Red Hat OpenShift Data Foundation 4.16

ストレージリソースの管理および割り当て

スナップショットおよびクローンを含む、OpenShift Data Foundation のコアサービスおよびホスト型アプリケーションにストレージを割り当てる方法の手順

Last Updated: 2024-08-17

Red Hat OpenShift Data Foundation 4.16 ストレージリソースの管理および割り当て

スナップショットおよびクローンを含む、OpenShift Data Foundation のコアサービスおよびホスト型アプリケーションにストレージを割り当てる方法の手順

法律上の通知

概要

このドキュメントでは、Red Hat OpenShift Data Foundation でコアサービスおよびホスト型アプリケーションにストレージを割り当てる方法を説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、用語の置き換えは、今後の複数のリリースにわたって段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があれば、ぜひお知らせください。

フィードバックを送信するには、Bugzilla チケットを作成します。

1. Bugzilla の Web サイトに移動します。
2. **Component** セクションで、**documentation** を選択します。
3. **Description** フィールドに、ドキュメントの改善に向けたご提案を記入してください。ドキュメントの該当部分へのリンクも記載してください。
4. **Submit Bug** をクリックします。

第1章 概要

このドキュメントでは、ストレージを作成し、設定し、Red Hat OpenShift Data Foundation のコアサービスまたはホスト型アプリケーションに割り当てる方法を説明します。

- 2章*ストレージクラス* カスタムのストレージクラスを作成する方法を説明します。
- 3章*ブロックプール* ブロックプールの作成、更新、および削除方法に関する情報を提供します。
- 4章*OpenShift Container Platform サービスのストレージの設定* コアとなる OpenShift Container Platform サービスに OpenShift Data Foundation を使用する方法を説明します。
- 6章*OpenShift Data Foundation を使用した OpenShift Container Platform アプリケーションのサポート* OpenShift Data Foundation を使用するように OpenShift Container Platform アプリケーションを設定する方法を提供します。
- 既存の外部の OpenShift Data Foundation クラスターへのファイルおよびオブジェクトストレージの追加
- 8章*Red Hat OpenShift Data Foundation に専用のワーカーノードを使用する方法* Red Hat OpenShift Data Foundation に専用のワーカーノードを使用する方法を提供します。
- 9章*永続ボリューム要求の管理* は、永続ボリューム要求の要求の管理とそれらの要求への対応の自動化に関する情報を提供します。
- 10章 *ターゲットボリュームのスペースを再利用* では、実際に使用可能なストレージスペースを回収する方法を説明しています。
- 12章*ボリュームスナップショット* ボリュームスナップショットを作成し、復元し、削除する方法を説明します。
- 13章 *ボリュームのクローン作成* ボリュームのクローンを作成する方法を説明します。
- 14章*Container Storage Interface (CSI) コンポーネントの配置の管理* 容認を設定してノードでコンテナーストレージのインターフェイスコンポーネントを起動する方法を提供します。

第2章 ストレージクラス

OpenShift Data Foundation Operator は、使用されるプラットフォームに応じてデフォルトのストレージクラスをインストールします。このデフォルトストレージクラスは Operator によって所有され、制御されるため、削除したり変更したりすることはできません。ただし、カスタムストレージクラスを作成して他のストレージリソースを使用したり、アプリケーションに異なる動作を提供したりできます。

注記

カスタムストレージクラスは、**外部モード** の OpenShift Data Foundation クラスターではサポートされません。

2.1. ストレージクラスおよびプールの作成

既存のプールを使用してストレージクラスを作成するか、ストレージクラスの作成中にストレージクラスの新規プールを作成できます。

前提条件

- OpenShift Container Platform の Web コンソールにログインしており、OpenShift Data Foundation クラスターが **Ready** 状態にあることを確認します。

手順

1. **Storage → StorageClasses** をクリックします。
2. **Create Storage Class**をクリックします。
3. ストレージクラスの **Name** および **Description** を入力します。
4. **Reclaim Policy** は、デフォルトオプションとして **Delete** に設定されています。この設定を使用します。
   回収ポリシーをストレージクラスで **Retain** に変更すると、永続ボリューム要求 (PVC) を削除した後でも、永続ボリューム (PV) は **Released** 状態のままになります。
5. **ボリュームバインディングモード** は、デフォルトオプションとして **WaitForConsumer** に設定されています。
   Immediate オプションを選択すると、PVC の作成時に PV がすぐに作成されます。
6. 永続ボリュームをプロビジョニングするためのプラグインとして、**RBD** または **CephFS Provisioner** を選択します。
7. ワークロードの **Storage system** を選択します。
8. 一覧から既存の **ストレージプール** を選択するか、新規プールを作成します。

注記

双方向レプリケーションデータ保護ポリシーは、デフォルト以外の RBD プールでのみサポートされます。追加のプールを作成することで双方向レプリケーションを使用できます。replica 2 プールのデータの可用性と整合性に関する考慮事項は、ナレッジベースのカスタマーソリューション記事 を参照してください。

新規プールの作成

a. **Create New Pool** をクリックします。

b. **Pool name** を入力します。

c. Data Protection Policy として **2-way-Replication** または **3-way-Replication** を選択します。

d. データを圧縮する必要がある場合は、**Enable compression** を選択します。
圧縮を有効にするとアプリケーションのパフォーマンスに影響がある可能性があり、書き込まれるデータがすでに圧縮または暗号化されている場合は効果的ではない可能性があります。圧縮を有効にする前に書き込まれたデータは圧縮されません。

e. **Create** をクリックして新規ストレージプールを作成します。

f. プールの作成後に **Finish** をクリックします。

9. オプション: **Enable Encryption** のチェックボックスを選択します。

10. **Create** をクリックしてストレージクラスを作成します。

2.2. 永続ボリュームの暗号化のためのストレージクラス

永続ボリューム (PV) 暗号化は、テナント (アプリケーション) の分離および機密性を保証します。PV 暗号化を使用する前に、PV 暗号化のストレージクラスを作成する必要があります。永続ボリュームの暗号化は RBD PV の場合にのみ利用できます。

OpenShift Data Foundation は、HashiCorp Vault および Thales CipherTrust Manager での暗号化パスフレーズの保存をサポートしています。永続的なボリュームの暗号化のために、外部の鍵管理システム (KMS) を使用して、暗号化対応のストレージクラスを作成することができます。ストレージクラスを作成する前に、KMS へのアクセスを設定する必要があります。

注記

PV 暗号化には、有効な Red Hat OpenShift Data Foundation Advanced サブスクリプションが必要です。詳細は、OpenShift Data Foundation サブスクリプションに関するナレッジベースの記事 を参照してください。

2.2.1. Key Management System (KMS) のアクセス設定

ユースケースに基づいて、次のいずれかの方法を使用して KMS へのアクセスを設定する必要があります。

- **vaulttokens** の使用: ユーザーはトークンを使用して認証できるようになります。
- **Thales CipherTrust Manager** の使用: Key Management Interoperability Protocol (KMIP) を使用します。
- **vaulttenantsa** (テクノロジープレビュー) の使用: ユーザーは **serviceaccounts** を使用して **Vault** で認証できるようになります。

重要

vaulttenantsa を使用した KMS へのアクセスはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

2.2.1.1. vaulttokens を使用した KMS へのアクセス設定

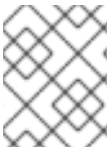
前提条件

- OpenShift Data Foundation クラスターは **Ready** 状態である。
- 外部の鍵管理システム (KMS) で、以下を実行している。
  - トークンのあるポリシーが存在し、**Vault** のキー値のバックエンドパスが有効になっていることを確認します。
  - **Vault** サーバーで署名済みの証明書を使用していることを確認します。

手順

テナントの namespace にシークレットを作成します。

1. OpenShift Container Platform Web コンソールで、**Workloads → Secrets**に移動します。
2. **Create → Key/value secret**をクリックします。
3. **Secret Name** に **ceph-csi-kms-token** と入力します。
4. **Key** に **token** と入力します。
5. **Value** を入力します。
   これは Vault のトークンです。**Browse** をクリックしてトークンが含まれるファイルを選択し、アップロードするか、テキストボックスにトークンを直接入力します。
6. **Create** をクリックします。



注記

トークンは、**ceph-csi-kms-token** を使用するすべての暗号化された PVC が削除された後にのみ削除できます。

2.2.1.2. Thales CipherTrust Manager を使用した KMS へのアクセスの設定

前提条件

1. KMIP クライアントが存在しない場合は作成します。ユーザーインターフェイスから、**KMIP → Client Profile → Add Profile** を選択します。
   a. プロファイルの作成中に、**CipherTrust** のユーザー名を **Common Name** フィールドに追加します。

2. **KMIP → Registration Token → New Registration Token** に移動してトークンを作成します。次のステップのためにトークンをコピーしておきます。

3. クライアントを登録するために、**KMIP → Registered Clients → Add Client** に移動します。**Name** を指定します。前のステップの **Registration Token** を貼り付けて、**Save** をクリックします。

4. **Save Private Key** と **Save Certificate** をクリックして、それぞれ秘密鍵とクライアント証明書をダウンロードします。

5. 新しい KMIP インターフェイスを作成するために、**Admin Settings → Interfaces → Add Interface** に移動します。

   a. **KMIP Key Management Interoperability Protocol** を選択し、**Next** をクリックします。

   b. 空いている **Port** を選択します。

   c. **Network Interface** は **all** を選択します。

   d. **Interface Mode** は **TLS, verify client cert, user name taken from client cert, auth request is optional** を選択します。

   e. (オプション) 物理削除を有効にして、鍵が削除されたときにメタデータとマテリアルの両方を削除できます。これはデフォルトでは無効にされます。

   f. 使用する CA を選択し、**Save** をクリックします。

6. サーバー CA 証明書を取得するために、新しく作成されたインターフェイスの右側にあるアクションメニュー (⋮) をクリックし、**Download Certificate** をクリックします。

手順

1. storageclass 暗号化のキー暗号化キー (KEK) として機能するキーを作成するには、次の手順に従います。

   a. **Keys → Add Key** に移動します。

   b. **Key Name** を入力します。

   c. **Algorithm** と **Size** をそれぞれ **AES** と **256** に設定します。

   d. **Create a key in Pre-Active state**を有効にして、アクティベーションの日時を設定します。

   e. **Key Usage** で **Encrypt** と **Decrypt** が有効になっていることを確認します。

   f. 新しく作成した鍵の ID をコピーして、デプロイメント中に一意の識別子として使用します。

2.2.1.3. vaulttenantsa を使用した KMS へのアクセスの設定

前提条件

- OpenShift Data Foundation クラスターは **Ready** 状態である。
- 外部の鍵管理システム (KMS) で、以下を実行している。

   a. ポリシーが存在し、Vault のキー値のバックエンドパスが有効になっていることを確認します

- ポリシーが存在し、Vault のキー値のバックエンドパスが有効になっていることを確認します。
- Vault サーバーで署名済みの証明書を使用していることを確認します。

- 以下のようにテナント namespace に以下の serviceaccount を作成します。

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

手順

OpenShift Data Foundation が **Vault** で認証して使用を開始する前に、Kubernetes 認証方法を設定する必要があります。以下の手順では、OpenShift Data Foundation が **Vault** で認証できるように、**serviceAccount**、**ClusterRole**、および **ClusterRoleBinding** を作成し、設定します。

1. 以下の YAML を Openshift クラスターに適用します。

```
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. serviceaccount トークンおよび CA 証明書のシークレットを作成します。

```
$ 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. シークレットからトークンと CA 証明書を取得します。

```
$ 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. OpenShift クラスターエンドポイントを取得します。

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

5. 前の手順で収集した情報を使用して、以下のように Vault で Kubernetes 認証方法を設定します。

```
$ 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. テナント namespace の Vault にロールを作成します。

```
$ 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 は、OpenShift Data Foundation が Vault を検索するデフォルトのロール名です。OpenShift Data Foundation クラスターのテナント namespace 内のデフォルトサービスアカウント名は、**ceph-csi-vault-sa** です。これらのデフォルト値は、テナント namespace に ConfigMap を作成して上書きできます。

デフォルト名の上書きに関する詳細は、Overriding Vault connection details using tenant ConfigMap を参照してください。

YAML 例

- PV 暗号化の **vaulttenantsa** メソッドを使用する storageclass を作成するには、既存の ConfigMap を編集するか、Vault との接続を確立するために必要なすべての情報を保持する **csi-kms-connection-details** という名前の ConfigMap を作成する必要があります。
以下の yaml のサンプルを使用して、**csi-kms-connection-detail** ConfigMap を更新または作成できます。

```
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	Vault での認証にサービスアカウントを使用するには、**vaulttenantsa** に設定します。
vaultAddress	ポート番号を持つ vault サーバーのホスト名または IP アドレス。
vaultTLSServerName	(オプション)vault の TLS サーバー名
vaultAuthPath	(オプション)Vault で kubernetes 認証メソッドが有効なパス。デフォルトのパスは **kubernetes** です。auth メソッドが **kubernetes** 以外のパスで有効になっている場合は、この変数を **"/v1/auth/<path>/login"** として設定する必要があります。
vaultAuthNamespace	(オプション) kubernetes 認証メソッドが有効な Vault namespace。
vaultNamespace	(オプション) キーの保存に使用されるバックエンドパスが存在する Vault namespace
vaultBackendPath	暗号化キーが保存される Vault のバックエンドパス
vaultCAFromSecret	Vault の CA 証明書が含まれる OpenShift Data Foundation クラスターのシークレット
vaultClientCertFromSecret	Vault のクライアント証明書を含む OpenShift Data Foundation クラスターのシークレット
vaultClientCertKeyFromSecret	Vault のクライアントプライベートキーが含まれる OpenShift Data Foundation クラスターのシークレット

tenantSA Name	(オプション) テナント namespace のサービスアカウント名。デフォルト値は **ceph-csi-vault-sa** です。別の名前を使用する場合は、この変数を適切に設定する必要があります。

2.2.2. 永続ボリュームの暗号化のためのストレージクラスの作成

前提条件

ユースケースに基づいて、以下のいずれかの KMS へのアクセスを確実に設定する必要があります。

- **vaulttokens** の使用: **vaulttokens** を使用した KMS へのアクセスの設定 の説明に従ってアクセスを設定してください。
- **vaulttenantsa** の使用 (テクノロジープレビュー): **vaulttenantsa** を使用した KMS へのアクセスの設定 の説明に従ってアクセスを設定してください。
- Thales CipherTrust Manager の使用 (KMIP を使用): Thales CipherTrust Manager を使用した KMS へのアクセスの設定 の説明に従ってアクセスを設定してください。
- (Azure プラットフォームのユーザーのみ) Azure Vault の使用 [テクノロジープレビュー]: 次の手順に従って、クライアント認証を設定し、Azure からクライアント認証情報を取得してください。
  1. Azure Vault を作成します。詳細は、Microsoft 製品ドキュメントの Quickstart: Create a key vault using the Azure portal を参照してください。
  2. 証明書ベースの認証を使用してサービスプリンシパルを作成します。詳細は、Microsoft 製品ドキュメントの Create an Azure service principal with Azure CLI を参照してください。
  3. Azure Key Vault のロールベースのアクセス制御 (RBAC) を設定します。詳細は、Enable Azure RBAC permissions on Key Vault を参照してください。

