Red Hat OpenShift Data Foundation 4.12

Google Cloud を使用した OpenShift Data Foundation のデプロイおよび管理

既存の Red Hat OpenShift Container Platform Google Cloud クラスターへの OpenShift Data Foundation のデプロイおよび管理手順

Last Updated: 2023-12-21

Red Hat OpenShift Data Foundation 4.12 Google Cloud を使用した OpenShift Data Foundation のデプロイおよび管理

既存の Red Hat OpenShift Container Platform Google Cloud クラスターへの OpenShift Data Foundation のデプロイおよび管理手順

法律上の通知

概要

Google Cloud 上の Red Hat OpenShift Container Platform を使用して Red Hat OpenShift Data Foundation をインストールおよび管理する方法について、このドキュメントをお読みください。
Deploying and managing OpenShift Data Foundation on Google Cloud is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

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、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。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)

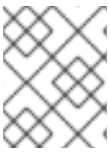
Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があれば、ぜひお知らせください。

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追加してください。
4. **Submit Bug** をクリックします。

はじめに

Red Hat OpenShift Data Foundation では、既存の Red Hat OpenShift Container Platform (RHOCP) Google Cloud クラスターでのデプロイメントをサポートします。



注記

Google Cloud では、内部の OpenShift Data Foundation クラスターのみがサポートされます。デプロイメント要件の詳細は、Planning your deployment を参照してください。

内部モードで OpenShift Data Foundation をデプロイするには、OpenShift Data Foundation のデプロイの準備についての章の要件を確認し、要件に応じて適切なデプロイメントプロセスを実行します。

- GoogleCloud への OpenShift Data Foundation のデプロイ
- スタンドアロンの Multicloud Object Gateway コンポーネントのデプロイ

第1章 OPENSHIFT DATA FOUNDATION のデプロイの準備

動的ストレージデバイスを使用して OpenShift Data Foundation を OpenShift Container Platform にデプロイすると、内部クラスターリソースを作成するオプションが提供されます。これにより、ベースサービスの内部プロビジョニングが可能になり、追加のストレージクラスをアプリケーションで使用できるようになります。

OpenShift Data Foundation のデプロイを開始する前に、以下を実行します。

1. オプション: 外部の鍵管理システム (KMS) HashiCorp Vault を使用してクラスター全体の暗号化を有効にする場合は、次の手順に従います。

   - 有効な Red Hat OpenShift Data Foundation Advanced サブスクリプションがあることを確認してください。OpenShift Data Foundation のサブスクリプションの仕組みを確認するには、OpenShift Data Foundation subscriptions に関するナレッジベースの記事 を参照してください。
   - 暗号化にトークン認証方法が選択されている場合は、Enabling cluster-wide encryption with the Token authentication using KMS を参照してください。
   - 暗号化に Kubernetes 認証方式が選択されている場合は、KMS を使用した Kubernetes 認証によるクラスター全体の暗号化の有効化 を参照してください。
   - 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。

2. オプション: 外部の鍵管理システム (KMS) Thales CipherTrust Manager を使用してクラスター全体の暗号化を有効にする場合は、まず Key Management Interoperability Protocol (KMIP) を有効にして、サーバーで署名付き証明書を使用する必要があります。以下の手順に従ってください。

   a. 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。

      i. 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。

   b. **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。

   c. 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。

   d. **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。

   e. 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。

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

      ii. 空いている **Port** を選択します。

      iii. **Network Interface** として **all** を選択します。

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

v. (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することができます。これはデフォルトでは無効になります。

vi. 使用する CA を選択し、**Save** をクリックします。

f. 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。

g. オプション: デプロイ時に StorageClass 暗号化を有効にする場合は、キー暗号化キー (KEK) として機能するキーを作成します。

i. **Keys → Add Key** に移動します。

ii. **Key Name** を入力します。

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

iv. **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。

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

vi. 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。

3. ノードの最小要件
OpenShift Data Foundation クラスターは、標準のデプロイメントリソース要件を満たしていない場合に、最小の設定でデプロイされます。プランニングガイドの リソース要件 セクションを参照してください。

4. 障害復旧の要件 テクノロジープレビュー
Red Hat OpenShift Data Foundation でサポートされる障害復旧機能では、障害復旧ソリューションを正常に実装するために以下の前提条件をすべて満たす必要があります。

