境仍然需要时间来配置网络资源，以允许到 NFS 服务器入口。

第 15 章 注解加密的 RBD 存储类

从 OpenShift Data Foundation 4.14 开始，当 OpenShift 控制台创建启用加密的 RADOS 块设备 (RBD)存储类时，会自动设置注解。但是，您需要在升级到 OpenShift Data Foundation 版本 4.14 之前为之前创建的任何加密的 RBD 存储类添加注解 cdi.kubevirt.io/clone-strategy=copy。这可让客户数据集成 (CDI) 使用主机辅助克隆，而不是默认的智能克隆。

用于访问加密卷的密钥与创建卷的命名空间相关联。当将加密卷克隆到新命名空间时，如置备新的 OpenShift Virtualization 虚拟机时，必须创建一个新卷，然后源卷的内容必须复制到新卷中。如果正确注解存储类，则会自动触发此行为。