手順

1. OpenShift Web コンソールで、**Storage → StorageClasses** に移動します。
2. **Create Storage Class**をクリックします。
3. ストレージクラスの **Name** および **Description** を入力します。
4. **Reclaim Policy** は **Delete** または **Retain** のいずれかを選択します。デフォルトでは、**Delete** が選択されます。
5. **Volume binding モード**に **Immediate** または **WaitForFirstConsumer** を選択します。**WaitForConsumer** はデフォルトオプションとして設定されます。
6. 永続ボリュームをプロビジョニングするのに使用されるプラグインである **RBD Provisioner openshift-storage.rbd.csi.ceph.com** を選択します。
7. ボリュームデータが保存される **Storage Pool** をリストから選択するか、新規プールを作成します。
8. **Enable encryption** チェックボックスを選択します。

- 次のいずれかのオプションを選択し、KMS 接続の詳細を設定します。
  - **Choose existing KMS connection**: ドロップダウンリストから既存の KMS 接続を選択します。このリストは、**csi-kms-connection-details** ConfigMap で利用可能な接続の詳細から設定されます。
    a. ドロップダウンから **Provider** を選択します。
    b. リストから特定のプロバイダーの **Key service** を選択します。
  - **Create new KMS connection**: これは、**vaulttoken** と **Thales CipherTrust Manager (using KMIP)** にのみ適用されます。
    a. 次のいずれかの **Key Management Service Provider** を選択し、必要な詳細情報を入力します。
      - **Vault**
        i. 一意の **Connection Name**、Vault サーバーのホストの **Address** ('https://<ホスト名または IP>')、ポート番号、および **Token** を入力します。
        ii. **Advanced Settings** をデプロイメントして、**Vault** 設定に基づいて追加の設定および証明書の詳細を入力します。
          A. OpenShift Data Foundation 専用かつ特有のキーと値のシークレットパスを **Backend Path** に入力します。
          B. オプション: **TLS Server Name** および **Vault Enterprise Namespace** を入力します。
          C. PEM でエンコードされた、該当の証明書ファイルをアップロードし、**CA Certificate**、**Client Certificate**、および **Client Private Key** を指定します。
          D. **Save** をクリックします。
      - **Thales CipherTrust Manager (KMIP を使用)**
        i. 一意の **Connection Name** を入力します。
        ii. **Address** および **Port** セクションで、Thales CipherTrust Manager の IP と、KMIP インターフェイスが有効になっているポートを入力します。たとえば、**Address**: 123.34.3.2、**Port**: 5696 などです。
        iii. **Client Certificate**、**CA certificate**、および **Client Private Key** をアップロードします。
        iv. 上記で生成された暗号化および復号化に使用する鍵の **Unique Identifier** を入力します。
        v. **TLS Server** フィールドはオプションであり、KMIP エンドポイントの DNS エントリーがない場合に使用します。たとえば、**kmip_all_<port>.ciphertrustmanager.local** などです。
      - **Azure Key Vault** (テクノロジープレビュー) (Azure プラットフォーム上の Azure ユーザーのみ)
        クライアント認証の設定とクライアント認証情報の取得については、**Microsoft Azure を使用した OpenShift Data Foundation のデプロイ** ガイ

ドの OpenShift Data Foundation クラスターの作成 セクションの「前提条件」を参照してください。

i. プロジェクト内のキー管理サービスの一意の **Connection name** を入力します。

ii. **Azure Vault URL** を入力します。

iii. **Client ID** を入力します。

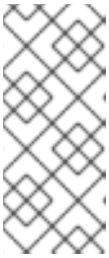
iv. **Tenant ID** を入力します。

v. **Certificate** ファイルを **.PEM** 形式でアップロードします。証明書ファイルには、クライアント証明書と秘密鍵が含まれている必要があります。

b. **Save** をクリックします。

c. **Create** をクリックします。

9. HashiCorp Vault 設定により、バックエンドパスによって使用されるキー/値 (KV) シークレットエンジン API バージョンの自動検出が許可されない場合は、ConfigMap を編集して **vaultBackend** パラメーターを追加します。



注記

vaultBackend は、バックエンドパスに関連付けられた KV シークレットエンジン API のバージョンを指定するために configmap に追加されるオプションのパラメーターです。値がバックエンドパスに設定されている KV シークレットエンジン API バージョンと一致していることを確認します。一致しない場合は、永続ボリューム要求 (PVC) の作成時に失敗する可能性があります。

a. 新規に作成されたストレージクラスによって使用されている encryptionKMSID を特定します。

i. OpenShift Web コンソールで、**Storage → Storage Classes** に移動します。

ii. **Storage class** 名 → **YAML** タブをクリックします。

iii. ストレージクラスによって使用されている **encryptionKMSID** を取得します。
以下に例を示します。

```
encryptionKMSID: 1-vault
```

b. OpenShift Web コンソールで **Workloads → ConfigMaps** に移動します。

c. KMS 接続の詳細を表示するには、**csi-kms-connection-details** をクリックします。

d. ConfigMap を編集します。

i. アクションメニュー (⋮) → **Edit ConfigMap** をクリックします。

ii. 以前に特定した **encryptionKMSID** に設定されるバックエンドに応じて、**vaultBackend** パラメーターを追加します。
KV シークレットエンジン API バージョン 1 の場合は **kv** を、KV シークレットエンジン API バージョン 2 の場合は **kv-v2** を、それぞれ割り当てることができます。

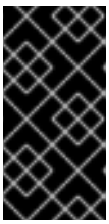
以下に例を示します。

```
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. 保存をクリックします。

次のステップ

- ストレージクラスを使用して、暗号化された永続ボリュームを作成できます。詳細は、永続ボリューム要求の管理 を参照してください。



重要

Red Hat はテクノロジーパートナーと連携して、このドキュメントをお客様へのサービスとして提供します。ただし、Red Hat では、HashiCorp 製品のサポートを提供していません。この製品に関するテクニカルサポートについては、HashiCorp にお問い合わせください。

2.2.2.1. テナント ConfigMap を使用した Vault 接続の詳細の上書き

Vault 接続の詳細は、**openshift-storage** namespace の **csi-kms-connection-details** ConfigMap で設定された値とは異なる設定オプションを使用して、OpenShift namespace に ConfigMap を作成することにより、テナントごとに再設定できます。ConfigMap はテナント namespace に配置する必要があります。テナント namespace の ConfigMap の値は、その namespace で作成される暗号化された永続ボリュームの **csi-kms-connection-details** ConfigMap に設定された値を上書きします。

手順

1. テナント namespace にあることを確認します。
2. **Workloads → ConfigMaps** をクリックします。
3. **Create ConfigMap** をクリックします。
4. yaml ファイルの例を以下に示します。指定のテナント namespace に過剰に使用される値は、以下に示すように **data** セクションで指定できます。

```
---
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. yaml を編集したら、**Create** をクリックします。

2.3. レプリカを 1 つ含まれるストレージクラス

アプリケーションで使用するレプリカが 1 つ含まれるストレージクラスを作成できます。これにより、冗長なデータコピーが回避され、アプリケーションレベルでの復元管理が可能になります。

警告

この機能を有効にすると、データレプリケーションのない単一のレプリカプールが作成され、アプリケーションに独自のレプリケーションがない場合は、データ損失、データ破損、および潜在的なシステム不安定のリスクが増加します。OSD が失われると、この機能を回復するのに長期間の停止を伴う手順が必要になります。すべてのアプリケーションのデータが失われる可能性があります。OSD に障害が発生した場合は、すべてのアプリケーションを再作成する必要があります。

手順

1. 次のコマンドを使用して単一レプリカ機能を有効にします。

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

2. **storagecluster** が **Ready** 状態であることを確認します。

```
$ oc get storagecluster
```

出力例:

```
NAME                 AGE   PHASE   EXTERNAL   CREATED AT             VERSION
ocs-storagecluster   10m   Ready              2024-02-05T13:56:15Z   4.15.0
```

3. 各障害ドメインに新しい **cephblockpool** が作成されます。**cephblockpools** が **Ready** 状態であることを確認します。

```
$ oc get cephblockpools
```

出力例:

```
NAME                                 PHASE
ocs-storagecluster-cephblockpool             Ready
ocs-storagecluster-cephblockpool-us-east-1a   Ready
ocs-storagecluster-cephblockpool-us-east-1b   Ready
ocs-storagecluster-cephblockpool-us-east-1c   Ready
```

4. 新しいストレージクラスが作成されたことを確認します。

```
$ oc get storageclass
```

出力例:

```
NAME                                 PROVISIONER                        RECLAIMPOLICY
VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
gp2 (default)                        kubernetes.io/aws-ebs              Delete
WaitForFirstConsumer   true                   104m
gp2-csi                              ebs.csi.aws.com                    Delete
WaitForFirstConsumer   true                   104m
gp3-csi                              ebs.csi.aws.com                    Delete
WaitForFirstConsumer   true                   104m
ocs-storagecluster-ceph-non-resilient-rbd   openshift-storage.rbd.csi.ceph.com      Delete
WaitForFirstConsumer   true                   46m
ocs-storagecluster-ceph-rbd          openshift-storage.rbd.csi.ceph.com      Delete
Immediate              true                   52m
ocs-storagecluster-cephfs            openshift-storage.cephfs.csi.ceph.com   Delete
Immediate              true                   52m
openshift-storage.noobaa.io          openshift-storage.noobaa.io/obc         Delete
Immediate              false                  50m
```

5. 新しい OSD Pod (3 つの osd-prepare Pod と 3 つの追加 Pod) が作成されます。新しい OSD Pod が **Running** 状態であることを確認します。

```
$ oc get pods | grep osd
```

出力例:

```
rook-ceph-osd-0-6dc76777bc-snhnm                                  2/2     Running     0          9m50s
rook-ceph-osd-1-768bdfdc4-h5n7k                                   2/2     Running     0          9m48s
rook-ceph-osd-2-69878645c4-bkdlq                                  2/2     Running     0          9m37s
rook-ceph-osd-3-64c44d7d76-zfxq9                                  2/2     Running     0          5m23s
rook-ceph-osd-4-654445b78f-nsgjb                                  2/2     Running     0          5m23s
rook-ceph-osd-5-5775949f57-vz6jp                                  2/2     Running     0          5m22s
rook-ceph-osd-prepare-ocs-deviceset-gp2-0-data-0x6t87-59swf       0/1     Completed   0
10m
rook-ceph-osd-prepare-ocs-deviceset-gp2-1-data-0klwr7-bk45t       0/1     Completed   0
10m
rook-ceph-osd-prepare-ocs-deviceset-gp2-2-data-0mk2cz-jx7zv       0/1     Completed   0
10m
```

2.3.1. 単一レプリカから OSD が失われた後の回復

レプリカ 1 (単一レプリカを持つストレージクラス) を使用する場合は、OSD が失われた場合に確実にデータが損失します。失われた OSD は障害状態になります。OSD 喪失後に回復するには、次の手順に従います。

手順

レプリカ 1 からデータが失われた後、アプリケーションを再び実行するには、次の回復手順に従います。まず、障害が発生している OSD があるドメインを特定する必要があります。

1. 障害が発生している OSD がどの障害ドメインにあるかわかっている場合は、次のコマンドを実行して、次の手順に必要な正確な **replica1-pool-name** を取得します。障害が発生している OSD がどこにあるかわからない場合は、ステップ 2 に進みます。

```
$ oc get cephblockpools
```

出力例:

```
NAME                                 PHASE
ocs-storagecluster-cephblockpool             Ready
ocs-storagecluster-cephblockpool-us-south-1   Ready
ocs-storagecluster-cephblockpool-us-south-2   Ready
ocs-storagecluster-cephblockpool-us-south-3   Ready
```

次の手順で使用するために、該当する障害ドメイン名をコピーし、ステップ 4 に進みます。

2. 障害が発生している OSD を見つけるには、**Error** 状態または **CrashLoopBackOff** 状態の OSD Pod を見つけます。

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

3. 障害が発生した OSD があった replica-1 プールを特定します。

   a. 障害が発生した OSD が実行されていたノードを特定します。

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

   b. 障害が発生した OSD が実行されていたノードの failureDomainLabel を特定します。

```
failure_domain_label=$(oc get storageclass ocs-storagecluster-ceph-non-resilient-rbd -o yaml | grep domainLabel |head -1 |awk -F':' '{print $2}')
```

```
failure_domain_value=$”(oc get pods $failed_osd_id -oyaml |grep topology-location-zone |awk ‘{print $2}’)”
```

   出力に、障害が発生している OSD がある replica-1 プール名が表示されます。次に例を示します。

```
replica1-pool-name= "ocs-storagecluster-cephblockpool-$failure_domain_value”
```

   $failure_domain_value は failureDomainName です。

4. replica-1 プールを削除します。

   a. toolbox Pod に接続します。

```
toolbox=$(oc get pod -l app=rook-ceph-tools -n openshift-storage -o
jsonpath='{.items[*].metadata.name}')

oc rsh $toolbox -n openshift-storage
```

b. replica-1 プールを削除します。コマンドで、replica-1 プール名を 2 回入力する必要があることに注意してください。次に例を示します。

```
ceph osd pool rm <replica1-pool-name> <replica1-pool-name> --yes-i-really-really-
mean-it
```

replica1-pool-name は、先ほど特定した障害ドメイン名に置き換えます。

5. デバイスの置き換え ガイドのプラットフォームに応じて、「動作するストレージデバイスまたは障害のあるストレージデバイスの置き換え」セクションの手順に従って、障害が発生している OSD をパージします。

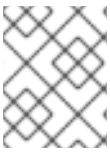
6. rook-ceph operator を再起動します。

```
$ oc delete pod -l rook-ceph-operator -n openshift-storage
```

7. 影響を受けるアプリケーションをそのアベイラビリティーゾーン内に再作成して、同じ名前の新しいプールの使用を開始します。

第3章 ブロックプール

OpenShift Data Foundation Operator は、使用されるプラットフォームに応じてデフォルトのストレージプールのセットをインストールします。これらのデフォルトストレージプールは Operator によって所有され、制御されるため、削除したり変更したりすることはできません。



注記

外部モード の OpenShift Data Foundation クラスターでは、複数のブロックプールはサポートされません。

3.1. 内部モードでのブロックプールの管理

OpenShift Container Platform を使用して、以下の機能を提供するストレージクラスにマップする複数のカスタムストレージプールを作成できます。

- それぞれに高可用性のあるアプリケーションを有効にして、2 つのレプリカを持つ永続ボリュームを使用できるようにします。これにより、アプリケーションのパフォーマンスが向上する可能性があります。
- 圧縮が有効にされているストレージクラスを使用して永続ボリューム要求の領域を節約します。

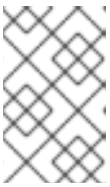
3.1.1. ブロックプールの作成

前提条件

- 管理者として OpenShift Container Platform Web コンソールにログインしている。

手順

1. **Storage → Data Foundation**をクリックします。
2. **Storage systems** タブでストレージシステムを選択し、**BlockPools** タブをクリックします。
3. **Create Block Pool** をクリックします。
4. **Pool name** を入力します。