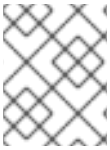
- 有効な Red Hat OpenShift Data Foundation Advanced サブスクリプション
- 有効な Red Hat Advanced Cluster Management for Kubernetes サブスクリプション
OpenShift Data Foundation のサブスクリプションの仕組みを確認するには、OpenShift Data Foundation subscriptions に関するナレッジベースの記事 を参照してください。

詳細な要件については、OpenShift ワークロード用の OpenShift Data Foundation Disaster Recovery の設定 ガイド、および Red Hat Advanced Cluster Management for Kubernetes ドキュメントの インストールガイド の **要件と推奨事項** のセクションを参照してください。

第2章 GOOGLECLOUD への OPENSHIFT DATA FOUNDATION のデプロイ

Google Cloud のインストーラーでプロビジョニングされるインフラストラクチャーによって提供される動的ストレージデバイスを使用して、OpenShift Data Foundation を OpenShift Container Platform にデプロイできます。これにより、内部クラスターリソースを作成でき、ベースサービスの内部プロビジョニングが可能になり、追加のストレージクラスをアプリケーションで使用可能にすることができます。

また、OpenShift Data Foundation で Multicloud Object Gateway (MCG) コンポーネントのみをデプロイすることもできます。詳細は、スタンドアロンの Multicloud Object Gateway のデプロイ を参照してください。



注記

Google Cloud では、内部の OpenShift Data Foundation クラスターのみがサポートされます。デプロイメント要件の詳細は、Planning your deployment を参照してください。

OpenShift Data Foundation のデプロイの準備 の章にある要件に対応していることを確認してから、動的ストレージデバイスを使用したデプロイに関する以下の手順を実行してください。

1. Red Hat OpenShift Data Foundation Operator のインストール
2. OpenShift Data Foundation クラスターの作成

2.1. RED HAT OPENSHIFT DATA FOUNDATION OPERATOR のインストール

Red Hat OpenShift Data Foundation Operator は、Red Hat OpenShift Container Platform Operator Hub を使用してインストールできます。

前提条件

- **cluster-admin** および operator インストールのパーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスできる。
- Red Hat OpenShift Container Platform クラスターにワーカーノードが少なくとも 3 つある。各ノードには 1 つのディスクが含まれ、3 つのディスク(PV)が必要です。ただし、1 つの PV はデフォルトで最終的に使用されません。これは想定される動作です。
- その他のリソース要件については、デプロイメントのプランニング ガイドを参照してください。

重要

- OpenShift Data Foundation のクラスター全体でのデフォルトノードセレクターを上書きする必要がある場合は、以下のコマンドを使用し、**openshift-storage** namespace の空のノードセレクターを指定できます (この場合、**openshift-storage** を作成します)。

```
$ oc annotate namespace openshift-storage openshift.io/node-selector=
```

- ノードに Red Hat OpenShift Data Foundation リソースのみがスケジュールされるように **infra** のテイントを設定します。これにより、サブスクリプションコストを節約できます。詳細は、ストレージリソースの管理および割り当て ガイドの **Red Hat OpenShift Data Foundation に専用のワーカーノードを使用する方法** を参照してください。

手順

1. OpenShift Web コンソールにログインします。
2. **Operators → OperatorHub** をクリックします。
3. スクロールするか、**OpenShift Data Foundation** を **Filter by keyword** ボックスに入力し、**OpenShift Data Foundation** Operator を検索します。
4. **Install** をクリックします。
5. **Install Operator** ページで、以下のオプションを設定します。
   a. Channel を **stable-4.12** として更新します。
   b. Installation Mode オプションに **A specific namespace on the cluster**を選択します。
   c. Installed Namespace に **Operator recommended namespace openshift-storage** を選択します。namespace **openshift-storage** が存在しない場合、これは Operator のインストール時に作成されます。
   d. 承認ストラテジー を **Automatic** または **Manual** として選択します。
   Automatic (自動) 更新を選択した場合、Operator Lifecycle Manager (OLM) は介入なしに、Operator の実行中のインスタンスを自動的にアップグレードします。

   Manual 更新を選択した場合、OLM は更新要求を作成します。クラスター管理者は、Operator を新しいバージョンに更新できるように更新要求を手動で承認する必要があります。
   e. **Console プラグイン** に **Enable** オプションが選択されていることを確認します。
   f. **Install** をクリックします。