注記

双方向レプリケーションデータ保護ポリシーの使用は、デフォルトプールではサポートされていません。ただし、追加のプールを作成する場合は、双方向レプリケーションを使用できます。

5. **Data protection policy**に **2-way Replication** または **3-way Replication** のいずれかを選択します。
6. オプション: データを圧縮する必要がある場合は、**Enable compression** のチェックボックスを選択します。
圧縮を有効にするとアプリケーションのパフォーマンスに影響がある可能性があり、書き込まれるデータがすでに圧縮または暗号化されている場合は効果的ではない可能性があります。圧縮を有効にする前に書き込まれたデータは圧縮されません。

7. **Create** をクリックします。

3.1.2. 既存プールの更新

前提条件

- 管理者として OpenShift Container Platform Web コンソールにログインしている。

手順

1. **Storage → Data Foundation**をクリックします。
2. **Storage systems** タブでストレージシステムを選択し、**BlockPools** をクリックします。
3. 更新するプールの末尾でアクションメニュー (⋮) をクリックします。
4. **Edit Block Pool** をクリックします。
5. 以下のようにフォームの詳細を変更します。

注記

双方向レプリケーションデータ保護ポリシーの使用は、デフォルトプールではサポートされていません。ただし、追加のプールを作成する場合は、双方向レプリケーションを使用できます。

   a. **Data protection policy** を 2-way Replication または 3-way Replication のいずれかに変更します。

   b. 圧縮オプションを有効または無効にします。
   圧縮を有効にするとアプリケーションのパフォーマンスに影響がある可能性があり、書き込まれるデータがすでに圧縮または暗号化されている場合は効果的ではない可能性があります。圧縮を有効にする前に書き込まれたデータは圧縮されません。

6. **Save** をクリックします。

3.1.3. プールの削除

以下の手順を使用して、OpenShift Data Foundation のプールを削除します。

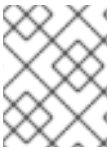
前提条件

- 管理者として OpenShift Container Platform Web コンソールにログインしている。

手順

1. .**Storage → Data Foundation**をクリックします。
2. **Storage systems** タブでストレージシステムを選択し、**BlockPools** タブをクリックします。
3. 削除するプールの末尾でアクションメニュー (⋮) をクリックします。
4. **Delete Block Pool** をクリックします。

5. **Delete** をクリックしてプールの削除を確認します。



注記

プールが PVC にバインドされていると、削除できません。このアクティビティーを実行する前に、すべてのリソースの割り当てを解除する必要があります。

第4章 OPENSHIFT CONTAINER PLATFORM サービスのストレージの設定

OpenShift Data Foundation を使用して、以下のような OpenShift Container Platform サービスのストレージを提供できます。

- OpenShift イメージレジストリー
- OpenShift モニタリング
- OpenShift ロギング (Loki)

これらのサービスのストレージを設定するプロセスは、OpenShift Data Foundation デプロイメントで使用されるインフラストラクチャーによって異なります。

警告

設定する次の OpenShift サービス用に十分なストレージ容量があることを常に確認してください。

- OpenShift イメージレジストリー
- OpenShift モニタリング
- OpenShift ロギング (Loki)
- OpenShift トレースプラットフォーム (Tempo)

これらの重要なサービス用のストレージのスペースが不足すると、OpenShift クラスターは動作不能になり、回復が非常に困難になります。

Red Hat は、これらのサービスのキュレーションおよび保持期間を短く設定することを推奨します。詳細は、OpenShift Container Platform ドキュメントの **Monitoring** の Curator スケジュールの設定 と Prometheus メトリックデータの保持期間の編集 を参照してください。

これらのサービスのストレージ領域が不足する場合は、Red Hat カスタマーサポートにお問い合わせください。

4.1. OPENSHIFT DATA FOUNDATION を使用するためのイメージレジストリーの設定

OpenShift Container Platform は、クラスターで標準ワークロードとして実行される、組み込まれたコンテナーイメージレジストリーを提供します。通常、レジストリーはクラスター上にビルドされたイメージの公開ターゲットとして、またクラスター上で実行されるワークロードのイメージのソースとして使用されます。

このセクションの手順に従って、OpenShift Data Foundation をコンテナーイメージレジストリーのストレージとして設定します。AWS では、レジストリーのストレージを変更する必要はありません。ただし vSphere およびベアメタルプラットフォームの場合は、OpenShift Data Foundation 永続ボリュー

ムに対してストレージを変更することが推奨されます。

警告

このプロセスでは、データを既存イメージレジストリーから新規イメージレジストリーに移行しません。既存のレジストリーにコンテナーイメージがある場合は、このプロセスを完了する前にレジストリーのバックアップを作成し、このプロセスの完了時にイメージを再登録します。

前提条件

- OpenShift Web コンソールへの管理者アクセス。
- OpenShift Data Foundation Operator が **openshift-storage** namespace にインストールされ、実行されている。OpenShift Web Console で、**Operators → Installed Operators** をクリックしてインストールされた Operator を表示します。
- イメージレジストリー Operator が **openshift-image-registry** namespace にインストールされ、実行されている。OpenShift Web コンソールで、**Administration → Cluster Settings → Cluster Operators** をクリックしてクラスター Operator を表示します。
- プロビジョナー **openshift-storage.cephfs.csi.ceph.com** を持つストレージクラスが利用可能である。OpenShift Web コンソールで、**Storage → StorageClasses**をクリックし、利用可能なストレージクラスを表示します。

手順

1. **使用するイメージレジストリーの永続ボリューム要求を作成します。**
   a. OpenShift Web コンソールで、**Storage → Persistent Volume Claims**をクリックします。
   b. **Project** を **openshift-image-registry** に設定します。
   c. **Create Persistent Volume Claim**をクリックします。
      i. 上記で取得した利用可能なストレージクラスリストから、プロビジョナー **openshift-storage.cephfs.csi.ceph.com** で Storage Class を指定します。
      ii. 永続ボリューム要求の **Name** を指定します (例: **ocs4registry**)。
      iii. **Shared Access (RWX)** の Access Mode を指定します。
      iv. 100 GB 以上の **Size** を指定します。
      v. **Create** をクリックします。
      新規永続ボリューム要求のステータスが **Bound** としてリスト表示されるまで待機します。
2. **クラスターのイメージレジストリーを、新規の永続ボリューム要求を使用するように設定します。**
   a. **Administration → Custom Resource Definitions**をクリックします。

b. **imageregistry.operator.openshift.io** グループに関連付けられた **Config** カスタムリソース定義をクリックします。

c. **Instances** タブをクリックします。

d. クラスターインスタンスの横にある **Action** メニュー (⋮) → **Edit Config** をクリックします。

e. イメージレジストリーの新規永続ボリューム要求を追加します。

i. 以下を **spec:** の下に追加し、必要に応じて既存の **storage:** セクションを置き換えます。

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

以下に例を示します。

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

ii. **Save** をクリックします。

3. **新しい設定が使用されていることを確認します。**

a. **Workloads** → **Pods** をクリックします。

b. **Project** を **openshift-image-registry** に設定します。

c. 新規 **image-registry-*** Pod が **Running** のステータスと共に表示され、以前の **image-registry-*** Pod が終了していることを確認します。

d. 新規の **image-registry-*** Pod をクリックし、Pod の詳細を表示します。

e. **Volumes** までスクロールダウンし、**registry-storage** ボリュームに新規永続ボリューム要求に一致する **Type** があることを確認します (例: **ocs4registry**)。

4.2. MULTICLOUD OBJECT GATEWAY を OPENSHIFT IMAGE REGISTRY バックエンドストレージとして使用する

オンプレミスの OpenShift デプロイメントでは、Multicloud Object Gateway (MCG) を OpenShift Container Platform (OCP) Image Registry バックエンドストレージとして使用できます。

MCG を OCP イメージレジストリーのバックエンドストレージとして設定するには、手順に記載されている手順に従います。

前提条件

- OCP Web コンソールへの管理アクセス。
- MCG を使用して実行中の OpenShift Data Foundation クラスター。

手順

1. オブジェクトバケット要求の作成 の手順に従って、**ObjectBucketClaim** を作成します。
2. **image-registry-private-configuration-user** シークレットを作成します。
   a. OpenShift web-console に移動します。
   b. **ObjectBucketClaim → ObjectBucketClaim Data** をクリックします。
   c. **ObjectBucketClaim data** の **openshift-image-registry namespace** で **MCG access key** と **MCG secret key** を探します。
   d. 次のコマンドを使用してシークレットを作成します。

```
$ 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. Image Registry Operator の **managementState** のステータスを **Managed** に変更します。

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

4. Image Registry Operator 設定ファイルの **spec.storage** セクションを編集します。
   a. Web コンソールから **Object Bucket Claim Data** セクションの **unique-bucket-name** と **regionEndpoint** を取得します。**または**、次のコマンドから regionEndpoint と unique-bucket-name の情報を取得することもできます。

```
$ oc describe  noobaa
```

   b. 以下の場合は、**regionEndpoint** を http://<Endpoint-name>:<port> として追加します。
      - storageclass が **ceph-rgw** storageclass、および
      - エンドポイントが、openshift-storage namespace からの内部 SVC を指している。
   c. Operator レジストリー設定ファイルに変更を加えた後、**image-registry** Pod が生成されます。

```
$ 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. イメージレジストリー設定をデフォルトにリセットします。

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

検証手順

- 次のコマンドを実行して、MCG が OpenShift Image Registry バックエンドストレージとして正常に設定されているかどうかを確認します。

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

出力例

```
$ 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
```

- (オプション) 次のコマンドを実行して、MCG が OpenShift Image Registry バックエンドストレージとして正常に設定されているかどうかを確認することもできます。

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

出力例

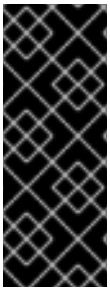
```
$ oc describe pod image-registry-b6c8f4596-ln88h

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. OPENSHIFT DATA FOUNDATION を使用するためのモニタリングの設定

OpenShift Data Foundation は、Prometheus と Alert Manager で構成されるモニタリングスタックを提供します。

このセクションの手順に従って、OpenShift Data Foundation をモニタリングスタックのストレージとして設定します。



重要

ストレージ領域が不足すると、モニタリングは機能しません。モニタリング用に十分なストレージ容量があることを常に確認します。

Red Hat は、このサービスの保持期間を短く設定することを推奨します。詳細は、OpenShift Container Platform ドキュメントのモニタリングガイドの Prometheus メトリクスデータの保持期間の変更 を参照してください。

前提条件

- OpenShift Web コンソールへの管理者アクセス。
- OpenShift Data Foundation Operator が **openshift-storage** namespace にインストールされ、実行されている。OpenShift Web コンソールで、**Operators → Installed Operators** をクリックし、インストールされた Operator を表示します。
- モニタリング Operator が **openshift-monitoring** namespace にインストールされ、実行されている。OpenShift Web コンソールで、**Administration → Cluster Settings → Cluster Operators** をクリックし、クラスター Operator を表示します。
- プロビジョナー **openshift-storage.rbd.csi.ceph.com** を持つストレージクラスが利用可能である。OpenShift Web コンソールで、**Storage → StorageClasses**をクリックし、利用可能なストレージクラスを表示します。

手順

1. OpenShift Web コンソールで、**Workloads → Config Maps** に移動します。
2. **Project** ドロップダウンを **openshift-monitoring** に設定します。
3. **Create Config Map** をクリックします。
4. 以下の例を使用して新規の **cluster-monitoring-config** config map を定義します。
   山括弧 (**<**、**>**) 内の内容を独自の値に置き換えます (例: **retention: 24h** または **storage: 40Gi**)。

   storageClassName、をプロビジョナー **openshift-storage.rbd.csi.ceph.com** を使用する **storageclass** に置き換えます。以下の例では、**storageclass** の名前は **ocs-storagecluster-ceph-rbd** です。

   cluster-monitoring-config config map の例

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-monitoring-config
  namespace: openshift-monitoring
data:
  config.yaml: |
    prometheusK8s:
      retention: <time to retain monitoring files, for example 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
```

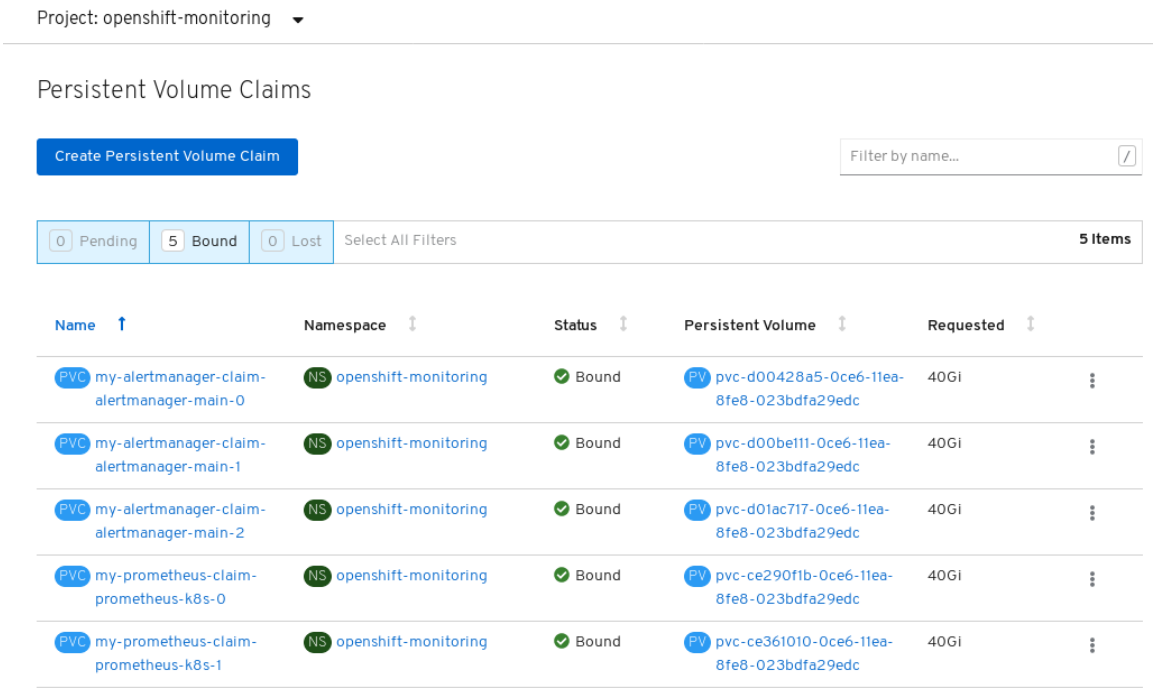
```
        resources:
          requests:
            storage: <size of claim, e.g. 40Gi>
```

5. **Create** をクリックして、config map を保存し、作成します。

検証手順

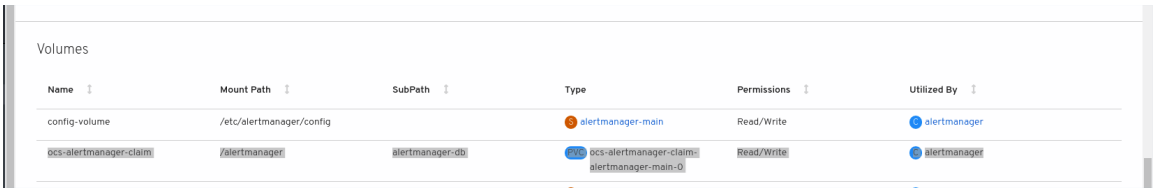
1. 永続ボリューム要求が Pod にバインドされていることを確認します。
   a. **Storage → Persistent Volume Claims** に移動します。
   b. **Project** ドロップダウンを **openshift-monitoring** に設定します。
   c. 5 つの永続ボリューム要求が **Bound** (バインド) の状態で表示され、3 つの **alertmanager-main-*** Pod および 2 つの **prometheus-k8s-*** Pod に割り当てられていることを確認します。

図4.1 作成済みのバインドされているストレージのモニタリング



2. 新規の **alertmanager-main-*** Pod が **Running** 状態で表示されることを確認します。
   a. **Workloads → Pods** に移動します。
   b. 新規の **alertmanager-main-*** Pod をクリックし、Pod の詳細を表示します。
   c. **Volumes** にスクロールダウンし、ボリュームに新規永続ボリューム要求のいずれかに一致する **Type ocs-alertmanager-claim** があることを確認します (例: **ocs-alertmanager-claim-alertmanager-main-0**)。

図4.2 alertmanager-main-* Pod に割り当てられた永続ボリューム要求

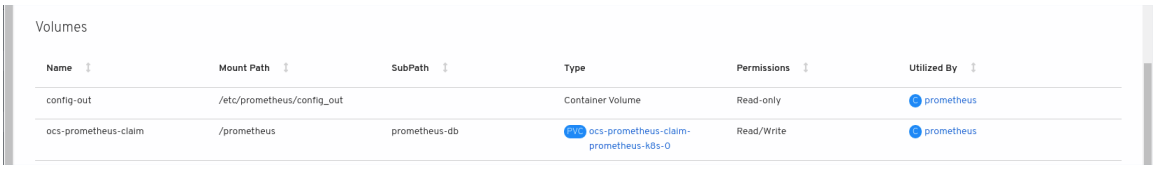


3. 新規 **prometheus-k8s-*** Pod が **Running** 状態で表示されることを確認します。

   a. 新規 **prometheus-k8s-*** Pod をクリックし、Pod の詳細を表示します。

   b. **Volumes** までスクロールダウンし、ボリュームに新規の永続ボリューム要求のいずれかに一致する **Type ocs-prometheus-claim** があることを確認します (例: **ocs-prometheus-claim-prometheus-k8s-0**)。

図4.3 prometheus-k8s-* Pod に割り当てられた永続ボリューム要求



4.4. オーバープロビジョニングレベルのポリシー制御

オーバープロビジョニング制御は、特定のアプリケーション namespace に基づいて、ストレージクラスターから消費される永続ボリューム要求の量にクォータを定義できるようにするメカニズムです。

オーバープロビジョニング制御メカニズムを有効にすると、ストレージクラスターから消費される PVC をオーバープロビジョニングするのを防ぐことができます。OpenShift は、**ClusterResourceQuota** を利用して、集約されたリソース消費をクラスタースコープで制限する制約を定義する柔軟性を備えています。詳細は、OpenShift ClusterResourceQuota を参照してください。

オーバープロビジョニング制御を使用すると、**ClusteResourceQuota** が開始し、各ストレージクラスのストレージ容量制限を設定できます。

OpenShift Data Foundation のデプロイメントの詳細は、製品ドキュメント を参照し、プラットフォームに応じたデプロイメント手順を選択してください。

前提条件

- OpenShift Data Foundation クラスターが作成されている。

手順

1. コマンドラインインターフェイスまたはユーザーインターフェイスのいずれかから **storagecluster** をデプロイします。

2. アプリケーションの namespace にラベルを付けます。

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

<desired_name>

アプリケーション namespace の名前を指定します (例: **quota-rbd**)。

<desired_label>

ストレージクォータのラベルを指定します (例: **storagequota1**)。

3. **storagecluster** を編集して、ストレージクラスのクォータ制限を設定します。

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

<ocs_storagecluster_name>

ストレージクラスターの名前を指定します。

4. 必要なハード制限を持つオーバープロビジョニング制御のエントリーを **StorageCluster.Spec** に追加します。

```
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>

ストレージクラスに必要なクォータ制限を指定します (例: **27Ti**)。

<storage_class_name>

クォータ制限を設定するストレージクラスの名前を指定します (例: **ocs-storagecluster-ceph-rbd**)。

<desired_quota_name>

ストレージクォータの名前を指定します (例: **quota1**)。

<desired_label>

ストレージクォータのラベルを指定します (例: **storagequota1**)。

5. 変更された **storagecluster** を保存します。

6. **clusterresourcequota** が定義されていることを確認します。

注記

前の手順で定義した **quotaName** (例: **quota1**) を持つ **clusterresourcequota** を想定します。

```
$ oc get clusterresourcequota -A

$ oc describe clusterresourcequota -A
```

-

4.5. OPENSHIFT DATA FOUNDATION のクラスターロギング

クラスターロギングをデプロイして、各種の OpenShift Container Platform サービスに関するログを集計できます。クラスターロギングのデプロイ方法の詳細は、クラスターロギングのデプロイ を参照してください。

OpenShift Container Platform の初回のデプロイメントでは、OpenShift Data Foundation はデフォルトで設定されず、OpenShift Container Platform クラスターはノードから利用可能なデフォルトストレージのみに依存します。OpenShift ロギング (ElasticSearch) のデフォルト設定を OpenShift Data Foundation で対応されるように編集し、OpenShift Data Foundation でサポートされるロギング (Elasticsearch) を設定できます。

重要

これらのサービスに十分なストレージ容量があることを常に確認してください。これらの重要なサービスのストレージ領域が不足すると、ロギングアプリケーションは動作しなくなり、復元が非常に困難になります。

Red Hat は、これらのサービスのキュレーションおよび保持期間を短く設定することを推奨します。詳細は、OpenShift Container Platform ドキュメントの クラスターロギングキュレーター を参照してください。

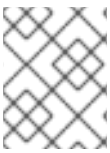
これらのサービスのストレージ領域が不足している場合は、Red Hat カスタマーポータルにお問い合わせください。

4.5.1. 永続ストレージの設定

ストレージクラス名およびサイズパラメーターを使用して、Elasticsearch クラスターの永続ストレージクラスおよびサイズを設定できます。Cluster Logging Operator は、これらのパラメーターに基づいて、Elasticsearch クラスターの各データノードに永続ボリューム要求を作成します。以下に例を示します。

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

この例では、クラスター内の各データノードが **200GiB** の **ocs-storagecluster-ceph-rbd** ストレージを要求する永続ボリューム要求にバインドされるように指定します。それぞれのプライマリーシャードは単一のレプリカによってサポートされます。シャードのコピーはすべてのノードにレプリケートされ、常に利用可能となり、冗長性ポリシーにより 2 つ以上のノードが存在する場合にコピーを復元できます。Elasticsearch レプリケーションポリシーの詳細は、クラスターロギングのデプロイおよび設定について の **Elasticsearch レプリケーションポリシー** を参照してください。



注記

ストレージブロックを省略すると、デプロイメントはデフォルトのストレージでサポートされます。以下に例を示します。

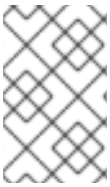
■

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

詳細は、クラスターロギングの設定 を参照してください。

4.5.2. OpenShift Data Foundation を使用するためのクラスターロギングの設定

このセクションの手順に従って、OpenShift Data Foundation を OpenShift クラスターロギングのストレージとして設定します。



注記

OpenShift Data Foundation では、ロギングを初めて設定する際に、すべてのログを取得できます。ただし、ロギングをアンインストールして再インストールすると、古いログが削除され、新しいログのみが処理されます。

前提条件

- OpenShift Web コンソールへの管理者アクセス。
- OpenShift Data Foundation Operator が **openshift-storage** namespace にインストールされ、実行されている。
- Cluster Logging Operator が **openshift-logging** namespace にインストールされ、実行されている。

手順

1. OpenShift Web コンソールの左側のペインから **Administration → Custom Resource Definitions** をクリックします。
2. Custom Resource Definitions ページで、**ClusterLogging** をクリックします。
3. Custom Resource Definition Overview ページで、Actions メニューから **View Instances** を選択するか、**Instances** タブをクリックします。
4. Cluster Logging ページで、**Create Cluster Logging** をクリックします。
データを読み込むためにページの更新が必要になる場合があります。
5. YAML において、**storageClassName**、をプロビジョナー **openshift-storage.rbd.csi.ceph.com** を使用する **storageclass** に置き換えます。以下の例では、**storageclass** の名前は **ocs-storagecluster-ceph-rbd** です。

```
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: {}
```

OpenShift Data Foundation ノードにテイントのマークが付けられている場合は、ロギング用に daemonset Pod のスケジューリングを有効にするために容認を追加する必要があります。

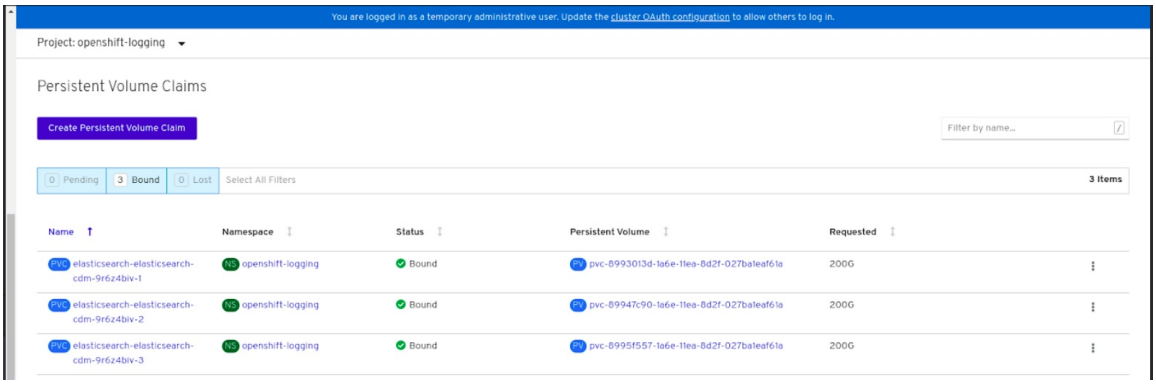
```
spec:
[...]
  collection:
    logs:
      fluentd:
        tolerations:
        - effect: NoSchedule
          key: node.ocs.openshift.io/storage
          value: 'true'
      type: fluentd
```

6. **Save** をクリックします。

検証手順

1. 永続ボリューム要求が **elasticsearch** Pod にバインドされていることを確認します。
   a. **Storage → Persistent Volume Claims** に移動します。
   b. **Project** ドロップダウンを **openshift-logging** に設定します。
   c. 永続ボリューム要求が **elasticsearch-*** Pod に割り当てられ、**Bound** (バインド) の状態で表示されることを確認します。

図4.4 作成済みのバインドされたクラスターロギング



2. 新規クラスターロギングが使用されていることを確認します。

   a. **Workload → Pods** をクリックします。

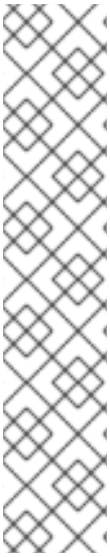
   b. プロジェクトを **openshift-logging** に設定します。

   c. 新規の **elasticsearch-*** Pod が **Running** 状態で表示されることを確認します。

   d. 新規の **elasticsearch-*** Pod をクリックし、Pod の詳細を表示します。

   e. **Volumes** までスクロールダウンし、elasticsearch ボリュームに新規永続ボリューム要求に一致する **Type** があることを確認します (例: **elasticsearch-elasticsearch-cdm-9r624biv-3**)。

   f. 永続ボリューム要求の名前をクリックし、PersistentVolumeClaim Overview ページでストレージクラス名を確認します。



注記

Elasticsearch Pod に割り当てられる PV の詳細シナリオを回避するために、キュレーターの時間を短く設定して使用するようにしてください。

Curator を、保持設定に基づいて Elasticsearch データを削除するように設定できます。以下の 5 日間のインデックスデータの保持期間をデフォルトとして設定することが推奨されます。

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

詳細は、Elasticsearch データのキュレーション を参照してください。

注記

永続ボリューム要求がサポートするクラスターロギングをアンインストールするには、それぞれのデプロイメントガイドのアンインストールに関する章に記載されている、クラスターロギング Operator の OpenShift Data Foundation からの削除に関する手順を使用します。

第5章 MULTUS ネットワークの作成

OpenShift Container Platform は、Multus CNI プラグインを使用して CNI プラグインのチェーンを許可します。クラスターのインストール中にデフォルトの Pod ネットワークを設定できます。デフォルトのネットワークは、クラスターの通常のネットワークトラフィックをすべて処理します。

利用可能な CNI プラグインに基づいて追加のネットワークを定義し、1 つまたは複数のネットワークを Pod に割り当てることができます。追加のネットワークを Pod に割り当てるには、インターフェイスの割り当て方法を定義する設定を作成する必要があります。

NetworkAttachmentDefinition (NAD) カスタムリソース (CR) を使用して、各インターフェイスを指定します。それぞれの NetworkAttachmentDefinition 内の CNI 設定は、インターフェイスの作成方法を定義します。

OpenShift Data Foundation は、macvlan と呼ばれる CNI プラグインを使用します。macvlan ベースの追加ネットワークを作成することで、ホスト上の Pod が物理ネットワークインターフェイスを使用して他のホストやそれらのホストの Pod と通信できます。macvlan ベースの追加ネットワークに割り当てられる各 Pod には固有の MAC アドレスが割り当てられます。

5.1. ネットワーク接続定義の作成

Multus を利用するには、正しいネットワーク設定を備えた動作中のクラスターが必要です。Multus 設定の要件 を参照してください。新規に作成された NetworkAttachmentDefinition (NAD) は、Storage Cluster のインストール時に選択できます。これは、Storage Cluster の前に作成する必要のある理由です。

注記

ネットワーク割り当ての定義では、**whereabouts** IP アドレス管理 (IPAM) のみを使用でき、**range** フィールドを指定する必要があります。**ipRanges** およびプラグインチェーンはサポートされていません。

Storage Cluster のインストール中に、新しく作成された **NetworkAttachmentDefinition** (NAD) を選択できます。これが、Storage Cluster を作成する前に NAD を作成する必要がある理由です。

プランニングガイドで説明されているように、作成する Multus ネットワークは、OpenShift Data Foundation トラフィックで利用可能なネットワークインターフェイスの数によって異なります。すべてのストレージトラフィックを 2 つのインターフェイス (デフォルトの OpenShift SDN に使用されるインターフェイス 1 つ) に分割するか、ストレージストレージトラフィック (パブリック) およびストレージレプリケーショントラフィック (プライベートまたはクラスター) にさらに分割することもできます。

以下は、同じインターフェイス上のすべてのストレージトラフィック (パブリックおよびクラスター) の **NetworkAttachmentDefinition** の例です。すべてのスケジュール可能なノードに 1 つの追加インターフェイスが必要です (別のネットワークインターフェイス上の OpenShift デフォルト SDN)。

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
 name: ceph-multus-net
 namespace: openshift-storage
spec:
 config: '{
     "cniVersion": "0.3.1",
     "type": "macvlan",
```

```
    "master": "eth0",
    "mode": "bridge",
    "ipam": {
      "type": "whereabouts",
      "range": "192.168.200.0/24",
      "routes": [
        {"dst": "NODE_IP_CIDR"}
      ]
    }
  }'
```