検証手順

- Operator が正常にインストールされると、**Web console update is available** メッセージを含むポップアップがユーザーインターフェイスに表示されます。このポップアップから **Refresh web console** をクリックして、反映するコンソールを変更します。
- Web コンソールに移動します。
  - Installed Operators に移動し、**OpenShift Data Foundation** Operator に、インストールが

正常に実行された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。

- **Storage** に移動し、**Data Foundation** ダッシュボードが使用可能かどうかを確認します。

2.2. トークン認証方法を使用した KMS を使用したクラスター全体の暗号化の有効化

トークン認証のために、Vault でキーと値のバックエンドパスおよびポリシーを有効にできます。

前提条件

- Vault への管理者アクセス。
- 有効な Red Hat OpenShift Data Foundation Advanced サブスクリプション。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。
- 後で変更できないため、命名規則に従って一意のパス名をバックエンド **path** として慎重に選択してください。

手順

1. Vault で Key/Value (KV) バックエンドパスを有効にします。
Vault KV シークレットエンジン API の場合は、バージョン 1 です。

```
$ vault secrets enable -path=odf kv
```

Vault KV シークレットエンジン API の場合は、バージョン 2 を使用します。

```
$ vault secrets enable -path=odf kv-v2
```

2. シークレットに対して書き込み操作または削除操作を実行するようにユーザーを制限するポリシーを作成します。

```
echo '
path "odf/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}
path "sys/mounts" {
capabilities = ["read"]
}'| vault policy write odf -
```

3. 上記のポリシーに一致するトークンを作成します。

```
$ vault token create -policy=odf -format json
```

2.3. KUBERNETES 認証方式を使用した KMS でのクラスター全体の暗号化の有効化

キー管理システム (KMS) を使用して、クラスター全体の暗号化に対して Kubernetes 認証方式を有効にできます。

前提条件

- Vault への管理者アクセス。
- 有効な Red Hat OpenShift Data Foundation Advanced サブスクリプション。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。
- OpenShift Data Foundation Operator は Operator Hub からインストールしておく。
- バックエンド **path** として一意のパス名を選択する。これは命名規則に厳密に準拠する必要があります。このパス名は後で変更できません。

手順

1. サービスアカウントを作成します。

```
$ oc -n openshift-storage create serviceaccount <serviceaccount_name>
```

ここで、**<serviceaccount_name>** はサービスアカウントの名前を指定します。

以下に例を示します。

```
$ oc -n openshift-storage create serviceaccount odf-vault-auth
```

2. **clusterrolebindings** と **clusterroles** を作成します。

```
$ oc -n openshift-storage create clusterrolebinding vault-tokenreview-binding --
clusterrole=system:auth-delegator --serviceaccount=openshift-
storage:_<serviceaccount_name>_
```

以下に例を示します。

```
$ oc -n openshift-storage create clusterrolebinding vault-tokenreview-binding --
clusterrole=system:auth-delegator --serviceaccount=openshift-storage:odf-vault-auth
```

3. **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: odf-vault-auth-token
  namespace: openshift-storage
  annotations:
    kubernetes.io/service-account.name: <serviceaccount_name>
type: kubernetes.io/service-account-token
data: {}
EOF
```

ここで、**<serviceaccount_name>** は、前の手順で作成したサービスアカウントです。

4. 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 odf-vault-auth-token -o jsonpath="
```

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

5. OCP 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}")
```

6. サービスアカウントの発行者を取得します。

```
$ oc proxy &
$ proxy_pid=$!
$ issuer="$( curl --silent http://127.0.0.1:8001/.well-known/openid-configuration | jq -r
.issuer)"
$ kill $proxy_pid
```

7. 前の手順で収集した情報を使用して、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="$SA_CA_CRT" \
        issuer="$issuer"
```

重要

発行者が空の場合は Vault で Kubernetes 認証方法を設定します。

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

8. Vault で Key/Value (KV) バックエンドパスを有効にします。
Vault KV シークレットエンジン API の場合は、バージョン 1 を使用します。

```
$ vault secrets enable -path=odf kv
```

Vault KV シークレットエンジン API の場合は、バージョン 2 を使用します。

```
$ vault secrets enable -path=odf kv-v2
```