注記

すべてのネットワークインターフェイス名は、Multus ネットワークに接続されているすべてのノードで同じである必要があります (例: **ocs-public-cluster** の場合は **ens2**)。

以下は、個別の Multus ネットワーク上のストレージトラフィックの **NetworkAttachmentDefinition** の例になります。これは、クライアントストレージトラフィックのパブリックおよびレプリケーショントラフィック用のクラスターです。Object Storage Device (OSD) Pod をホストする OpenShift ノードに 2 つの追加インターフェイスと、他のスケジュール可能なすべてのノードで 1 つの追加インターフェイス (OpenShift デフォルト SDN) が必要です。

```
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 の例:

```
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"
    }
  }'
```

注記

すべてのネットワークインターフェイス名は、Multus ネットワークにアタッチされたすべてのノードで同じである必要があります (つまり、**ocs-public** の場合は **ens2**、**ocs-cluster**の場合は **ens3** です)。

第6章 OPENSHIFT DATA FOUNDATION を使用した OPENSHIFT CONTAINER PLATFORM アプリケーションのサポート

OpenShift Container Platform のインストール時に OpenShift Data Foundation を直接インストールすることはできません。ただし、Operator Hub を使用して OpenShift Data Foundation を既存の OpenShift Container Platform にインストールし、OpenShift Container Platform アプリケーションを OpenShift Data Foundation でサポートされるように設定することができます。

前提条件

- OpenShift Container Platform がインストールされ、OpenShift Web コンソールへの管理者アクセスがある。
- OpenShift Data Foundation が **openshift-storage** namespace にインストールされ、実行されている。

手順

1. OpenShift Web コンソールで、以下のいずれかを実行します。
   - **Workloads → Deployments** をクリックします。
     Deployments ページで、以下のいずれかを実行できます。
     - 既存のデプロイメントを選択し、**Action** メニュー (⋮) から **Add Storage** オプションをクリックします。
     - 新規デプロイメントを作成してからストレージを追加します。
       i. **Create Deployment** をクリックして新規デプロイメントを作成します。
       ii. 要件に応じて **YAML** を編集し、デプロイメントを作成します。
       iii. **Create** をクリックします。
       iv. ページ右上の **Actions** ドロップダウンメニューから **Add Storage** を選択します。
   - **Workloads → Deployment Configs**をクリックします。
     Deployment Configs ページで、以下のいずれかを実行できます。
     - 既存のデプロイメントを選択し、**Action** メニュー (⋮) から **Add Storage** オプションをクリックします。
     - 新規デプロイメントを作成してからストレージを追加します。
       i. **Create Deployment Config** をクリックし、新規デプロイメントを作成します。
       ii. 要件に応じて **YAML** を編集し、デプロイメントを作成します。
       iii. **Create** をクリックします。
       iv. ページ右上の **Actions** ドロップダウンメニューから **Add Storage** を選択します。
2. Add Storage ページで、以下のオプションのいずれかを選択できます。
   - **Use existing claim** オプションをクリックし、ドロップダウンリストから適切な PVC を選

択します。

- **Create new claim** オプションをクリックします。
  a. **Storage Class** ドロップダウンリストから適切な **CephFS** または **RBD** ストレージクラスを選択します。
  b. の名前を指定します。
  c. アクセスモード ReadWriteOnce (RWO) または ReadWriteMany (RWX) を選択します。

注記

ReadOnlyMany (ROX) はサポートされないため、非アクティブになります。

  d. 必要なストレージ容量のサイズを選択します。

注記

ブロック PV を拡張することはできますが、永続ボリューム要求の作成後にストレージ容量のサイズを縮小することはできません。

3. コンテナー内のマウントパスボリュームのマウントパスとサブパス (必要な場合) を指定します。
4. **Save** をクリックします。

検証手順

1. 設定に応じて、以下のいずれかを実行します。
   - **Workloads → Deployments** をクリックします。
   - **Workloads → Deployment Configs**をクリックします。
2. 必要に応じてプロジェクトを設定します。
3. ストレージを追加したデプロイメントをクリックして、デプロイメントの詳細を表示します。
4. **Volumes** までスクロールダウンし、デプロイメントに、割り当てた永続ボリューム要求に一致する **Type** があることを確認します。
5. 永続ボリューム要求の名前をクリックし、Persistent Volume Claim Overview ページでストレージクラス名を確認します。

第7章 既存の外部の OPENSHIFT DATA FOUNDATION クラスターへのファイルおよびオブジェクトストレージの追加

OpenShift Data Foundation が外部モードで設定されている場合に、永続ボリューム要求および Object Bucket Claim (オブジェクトバケット要求) 向けにストレージを提供する方法は複数あります。

- ブロックストレージ用の永続ボリューム要求は、外部の Red Hat Ceph Storage クラスターから直接提供されます。
- ファイルストレージ用の永続ボリューム要求は、メタデータサーバー (MDS) を外部の Red Hat Ceph Storage クラスターに追加して提供できます。
- オブジェクトストレージのオブジェクトバケット要求は、Multicloud Object Gateway を使用するか、Ceph Object Gateway を外部の Red Hat Ceph Storage クラスターに追加して提供できます。

ブロックストレージのみを提供するために最初にデプロイした外部 OpenShift Data Foundation クラスターに、ファイルストレージ (メタデータサーバーを使用) またはオブジェクトストレージ (Ceph Object Gateway を使用)、あるいはその両方を追加するには、次のプロセスを使用します。

前提条件

- OpenShift Data Foundation 4.15 が OpenShift Container Platform バージョン 4.16 以降にインストールされ、実行されている。また、外部モードの OpenShift Data Foundation Cluster が **Ready** 状態にある。
- 外部の Red Hat Ceph Storage クラスターが以下のいずれかまたは両方で設定されている。
  - オブジェクトストレージ用に OpenShift Container Platform クラスターがアクセスできる Ceph Object Gateway (RGW) エンドポイント
  - ファイルストレージ用のメタデータサーバー (MDS) プール
- 外部の OpenShift Data Foundation クラスターのデプロイメント時に **ceph-external-cluster-details-exporter.py** スクリプトで使用されるパラメーターを把握している。

手順

1. 以下のコマンドを使用して Python スクリプト **ceph-external-cluster-details-exporter.py** の OpenShift Data Foundation バージョンをダウンロードします。

```
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. 更新パーミッションは、外部の Red Hat Ceph Storage クラスターのクライアントノードで **ceph-external-cluster-details-exporter.py** を実行して、外部の Red Hat Ceph Storage クラスターを制限します。これを行うには、Red Hat Ceph Storage の管理者への問い合わせが必要になる場合があります。

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

--run-as-user

OpenShift Data Foundation クラスターのデプロイメント時に使用されるクライアント名。別のクライアント名が設定されていない場合は、デフォルトのクライアント名 **client.healthchecker** を使用します。

--rgw-pool-prefix

Ceph Object Gateway プールに使用する接頭辞。デフォルトの接頭辞を使用している場合は、省略できます。

3. 外部の Red Hat Ceph Storage クラスターから設定詳細を生成して保存します。

   a. 外部の Red Hat Ceph Storage クラスターのクライアントノードで **ceph-external-cluster-details-exporter.py** を実行して、設定の詳細を生成します。

```
# 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

オプション。OpenShift Container Platform クラスターから到達可能な、アクティブ mgr およびスタンバイ mgr の IP アドレスのコンマ区切りリストを受け入れます。指定しないと、値が自動的に入力されます。

--monitoring-endpoint-port

オプション。**--monitoring-endpoint** で指定された ceph-mgr Prometheus エクスポーターに関連付けられるポートです。指定しないと、値が自動的に入力されます。

--run-as-user

OpenShift Data Foundation クラスターのデプロイメント時に使用されるクライアント名。別のクライアント名が設定されていない場合は、デフォルトのクライアント名 client.healthchecker を使用します。

--rgw-endpoint

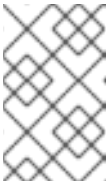
このパラメーターを指定して OpenShift Data Foundation の Ceph Object Gateway でオブジェクトストレージをプロビジョニングします (任意のパラメーター)。

--rgw-pool-prefix

Ceph Object Gateway プールに使用する接頭辞。デフォルトの接頭辞を使用している場合は、省略できます。

ユーザーパーミッションは、以下のように更新されます。

```
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
```



注記

Ceph Object Gateway の詳細 (指定されている場合) 以外の全パラメーター (任意の引数を含む) は、OpenShift Data Foundation を外部モードでデプロイした時に使用したものと同じです。

b. スクリプトの出力を **external-cluster-config.json** ファイルに保存します。
以下の出力例では、生成された設定変更を太字で示しています。


```
[{"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. 生成された JSON ファイルをアップロードします。

   a. OpenShift Web コンソールにログインします。

   b. **Workloads → Secrets** をクリックします。

   c. **プロジェクト** を **openshift-storage** に設定します。

   d. **rook-ceph-external-cluster-details** をクリックします。

   e. **Actions** (⋮) → **Edit Secret** をクリックします。

   f. **Browse** をクリックして **external-cluster-config.json** ファイルをアップロードします。

   g. **Save** をクリックします。

検証手順

- OpenShift Data Foundation クラスターが正常であり、データが回復性があることを確認するには、**Storage → Data foundation → Storage Systems** タブに移動してから、ストレージシステム名をクリックします。
  - **Overview → Block and File** タブで Status カードをチェックして、**Storage Cluster** に正常であることを示す緑色のチェックマークが表示されていることを確認します。
- ファイルストレージ用のメタデータサーバーを追加した場合:
  a. **Workloads → Pods** をクリックして、**csi-cephfsplugin-*** Pod が新規作成され、状態が **Running** であることを確認します。
  b. **Storage → Storage Classes** をクリックして **ocs-external-storagecluster-cephfs** ストレージクラスが作成されていることを確認します。
- オブジェクトストレージ用に Ceph Object Gateway を追加した場合:

a. **Storage → Storage Classes** をクリックして **ocs-external-storagecluster-ceph-rgw** ストレージクラスが作成されていることを確認します。

b. OpenShift Data Foundation クラスターが正常であり、データが回復性があることを確認するには、**Storage → Data foundation → Storage Systems** タブに移動してから、ストレージシステム名をクリックします。

c. **Object** タブをクリックして、**Object Service** および **Data resiliency** に正常であることを示す緑色のチェックマークが表示されていることを確認します。

第8章 RED HAT OPENSHIFT DATA FOUNDATION に専用のワーカーノードを使用する方法

Red Hat OpenShift Container Platform サブスクリプションには、OpenShift Data Foundation サブスクリプションが必要です。ただし、インフラストラクチャーノードを使用して OpenShift Data Foundation リソースをスケジュールしている場合は、OpenShift Container Platform のサブスクリプションコストを節約できます。

マシン API サポートの有無にかかわらず複数の環境全体で一貫性を維持することが重要です。そのため、いずれの場合でも、worker または infra のいずれかのラベルが付けられたノードの特別なカテゴリーや、両方のロールを使用できるようにすることが強く推奨されます。詳細は、「インフラストラクチャーノードの手動作成」 セクションを参照してください。

8.1. インフラストラクチャーノードの仕組み

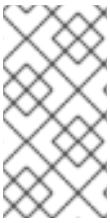
OpenShift Data Foundation で使用するインフラストラクチャーノードにはいくつかの属性があります。ノードが RHOCP エンタイトルメントを使用しないようにするには、**infra** ノードロールのラベルが必要です。**infra** ノードロールラベルは、OpenShift Data Foundation を実行するノードには OpenShift Data Foundation エンタイトルメントのみが必要となるようにします。

- **node-role.kubernetes.io/infra** のラベル

infra ノードが OpenShift Data Foundation リソースのみをスケジュールできるようにするには、**NoSchedule** 効果のある OpenShift Data Foundation テイントを追加する必要もあります。

- **node.ocs.openshift.io/storage="true"** のテイント

RHOCP サブスクリプションコストが適用されないように、ラベルは RHOCP ノードを **infra** ノードとして識別します。テイントは、OpenShift Data Foundation 以外のリソースがテイントのマークが付けられたノードでスケジュールされないようにします。



注記

ノードにストレージテイントを追加するには、**openshift-dns daemonset** などの他の **daemonset** Pod の容認処理が必要になる場合があります。容認を管理する方法については、ナレッジベースの記事 Openshift-dns daemonsets doesn't include toleration to run on nodes with taints を参照してください。

OpenShift Data Foundation サービスの実行に使用されるインフラストラクチャーノードで必要なテイントおよびラベルの例:

```
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. インフラストラクチャーノードを作成するためのマシンセット

マシン API が環境でサポートされている場合は、インフラストラクチャーノードのプロビジョニングを行うマシンセットのテンプレートにラベルを追加する必要があります。マシン API が作成するノードに手動でラベルを追加するアンチパターンを回避します。これを実行することは、デプロイメントで作成される Pod にラベルを追加することに似ています。いずれの場合も、Pod/ノードに障害が発生すると、交換された Pod/ノードには適切なラベルがありません。

注記

EC2 環境では、3 つのマシンセットが必要です。それぞれは、異なるアベイラビリティーゾーン (us-east-2a、us-east-2b、us-east-2c など) でインフラストラクチャーノードをプロビジョニングするように設定されます。現時点で、OpenShift Data Foundation は 4 つ以上のアベイラビリティーゾーンへのデプロイをサポートしていません。

以下の Machine Set テンプレートのサンプルは、インフラストラクチャーノードに必要な適切なテイントおよびラベルを持つノードを作成します。これは OpenShift Data Foundation サービスを実行するために使用されます。

```
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

インフラストラクチャーノードにテイントを追加する場合は、fluentd Pod など、他のワークロードのテイントにも容認を追加する必要があります。詳細は、Red Hat ナレッジベースのソリューション記事 OpenShift 4 のインフラストラクチャーノード を参照してください。

8.3. インフラストラクチャーノードの手動作成

マシン API が環境内でサポートされない場合にのみ、ラベルはノードに直接適用される必要があります。手動作成では、OpenShift Data Foundation サービスをスケジュールするために少なくとも 3 つの RHOCP ワーカーノードが利用可能であり、これらのノードに CPU およびメモリーリソースが十分にある必要があります。RHOCP サブスクリプションコストの発生を防ぐには、以下が必要です。

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

また、**NoSchedule** OpenShift Data Foundation テイントを追加することも、**infra** ノードが OpenShift Data Foundation リソースのみをスケジュールし、その他の OpenShift Data Foundation ワークロードを拒否できるようにするために必要です。

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

警告

ノードロール node-role.kubernetes.io/worker="" は削除しないでください。

node-role.kubernetes.io/worker="" ノードロールを削除すると、OpenShift スケジューラーおよび MachineConfig リソースの両方に変更が加えられない場合に問題が発生する可能性があります。

すでに削除されている場合は、各 **infra** ノードに再度追加する必要があります。**node-role.kubernetes.io/infra=""** ノードロールおよび OpenShift Data Foundation テイントを追加するだけで、エンタイトルメント免除要件を満たすことができます。

8.4. ユーザーインターフェイスからノードのテイント

このセクションでは、OpenShift Data Foundation のデプロイ後にノードをテイントする手順を説明します。

手順

1. OpenShift Web Console で、**Compute → Nodes** をクリックし、テイントする必要のあるノードを選択します。
2. **Details** ページで、**Edit taints** をクリックします。
3. **Key** <nodes.openshift.ocs.io/storage>、**Value** <true>、および **Effect**<Noschedule> フィールドに値を入力します。
4. Save をクリックします。

検証手順

- 次の手順に従って、ノードが正常にテイントされたことを確認します。
  - **Compute → Nodes** に移動します。
  - ノードを選択してステータスを確認し、**YAML** タブをクリックします。
  - **specs** セクションで、次のパラメーターの値を確認します。

    ```
    Taints:
    ```

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

関連情報

詳細は、VMware vSphere での OpenShift Data Foundation クラスターの作成 を参照してください。

第9章 永続ボリューム要求の管理

9.1. OPENSHIFT DATA FOUNDATION を使用するためのアプリケーション POD の設定

このセクションの手順に従って、OpenShift Data Foundation をアプリケーション Pod のストレージとして設定します。

前提条件

- OpenShift Web コンソールへの管理者アクセス。
- OpenShift Data Foundation Operator が **openshift-storage** namespace にインストールされ、実行されている。OpenShift Web Console で、**Operators → Installed Operators** をクリックしてインストールされた Operator を表示します。
- OpenShift Data Foundation が提供するデフォルトのストレージクラスが利用可能である。OpenShift Web コンソールで **Storage → StorageClasses** をクリックし、デフォルトのストレージクラスを表示します。

手順

1. **使用するアプリケーションの永続ボリューム要求を作成します。**
   a. OpenShift Web コンソールで、**Storage → Persistent Volume Claims**をクリックします。
   b. アプリケーション Pod の **Project** を設定します。
   c. **Create Persistent Volume Claim**をクリックします。
      i. OpenShift Data Foundation によって提供される **Storage Class** を指定します。
      ii. PVC **Name** (例: **myclaim**) を指定します。
      iii. 必要な **Access Mode** を選択します。

      注記

      IBM FlashSystem では **Access Mode** の **Shared access (RWX)** はサポートされません。

      iv. Rados Block Device (RBD) の場合は、**Access mode** が ReadWriteOnce (**RWO**) であれば、必須の **Volume mode** を選択します。デフォルトのボリュームモードは、**Filesystem** です。
      v. アプリケーション要件に応じて **Size** を指定します。
      vi. **Create** をクリックし、PVC のステータスが **Bound** になるまで待機します。
2. **新規または既存のアプリケーション Pod を新規 PVC を使用するように設定します。**
   - 新規アプリケーション Pod の場合は、以下の手順を実行します。
     i. **Workloads →Pods** をクリックします。

ii. 新規アプリケーション Pod を作成します。

iii. **spec:** セクションの下に **volumes:** セクションを追加し、新規 PVC をアプリケーション Pod のボリュームとして追加します。

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

以下に例を示します。

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

- 既存のアプリケーション Pod の場合、以下の手順を実行します。

i. **Workloads →Deployment Configs** をクリックします。

ii. アプリケーション Pod に関連付けられた必要なデプロイメント設定を検索します。

iii. **Action メニュー (⋮) → Edit Deployment Config** をクリックします。

iv. **spec:** セクションの下に **volumes:** セクションを追加し、新規 PVC をアプリケーション Pod のボリュームとして追加し、**Save** をクリックします。

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

以下に例を示します。

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

3. **新しい設定が使用されていることを確認します。**

a. **Workloads → Pods** をクリックします。

b. アプリケーション Pod の **Project** を設定します。

c. アプリケーション Pod が **Running** ステータスで表示されていることを確認します。

d. アプリケーション Pod 名をクリックし、Pod の詳細を表示します。

e. **Volumes** セクションまでスクロールダウンし、ボリュームに新規永続ボリューム要求に一致する **Type** があることを確認します (例: **myclaim**)。

9.2. 永続ボリューム要求の要求ステータスの表示

以下の手順を使用して、PVC 要求のステータスを表示します。

前提条件

- OpenShift Data Foundation への管理者アクセス。

手順

1. OpenShift Web コンソールにログインします。
2. **Storage → Persistent Volume Claims**をクリックします。
3. **Filter** テキストボックスを使用して、必要な PVC 名を検索します。また、リストを絞り込むために Name または Label で PVC のリストをフィルターすることもできます。
4. 必要な PVC に対応する **Status** 列を確認します。
5. 必要な **Name** をクリックして PVC の詳細を表示します。

9.3. 永続ボリューム要求の要求イベントの確認

以下の手順を使用して、永続ボリューム要求 (PVC) の要求イベントを確認し、これに対応します。

前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

1. OpenShift Web Console で、**Storage → Data Foundation**をクリックします。
2. **Storage systems** タブでストレージシステムを選択し、**Overview → Block and File** タブをクリックします。
3. **Inventory** カードを見つけ、エラーのある PVC の数を確認します。
4. **Storage → Persistent Volume Claims**をクリックします。
5. **Filter** テキストボックスを使用して、必要な PVC を検索します。
6. PVC 名をクリックし、**Events** に移動します。
7. 必要に応じて、または指示に応じてイベントに対応します。

9.4. 永続ボリューム要求の拡張

OpenShift Data Foundation 4.6 以降では、永続ボリューム要求を拡張する機能が導入され、永続ストレージリソース管理の柔軟性が向上します。

拡張は、以下の永続ボリュームでサポートされます。

- ボリュームモードが **Filesystem** の Ceph File System (CephFS) をベースとする PVC (ReadWriteOnce (RWO) および ReadWriteMany (RWX) アクセス)。

- ボリュームモードが **Filesystem** の Ceph RADOS Block Device (Ceph RBD) をベースとする PVC (ReadWriteOnce (RWO) アクセス)。
- ボリュームモードが **Block** の Ceph RADOS Block Device (Ceph RBD) をベースとする PVC (ReadWriteOnce (RWO) アクセス)。
- ボリュームモードが **Filesystem** の Ceph File System (CephFS) または Network File System (NFS) に基づく ReadWriteOncePod (RWOP) が指定された PVC。
- ボリュームモードが **Filesystem** の Ceph RADOS Block Device (Ceph RBD) をベースとする PVC (ReadWriteOncePod (RWOP) アクセス)。RWOP アクセスモードでは、単一ノードの単一 Pod によってボリュームを読み取り/書き込みとしてマウントします。

注記

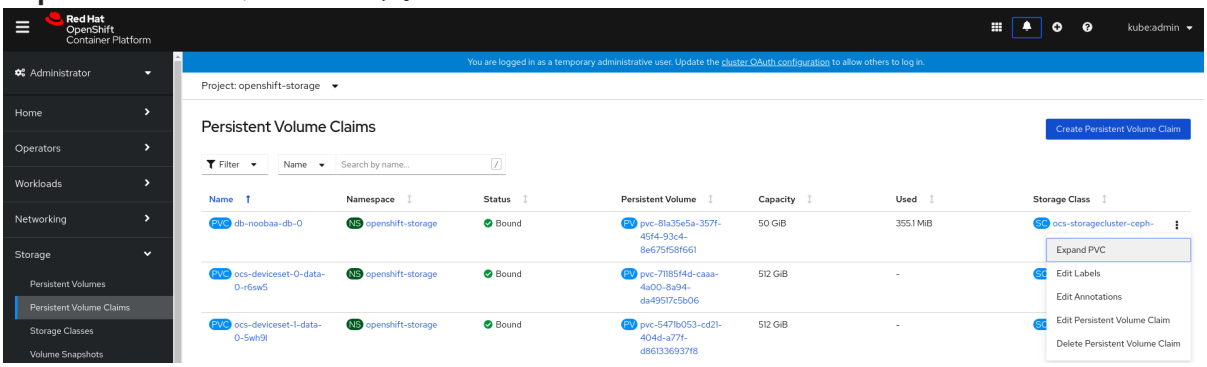
PVC の拡張は OSD、MON、および暗号化された PVC ではサポートされません。

前提条件

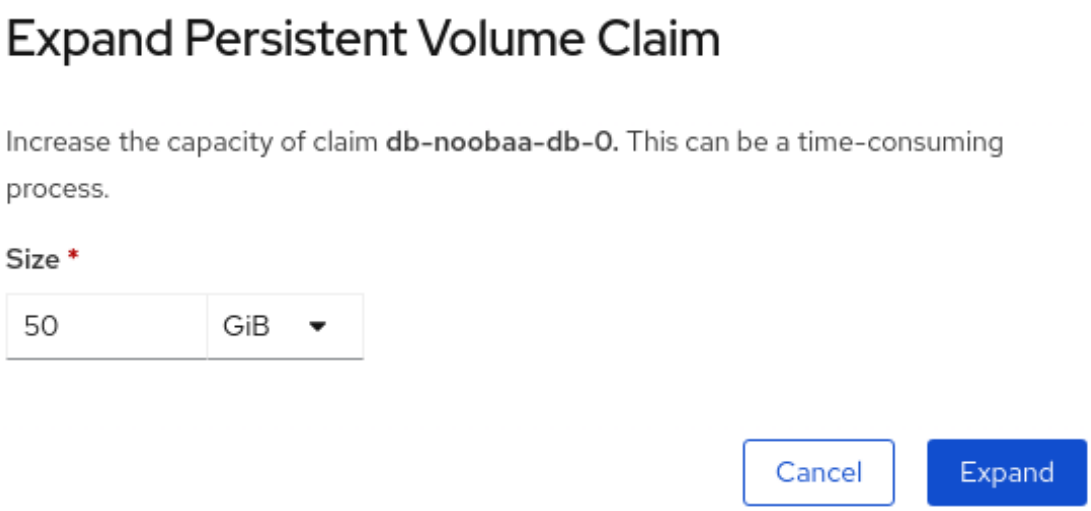
- OpenShift Web コンソールへの管理者アクセス。

手順

1. OpenShift Web コンソールで、**Storage → Persistent Volume Claims** に移動します。
2. 拡張する永続ボリューム要求の横にある Action メニュー (⋮) をクリックします。
3. **Expand PVC** をクリックします。



4. 永続ボリューム要求の新しいサイズを選択してから、**Expand** をクリックします。



5. 拡張を確認するには、PVC の詳細ページに移動し、**Capacity** フィールドでサイズが正しく要求されていることを確認します。

注記

Ceph RADOS Block Device (RBD) に基づいて PVC を拡張し、PVC がまだ Pod に割り当てられていない場合は、PVC の詳細ページで **Condition type** が **FileSystemResizePending** になります。ボリュームをマウントすると、ファイルシステムのサイズ変更が正常に実行し、新しいサイズが **Capacity** フィールドに反映されます。

9.5. 動的プロビジョニング

9.5.1. 動的プロビジョニングについて

StorageClass リソースオブジェクトは、要求可能なストレージを記述し、分類するほか、要求に応じて動的にプロビジョニングされるストレージのパラメーターを渡すための手段を提供します。StorageClass オブジェクトは、さまざまなレベルのストレージおよびストレージへのアクセスを制御するための管理メカニズムとしても機能します。クラスター管理者 (**cluster-admin**) またはストレージ管理者 (**storage-admin**) は、ユーザーが基礎となるストレージボリュームソースに関する詳しい知識なしに要求できる StorageClass オブジェクトを定義し、作成します。

OpenShift Container Platform の永続ボリュームフレームワークはこの機能を有効にし、管理者がクラスターに永続ストレージをプロビジョニングできるようにします。フレームワークにより、ユーザーは基礎となるインフラストラクチャーの知識がなくてもこれらのリソースを要求できるようになります。

OpenShift Container Platform では、数多くのストレージタイプを永続ボリュームとして使用することができます。ストレージプラグインは、静的プロビジョニング、動的プロビジョニング、または両方のプロビジョニングタイプをサポートする場合があります。

9.5.2. OpenShift Data Foundation での動的プロビジョニング

Red Hat OpenShift Data Foundation は、コンテナー環境向けに最適化されたソフトウェアで定義されるストレージです。このソフトウェアは、OpenShift Container Platform の Operator として実行されるため、コンテナーの永続ストレージ管理を高度に統合し、簡素化できます。

OpenShift Data Foundation は、以下を含む各種のストレージタイプをサポートします。

- データベースのブロックストレージ
- 継続的な統合、メッセージングおよびデータ集約のための共有ファイルストレージ
- アーカイブ、バックアップ、およびメディアストレージのオブジェクトストレージ

バージョン 4 では、Red Hat Ceph Storage を使用して永続ボリュームをサポートするファイル、ブロック、およびオブジェクトストレージを提供し、Rook.io を使用して永続ボリュームおよび要求のプロビジョニングを管理し、オーケストレーションします。NooBaa はオブジェクトストレージを提供し、その Multicloud Gateway は複数のクラウド環境でのオブジェクトのフェデレーションを可能にします (テクノロジープレビューとしてご利用いただけます)。

OpenShift Data Foundation 4 では、RADOS Block Device (RBD) および Ceph File System (CephFS) の Red Hat Ceph Storage Container Storage Interface (CSI) ドライバーが動的プロビジョニング要求を処理します。PVC 要求が動的に送信される場合は、CSI ドライバーで以下のオプションを使用できます。

- ボリュームモードが **Block** の Ceph RBD をベースとする PVC (ReadWriteOnce (RWO) および ReadWriteMany (RWX) アクセス) を作成します。
- ボリュームモードが **Filesystem** の Ceph RBD をベースとする PVC (ReadWriteOnce (RWO) アクセス) を作成します。
- ボリュームモードが **Filesystem** の CephFS をベースとする PVC (ReadWriteOnce (RWO) および ReadWriteMany (RWX) アクセス) を作成します。
- CephFS、NFS、および RBD に基づく ReadWriteOncePod (RWOP) アクセスが割り当てられた PVC を作成します。RWOP アクセスモードでは、単一ノードの単一 Pod によってボリュームを読み取り/書き込みとしてマウントします。

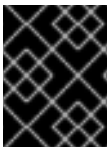
使用するドライバー (RBD または CephFS) の判断は、**storageclass.yaml** ファイルのエントリーに基づいて行われます。

9.5.3. 利用可能な動的プロビジョニングプラグイン

OpenShift Container Platform は、以下のプロビジョナープラグインを提供します。 これらには、クラスターの設定済みプロバイダーの API を使用して新規ストレージリソースを作成する動的プロビジョニング用の一般的な実装が含まれます。

ストレージタイプ	プロビジョナープラグインの名前	注記
OpenStack Cinder	**kubernetes.io/cinder**	
AWS Elastic Block Store (EBS)	**kubernetes.io/aws-ebs**	複数クラスターを複数の異なるゾーンで使用する際の動的プロビジョニングの場合は、各ノードに **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>** のタグを付けます。ここで、**<cluster_name>** および **<cluster_id>** はクラスターごとに固有の値になります。

ストレージタイプ	プロビジョナープラグインの名前	注記
AWS Elastic File System (EFS)		動的プロビジョニングは、EFS プロビジョナー Pod で実行され、プロビジョナープラグインでは実行されません。
Azure Disk	**kubernetes.io/azure-disk**	
Azure File	**kubernetes.io/azure-file**	**persistent-volume-binder** ServiceAccount では、Azure ストレージアカウントおよびキーを保存するためにシークレットを作成し、取得するためのパーミッションが必要です。
GCE Persistent Disk (gcePD)	**kubernetes.io/gce-pd**	マルチゾーン設定では、GCE プロジェクトごとに OpenShift Container Platform クラスターを実行し、現行クラスターのノードが存在しないゾーンで PV が作成されないようにすることが推奨されます。
VMware vSphere	**kubernetes.io/vsphere-volume**	
Red Hat Virtualization	**csi.ovirt.org**	



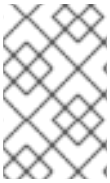
重要

選択したプロビジョナープラグインでは、関連するクラウド、ホスト、またはサードパーティープロバイダーを、関連するドキュメントに従って設定する必要もあります。

第10章 ターゲットボリュームのスペースを再利用

削除されたファイルまたはゼロデータのチャンクは、Ceph クラスター上のストレージ領域を占有する場合があり、その結果、利用可能なストレージ領域が不正確に報告されます。スペースの再利用操作では、ターゲットボリュームに対して次の操作を実行することで、このような不一致を削除します。

- **fstrim**: この操作は、**Filesystem** モードのボリューム上で、スペース再利用操作の実行時にボリュームが Pod にマウントされている場合にのみ実行されます。
- **rbd sparsify**: この操作は、ボリュームがどの Pod にも接続されていないときに実行し、4M サイズのゼロ化データのチャンクによって占められていたスペースを回収します。



注記

- スペースの回収操作は、Ceph RBD ボリュームでのみサポートされます。
- スペースのの回収操作の実行時には、パフォーマンスが低下します。

以下のいずれかの方法を使用して、スペースを回収できます。

- Annotating PersistentVolumeClaims を使用したスペース再利用操作の有効化 (スペースの回収操作を有効にするために使用する推奨方法)
- ReclaimSpaceJob を使用したスペースの回収操作の有効化
- ReclaimSpaceCronJob を使用したスペースの回収操作の有効化

10.1. ANNOTATING PERSISTENTVOLUMECLAIMS を使用してスペースの回収操作を有効にする

この手順を使用して **PersistentVolumeClaims** にアノテーションを付け、指定されたスケジュールに基づいてスペースの回収操作を自動的に呼び出すことができるようにします。

注記

- スケジュール値は、定期的な操作リクエストの間隔を設定する Kubernetes CronJob と同じ形式です。
- 推奨されるスケジュール間隔は **@weekly** です。スケジュール間隔の値が空であるか無効な形式の場合、デフォルトのスケジュール値は **@weekly** に設定されます。複数の **ReclaimSpace** 操作を **@weekly** または同時にスケジュールしないでください。
- スケジュールされた各操作間のサポートされる最小間隔は、少なくとも 24 時間です。たとえば、**@daily** (毎日 00:00) または **0 3 * * *** (毎日 3:00) です。
- **ReclaimSpace** 操作は、オフピーク、メンテナンス時間帯、またはワークロードの入出力が低いと予想される時間帯にスケジュールします。
- **ReclaimSpaceCronJob** は、**schedule** が変更されると再作成されます。アノテーションが削除されると自動的に削除されます。

手順

1. 永続ボリュームクレーム (PVC) の詳細を取得します。

```
$ 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. アノテーション **reclaimspace.csiaddons.openshift.io/schedule=@monthly** を PVC に追加して、**reclaimspacecronjob** を作成します。

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

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

3. **reclaimspacecronjob** が **"<pvc-name>-xxxxxxx"** の形式で作成されていることを確認します。

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

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

4. このジョブを自動的に実行するようにスケジュールを変更します。

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

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

5. **reclaimspacecronjob** のスケジュールが変更されていることを確認します。

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

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

10.2. RECLAIMSPACEJOB を使用したスペースの回収操作の有効化

ReclaimSpaceJob は、ターゲットボリュームでスペースの回収操作を呼び出すように設計された名前付きのカスタムリソース (CR) です。これは、スペースの回収操作をすぐに開始する 1 回限りの方法です。ユーザーは、必要に応じて **ReclaimSpaceJob** CR の作成を繰り返して、スペースの回収操作を繰り返す必要があります。

注記

- スペース回収操作の推奨間隔は **weekly** です。
- 各操作間の最小間隔は少なくとも **24 hours** であることを確認してください。
- スペース回収操作は、オフピーク時、メンテナンス時間帯、またはワークロードの入出力が低いと予想されるときにスケジュールします。

手順

1. スペースの回収操作用に次のカスタムリソースを作成して適用します。

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

詳細は以下のようになります。

target

操作が実行されるボリュームターゲットを示します。

persistentVolumeClaim

PersistentVolumeClaim の名前。

backOfflimit

スペース回収操作を **failed** としてマークするまでの最大再試行回数を指定します。デフォルト値は **6** です。許可される最大値と最小値はそれぞれ **60** と **0** です。

retryDeadlineSeconds

操作が終了するまでの期間を秒単位で指定します。これは開始時刻を基準とします。値は正の整数である必要があります。デフォルト値は **600** 秒で、許容最大値は **1800** 秒です。

timeout

CSI ドライバーに送信される **grpc** 要求のタイムアウトを秒単位で指定します。タイムアウト値が指定されていない場合、デフォルトは global reclaimspace timeout の値に設定されます。タイムアウトの最小許容値は 60 です。

2. 操作が完了したら、カスタムリソースを削除します。

10.3. RECLAIMSPACECRONJOB を使用したスペースの回収操作の有効化

ReclaimSpaceCronJob は毎日、毎週などの指定されたスケジュールに基づいてスペースの回収操作を呼び出します。**ReclaimSpaceCronJob** は永続ボリューム要求に対して 1 回だけ作成する必要があります。スケジュール属性を使用すると、**CSI-addons** コントローラーは、要求された時間と間隔で **ReclaimSpaceJob** を作成します。

注記

- 推奨されるスケジュール間隔は **@weekly** です。
- スケジュールされた各操作間の最小間隔は、少なくとも 24 時間である必要があります。たとえば、**@daily** (毎日 00:00) または "0 3 * * *" (毎日 3:00) です。
- ReclaimSpace 操作は、オフピーク、メンテナンス時間帯、またはワークロードの入出力が低いと予想される時間帯にスケジュールします。

手順

1. スペースの回収操作用に次のカスタムリソースを作成して適用します

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

詳細は以下のようになります。

concurrencyPolicy

以前の **ReclaimSpaceJob** がまだ実行されている間に、新しい **ReclaimSpaceJob** が **ReclaimSpaceCronJob** によってスケジュールされた場合の変更を示します。デフォルトの **Forbid** は新しいジョブの開始を防ぎますが、**Replace** を使用すると、障害状態にある可能性のある実行中のジョブを削除して、新しいジョブを作成できます。

failedJobsHistoryLimit

トラブルシューティングのために保持される、失敗した **ReclaimSpaceJobs** の数を指定します。

jobTemplate

要求された **ReclaimSpaceJob** 操作の詳細を記述する **ReclaimSpaceJob.spec** 構造体を指定します。

successfulJobsHistoryLimit

成功した **ReclaimSpaceJob** 操作の数を指定します。

schedule

定期的な操作リクエストの間隔を指定します。Kubernetes CronJobs と同じ形式です。

2. スペース回収操作の実行が不要になった場合、またはターゲット PVC が削除された場合は、**ReclaimSpaceCronJob** カスタムリソースを削除します。

10.4. スペース再利用操作に必要なタイムアウトのカスタマイズ

RBD ボリュームのサイズとそのデータパターンによっては、スペースの再利用操作が **context deadline exceeded** エラーで失敗する可能性があります。タイムアウト値を増やすことで、これを回避できます。

次の例では、対応する **ReclaimSpaceJob** の **-o yaml** を検査することで、失敗したステータスを示しています。

例

```
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
```

次の **configmap** を作成して、グローバルレベルでカスタムタイムアウトを設定することもできます。

例

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

csi-addons Operator Pod を再起動します。

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

上記の **configmap** の作成後に開始されたすべてのスペース再利用操作では、カスタマイズされたタイムアウトが使用されます。

' :leveloffset: +1

第11章 古いサブボリュームの検出と消去 (テクノロジープレビュー)

古いサブボリュームには、対応する **k8s** 参照が割り当てられていない場合があります。このようなサブボリュームは不要なため、削除できます。ODF CLI ツールを使用して、古くなったサブボリュームを見つけて削除できます。

重要

ODF CLI ツールを使用して古いサブボリュームを削除する機能は、テクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

1. カスタマーポータル から ODF CLI ツールをダウンロードします。

手順

1. **subvolumes** コマンドで **--stale** フラグを使用して、古いサブボリュームを見つけます。

```
$ odf subvolume ls --stale
```

出力例:

```
# Filesystem  Subvolume Subvolumegroup State
# ocs-storagecluster-cephfilesystem csi-vol-427774b4-340b-11ed-8d66-0242ac110004 csi stale
# ocs-storagecluster-cephfilesystem csi-vol-427774b4-340b-11ed-8d66-0242ac110005 csi stale
```

2. 古くなったサブボリュームを削除します。

```
odf subvolume delete <subvolumes> <filesystem> <subvolumegroup>
```

<subvolumes> は、最初のコマンドの出力から得られるサブボリュームのコンマ区切りリストに置き換えます。サブボリュームは、ファイルシステムとサブボリュームグループが同じである必要があります。

<filesystem> と **<subvolumegroup>** は、最初のコマンドの出力から得られるファイルシステムとサブボリュームグループに置き換えます。

以下に例を示します。

```
odf subvolume delete csi-vol-427774b4-340b-11ed-8d66-0242ac110004,csi-vol-427774b4-340b-11ed-8d66-0242ac110005 ocs-storagecluster csi
```

出力例:

```
# Info: subvolume csi-vol-427774b4-340b-11ed-8d66-0242ac110004 deleted
# Info: subvolume csi-vol-427774b4-340b-11ed-8d66-0242ac110004 deleted
```

第12章 ボリュームスナップショット

ボリュームスナップショットは、特定の時点におけるクラスター内のストレージボリュームの状態を表します。これらのスナップショットは、毎回フルコピーを作成する必要がないため、より効率的にストレージを使用するのに役立ち、アプリケーション開発のビルディングブロックとして使用できます。

ボリュームスナップショットクラスを使用すると、管理者はボリュームスナップショットオブジェクトに属する異なる属性を指定できます。OpenShift Data Foundation Operator は、使用されるプラットフォームに応じてデフォルトのボリュームスナップショットクラスをインストールします。これらのデフォルトボリュームスナップショットクラスは Operator によって所有され、制御されるため、削除したり変更したりすることはできません。

同じ永続ボリューム要求 (PVC) のスナップショットを複数作成できますが、スナップショットの定期的な作成をスケジュールすることはできません。

- CephFS の場合は、PVC ごとに最大 100 スナップショットを作成できます。
- RADOS Block Device (RBD) の場合は、PVC ごとに最大 512 スナップショットを作成できます。

注記

永続ボリュームの暗号化がボリュームのスナップショットをサポートするようになりました。

12.1. ボリュームスナップショットの作成

Persistent Volume Claim ページまたは Volume Snapshots ページのいずれかからボリュームスナップショットを作成できます。

前提条件

- 一貫性のあるスナップショットを使用するには、PVC は **Bound** 状態にあり、使用されていない必要があります。スナップショットを作成する前に、必ずすべての IO を停止してください。

注記

OpenShift Data Foundation は、Pod が PVC のボリュームスナップショットを使用している場合のみ、その PVC のボリュームスナップショットのクラッシュー貫性を提供します。アプリケーションの一貫性を保つために、まず実行中の Pod を破棄してスナップショットの一貫性を確保するか、アプリケーションが提供する静止メカニズムを使用してこれを確保します。

手順

Persistent Volume Claims ページで以下を実行します。

1. OpenShift Web コンソールで、**Storage → Persistent Volume Claims**をクリックします。
2. ボリュームのスナップショットを作成するには、以下のいずれかを実行します。
   - 必要な PVC の横にある Action メニュー (⋮) → **Create Snapshot** をクリックします。
   - スナップショットを作成する PVC をクリックし、**Actions → Create Snapshot** をクリックします。

3. ボリュームスナップショットの **Name** を入力します。
4. ドロップダウンリストから **Snapshot Class** を選択します。
5. **Create** をクリックします。作成されるボリュームスナップショットの Details ページにリダイレクトされます。

Volume Snapshots ページで以下を実行します。

1. OpenShift Web コンソールで **Storage → Volume Snapshots** をクリックします。
2. **Volume Snapshots** ページで、**Create Volume Snapshot** をクリックします。
3. ドロップダウンリストから必要な **Project** を選択します。
4. ドロップダウンリストから **Persistent Volume Claim** を選択します。
5. スナップショットの **Name** を入力します。
6. ドロップダウンリストから **Snapshot Class** を選択します。
7. **Create** をクリックします。作成されるボリュームスナップショットの Details ページにリダイレクトされます。

検証手順

- PVC の **Details** ページに移動し、**Volume Snapshots** タブをクリックしてボリュームスナップショットのリストを表示します。新規スナップショットがリスト表示されていることを確認します。
- OpenShift Web コンソールで **Storage → Volume Snapshots** をクリックします。新規スナップショットがリスト表示されていることを確認します。
- ボリュームスナップショットが **Ready** 状態になるまで待機します。

12.2. ボリュームスナップショットの復元

ボリュームスナップショットを復元する際に、新規の永続ボリューム要求が作成されます。復元される PVC はボリュームスナップショットおよび親 PVC とは切り離されています。

Persistent Volume Claim ページまたは Volume Snapshots ページのいずれかからボリュームスナップショットを復元できます。

手順

Persistent Volume Claims ページで以下を実行します。

親 PVC が存在する場合に限り、Persistent Volume Claims ページからボリュームスナップショットを復元できます。

1. OpenShift Web コンソールで、**Storage → Persistent Volume Claims** をクリックします。
2. ボリュームスナップショットと共に PVC 名をクリックし、ボリュームスナップショットを新規 PVC として復元します。

3. **Volume Snapshots** タブで、復元するボリュームスナップショットの横にある Action メニュー (⋮) をクリックします。
4. **Restore as new PVC** をクリックします。
5. 新規 PVC の名前を入力します。
6. **Storage Class** 名を選択します。
7. 任意の **Access Mode** を選択します。

重要

ReadOnlyMany (ROX) アクセスモードは開発者プレビュー機能であり、開発者プレビューのサポート制限の対象となります。開発者プレビューリリースは、実稼働環境で実行することは意図されておらず、Red Hat カスタマーポータルのケース管理システムではサポートされません。ReadOnlyMany 機能に関してサポートが必要な場合は、ocs-devpreview@redhat.com メーリングリストに連絡してください。Red Hat Development Team のメンバーが稼働状況とスケジュールに応じて可能な限り迅速に対応します。ROX アクセスモードの使用については、Creating a clone or restoring a snapshot with the new readonly access mode を参照してください。

8. オプション: RBD の場合は、**Volume mode** を選択します。
9. **Restore** をクリックします。新規 PVC の詳細ページにリダイレクトされます。

Volume Snapshots ページで以下を実行します。

1. OpenShift Web コンソールで **Storage → Volume Snapshots** をクリックします。
2. **Volume Snapshots** タブで、復元するボリュームスナップショットの横にある Action メニュー (⋮) をクリックします。
3. **Restore as new PVC** をクリックします。
4. 新規 PVC の名前を入力します。
5. **Storage Class** 名を選択します。
6. 任意の **Access Mode** を選択します。

重要

ReadOnlyMany (ROX) アクセスモードは開発者プレビュー機能であり、開発者プレビューのサポート制限の対象となります。開発者プレビューリリースは、実稼働環境で実行することは意図されておらず、Red Hat カスタマーポータルのケース管理システムではサポートされません。ReadOnlyMany 機能に関してサポートが必要な場合は、ocs-devpreview@redhat.com メーリングリストに連絡してください。Red Hat Development Team のメンバーが稼働状況とスケジュールに応じて可能な限り迅速に対応します。ROX アクセスモードの使用については、Creating a clone or restoring a snapshot with the new readonly access mode を参照してください。

7. オプション: RBD の場合は、**Volume mode** を選択します。

8. **Restore** をクリックします。新規 PVC の詳細ページにリダイレクトされます。

検証手順

- OpenShift Web コンソールから **Storage → Persistent Volume Claims** をクリックし、新規 PVC が **Persistent Volume Claims** ページにリスト表示されていることを確認します。
- 新規 PVC が **Bound** の状態になるまで待機します。

12.3. ボリュームスナップショットの削除

前提条件

- ボリュームスナップショットを削除する場合は、その特定のボリュームスナップショットで使用されるボリュームスナップショットクラスが存在している必要があります。

手順

Persistent Volume Claims ページで以下を実行します。

1. OpenShift Web コンソールで、**Storage → Persistent Volume Claims** をクリックします。
2. 削除する必要のあるボリュームスナップショットがある PVC 名をクリックします。
3. **Volume Snapshots** タブで、必要なボリュームスナップショットの横にある Action メニュー (⋮) → **Delete Volume Snapshot** をクリックします。

Volume Snapshots ページで以下を実行します。

1. OpenShift Web コンソールで **Storage → Volume Snapshots** をクリックします。
2. **Volume Snapshots** ページで、必要なスナップショットの横にある Action メニュー (⋮) → **Delete Volume Snapshot** をクリックします。

検証手順

- 削除されたボリュームスナップショットが PVC の詳細ページの **Volume Snapshots** タブにないことを確認します。
- **Storage → Volume Snapshots** をクリックし、削除されたボリュームスナップショットがリスト表示されていないことを確認します。

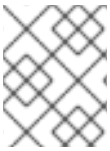
第13章 ボリュームのクローン作成

クローンは、標準のボリュームとして使用される既存のストレージボリュームの複製です。ボリュームのクローンを作成し、データの特定の時点のコピーを作成します。永続ボリューム要求は別のサイズでクローンできません。CephFS および RADOS Block Device (RBD) の両方で、PVC ごとに最大 512 のクローンを作成できます。

13.1. クローンの作成

前提条件

- ソース PVC は **Bound** 状態にある必要があり、使用中の状態にすることはできません。

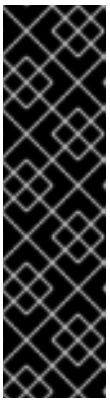


注記

Pod が PVC を使用している場合は、PVC のクローンを作成しません。これを実行すると、PVC が一時停止 (停止) されないため、データが破損する可能性があります。

手順

1. OpenShift Web コンソールで、**Storage → Persistent Volume Claims** をクリックします。
2. クローンを作成するには、以下のいずれかを実行します。
   - 必要な PVC の横にある Action メニュー **(⋮) → Clone PVC** をクリックします。
   - クローンを作成する必要のある PVC をクリックし、**Actions → Clone PVC** をクリックします。
3. クローンの **Name** を入力します。
4. 任意のアクセスモードを選択します。



重要

ReadOnlyMany (ROX) アクセスモードは開発者プレビュー機能であり、開発者プレビューのサポート制限の対象となります。開発者プレビューリリースは、実稼働環境で実行することは意図されておらず、Red Hat カスタマーポータルのケース管理システムではサポートされません。ReadOnlyMany 機能に関してサポートが必要な場合は、ocs-devpreview@redhat.com メーリングリストに連絡してください。Red Hat Development Team のメンバーが稼働状況とスケジュールに応じて可能な限り迅速に対応します。ROX アクセスモードの使用については、Creating a clone or restoring a snapshot with the new readonly access mode を参照してください。

5. 必要なクローンのサイズを入力します。
6. クローンを作成するストレージクラスを選択します。
   ストレージクラスは任意の RBD ストレージクラスにすることができ、必ずしも親 PVC と同じである必要はありません。
7. **Clone** をクリックします。新規 PVC の詳細ページにリダイレクトされます。
8. クローン作成された PVC のステータスが **Bound** になるまで待機します。

クローン作成された PVC が Pod で使用できるようになります。このクローン作成された PVC は dataSource PVC とは切り離されています。

第14章 CONTAINER STORAGE INTERFACE (CSI) コンポーネントの配置の管理

各クラスターは、**infra** や **storage** ノードなどの数多くの専用ノードで構成されます。ただし、カスタムテイントを持つ **infra** ノードは、ノードで OpenShift Data Foundation 永続ボリューム要求を使用することができません。そのため、このようなノードを使用する必要がある場合は、容認を設定してノードで **csi-plugins** を起動することができます。

手順

1. configmap を編集して、カスタムテイントの容認を追加します。エディターを終了する前に必ず保存します。

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

2. **configmap** を表示して、追加された容認を確認します。

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

テイント **nodetype=infra:NoSchedule** の追加された容認の出力例

```
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:
[...]
```

注記

容認の **value** フィールドで、すべての文字列以外の値に二重引用符が含まれていることを確認してください。たとえば、boolean 型の値 **true** と int 型の値 **1** は、"true" と "1" という形で入力する必要があります。

3. 独自の infra ノードで **csi-cephfsplugin-*** および **csi-rbdplugin-*** Pod の起動に失敗した場合は、**rook-ceph-operator** を再起動します。

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

例:

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

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

検証手順

csi-cephfsplugin-* および **csi-rbdplugin-*** Pod が **infra** ノードで実行されていることを確認します。

第15章 NFS を使用したエクスポートの作成

このセクションでは、NFS を使用してエクスポートを作成し、OpenShift クラスターから外部でアクセスできるようにする方法を説明します。

以下の手順に従って、エクスポートを作成し、OpenShift クラスターから外部でアクセスします。

- 「NFS 機能の有効化」
- 「NFS エクスポートの作成」
- 「クラスター内での NFS エクスポートの使用」
- 「OpenShift クラスターから外部での NFS エクスポートの使用」

15.1. NFS 機能の有効化

NFS 機能を使用するには、クラスターの作成後にコマンドラインインターフェイス (CLI) を使用してストレージクラスター内で NFS 機能を有効にする必要があります。ユーザーインターフェイスを使用してストレージクラスターを作成するときに NFS 機能を有効にすることもできます。

前提条件

- OpenShift Data Foundation が **openshift-storage** namespace にインストールされ、実行されている。
- OpenShift Data Foundation のインストールに CephFilesystem が含まれている。

手順

- 次のコマンドを実行して、CLI から NFS 機能を有効にします。

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

検証手順

NFS のインストールと設定は、次の条件が満たされると完了します。

- **ocs-storagecluster-cephnfs** という名前の CephNFS リソースのステータスは **Ready** です。
- すべての **csi-nfsplugin-*** Pod が実行されているかどうかを確認します。

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

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

出力には複数の Pod があります。以下に例を示します。

```
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
```

15.2. NFS エクスポートの作成

NFS エクスポートは、**ocs-storagecluster-ceph-nfs** StorageClass に対して永続ボリューム要求 (PVC) を作成することによって作成されます。

NFS PVC は、次の 2 つの方法で作成できます。

yaml を使用した NFS PVC の作成

以下は、PVC の例になります。

注記

volumeMode: Block は NFS ボリュームでは機能しません。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: <desired_name>
spec:
 accessModes:
   - ReadWriteOnce
 resources:
   requests:
     storage: 1Gi
 storageClassName: ocs-storagecluster-ceph-nfs
```

<desired_name>

my-nfs-export など、PVC の名前を指定します。

PVC が **Bound** 状態になると、エクスポートが作成されます。

OpenShift Container Platform Web コンソールからの NFS PVC の作成

前提条件

- OpenShift Container Platform Web コンソールにログインし、ストレージクラスターに対して NFS 機能が有効になっていることを確認している。

手順

1. OpenShift Web コンソールで、**Storage → Persistent Volume Claims**をクリックします。
2. **Project** を **openshift-storage** に設定します。
3. **Create PersistentVolumeClaim** をクリックします。
   a. **Storage Class**、**ocs-storagecluster-ceph-nfs** を指定します。
   b. **my-nfs-export** などの **PVC 名** を指定します。
   c. 必要な **Access Mode** を選択します。
   d. アプリケーション要件に応じて **Size** を指定します。

e. **Volume mode** を **Filesystem** として選択します。
注記: **Block** モードは、NFS PVC ではサポートされていません。

f. **Create** をクリックし、PVC のステータスが **Bound** になるまで待機します。

15.3. クラスター内での NFS エクスポートの使用

Kubernetes アプリケーション Pod は、以前に作成された PVC をマウントすることによって作成された NFS エクスポートを使用することができます。

以下の 2 つの方法のいずれかで PVC をマウントすることができます。

YAML の使用:

以下は、「NFS エクスポートの作成」 で作成されたサンプル PVC を使用する Pod の例です。

```
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>

my-nfs-export など、以前に作成した PVC を指定します。

OpenShift Container Platform Web コンソールの使用

手順

1. OpenShift Container Platform Web コンソールで、**Workloads → Pods** に移動します。
2. **Create Pod** をクリックして、新しいアプリケーション Pod を作成します。
3. **metadata** セクションの下に名前を追加します。たとえば、**namespace** のある **nfs-export-example** を **openshift-storage** として追加します。
4. **spec:** セクションの下に、**image** セクションと **volumeMounts** セクションを含む **containers:** セクションを追加します。

```
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
```

以下に例を示します。

```
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. **spec:** セクションの下に、**volume:** セクションを追加して、NFS PVC をアプリケーション Pod のボリュームとして追加します。

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

以下に例を示します。

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

15.4. OPENSHIFT クラスターから外部での NFS エクスポートの使用

OpenShift クラスター外の NFS クライアントは、以前に作成された PVC によって作成された NFS エクスポートをマウントできます。

手順

1. **nfs** フラグが有効になると、単一サーバーの CephNFS が Rook によってデプロイされます。次のステップで使用する **nfs-ganesha** サーバーの **ceph_nfs** フィールドの値を取得する必要があります。

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

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

以下に例を示します。

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

2. Kubernetes LoadBalancer Service を作成して、OpenShift クラスターの外部に NFS サーバーを公開します。以下の例では、LoadBalancer Service を作成し、OpenShift Data Foundation によって作成された NFS サーバーを参照します。

```
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
```

<my-nfs> を手順 1 で取得した値に置き換えます。

3. 接続情報を収集します。外部クライアントがエクスポートに接続するために必要な情報は、PVC 用に作成された永続ボリューム (PV) と、前の手順で作成された LoadBalancer Service のステータスから取得されます。

   a. PV から共有パスを取得します。

      i. NFS エクスポートの PVC に関連付けられた PV の名前を取得します。

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

      <pvc_name> を独自の PVC 名に置き換えます。以下に例を示します。

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

      ii. 前に取得した PV 名を使用して、NFS エクスポートの共有パスを取得します。

```
$ 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. NFS サーバーの Ingress アドレスを取得します。サービスの Ingress ステータスには、複数のアドレスが存在する場合があります。外部クライアントに使用するアドレスを選択しま

す。以下の例では、ホスト名 **ingress-id.somedomain.com** という 1 つのアドレスしかありません。

```
$ 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. 前の手順の共有パスと Ingress アドレスを使用して、外部クライアントを接続します。次の例では、エクスポートをクライアントのディレクトリーパス **/export/mount/path** にマウントします。

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

これがすぐに機能しない場合は、Kubernetes 環境が、NFS サーバーへの Ingress を許可するためのネットワークリソースの設定に、まだ時間がかかっている可能性があります。

第16章 暗号化された RBD ストレージクラスへのアノテーション設定

OpenShift Data Foundation 4.14 以降では、OpenShift コンソールが暗号化を有効にして RADOS ブロックデバイス (RBD) ストレージクラスを作成すると、アノテーションが自動的に設定されます。ただし、OpenShift Data Foundation バージョン 4.14 に更新する前に作成された暗号化された RBD ストレージクラスのいずれかに対して、アノテーション **cdi.kubevirt.io/clone-strategy=copy** を追加する必要があります。これにより、顧客データ統合 (CDI) で、デフォルトのスマートクローン作成の代わりにホスト支援型のクローン作成を使用できるようになります。

暗号化されたボリュームへのアクセスに使用されるキーは、ボリュームが作成された namespace に関連付けられます。新しい OpenShift Virtualization 仮想マシンのプロビジョニングなど、暗号化されたボリュームを新しい namespace にクローン作成する場合は、新しいボリュームを作成し、ソースボリュームのコンテンツを新しいボリュームにコピーする必要があります。ストレージクラスに適切にアノテーションが付けられている場合は、この動作が自動的にトリガーされます。

第17章 OSD バックフィル中にクライアント IO またはリカバリー IO を高速化する

メンテナンス期間中に、クライアント IO またはリカバリー IO のいずれかを優先できます。クライアント IO よりもリカバリー IO を優先すると、OSD のリカバリー時間が大幅に短縮されます。有効なリカバリープロファイルとして、**balanced**、**high_client_ops**、および **high_recovery_ops** を選択できます。以下の手順に従ってリカバリープロファイルを設定してください。

前提条件

- カスタマーポータル から **odf-cli** ツールをダウンロードします。

手順

1. 現在のリカバリープロファイルを確認します。

```
$ odf get recovery-profile
```

2. リカバリープロファイルを変更します。

```
$ odf set recovery-profile <option>
```

option は、**balanced**、**high_client_ops**、または **high_recovery_ops** のいずれかに置き換えます。

3. 更新されたリカバリープロファイルを確認します。

```
$ odf get recovery-profile
```