9. シークレットに対して **write** または **delete** 操作を実行するようにユーザーを制限するポリシーを作成します。

```
echo '
path "odf/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}
```

```
path "sys/mounts" {
capabilities = ["read"]
}'| vault policy write odf -
```

10. ロールを作成します。

```
$ vault write auth/kubernetes/role/odf-rook-ceph-op \
      bound_service_account_names=rook-ceph-system,rook-ceph-osd,noobaa \
      bound_service_account_namespaces=openshift-storage \
      policies=odf \
      ttl=1440h
```

ロール **odf-rook-ceph-op** は、後でストレージシステムの作成中に KMS 接続の詳細を設定するときに使用されます。

```
$ vault write auth/kubernetes/role/odf-rook-ceph-osd \
      bound_service_account_names=rook-ceph-osd \
      bound_service_account_namespaces=openshift-storage \
      policies=odf \
      ttl=1440h
```

2.4. OPENSHIFT DATA FOUNDATION クラスターの作成

OpenShift Data Foundation Operator のインストール後に OpenShift Data Foundation クラスターを作成します。

前提条件

- OpenShift Data Foundation Operator は Operator Hub からインストールしておく。詳細は、Installing OpenShift Data Foundation Operator を参照してください。
- Google Cloud プラットフォームのデフォルトのストレージクラスはハードディスクドライブ (HDD) を使用することに注意してください。パフォーマンスを向上させるためにソリッドステートドライブ (SSD) ベースのディスクを使用するには、以下の **ssd-storeageclass.yaml** の例に示されるように **pd-ssd** を使用してストレージクラスを作成する必要があります。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: faster
provisioner: kubernetes.io/gce-pd
parameters:
 type: pd-ssd
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
```

手順

1. 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。
   選択された **Project** が **openshift-storage** であることを確認します。
2. **OpenShift Data Foundation** Operator をクリックした後、**Create StorageSystem** をクリックします。

3. **Backing storage** ページで、以下を選択します。

   a. **Deployment type** オプションで **Full Deployment** を選択します。

   b. **Use an existing StorageClass** オプションを選択します。

   c. **Storage Class** を選択します。
   デフォルトでは、**standard** に設定されています。ただし、パフォーマンスを向上させるために SSD ベースのディスクを使用するようにストレージクラスを作成している場合は、ストレージクラスを選択する必要があります。

   d. **Next** をクリックします。

4. **Capacity and nodes** ページで、必要な情報を提供します。

   a. ドロップダウンリストから **Requested Capacity** の値を選択します。デフォルトで、これは **2 TiB** に設定されます。

   注記

   初期ストレージ容量を選択すると、クラスターの拡張は、選択された使用可能な容量を使用してのみ実行されます (raw ストレージの 3 倍)。

   b. **Select Nodes** セクションで、少なくとも 3 つの利用可能なノードを選択します。

   c. オプション: 選択したノードを OpenShift Data Foundation 専用にする場合は、**Taint nodes** チェックボックスを選択します。
   複数のアベイラビリティーゾーンを持つクラウドプラットフォームの場合は、ノードが異なる場所/アベイラビリティーゾーンに分散されていることを確認します。

   選択したノードが集約された 30 CPU および 72 GiB の RAM の OpenShift Data Foundation クラスターの要件と一致しない場合は、最小クラスターがデプロイされます。ノードの最小要件については、**プランニング**ガイドのリソース要件セクションを参照してください。

   d. **Next** をクリックします。

5. オプション: **Security and network** ページで、要件に応じて以下を設定します。

   a. 暗号化を有効にするには、**Enable data encryption for block and file storage**を選択します。

   b. 暗号化レベルのいずれかまたは両方を選択します。

   - **クラスター全体の暗号化**
     クラスター全体を暗号化します (ブロックおよびファイル)。

   - **StorageClass の暗号化**
     暗号化対応の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 (ブロックのみ) を作成します。

   c. オプション: **Connect to an external key management service**チェックボックスを選択します。これはクラスター全体の暗号化の場合はオプションになります。

      i. **Key Management Service Provider** ドロップダウンリストから、**Vault** または **Thales CipherTrust Manager (using KMIP)** を選択します。**Vault** を選択した場合は、次の手順に進みます。**Thales CipherTrust Manager (using KMIP)** を選択した場合は、手順 iii

に進みます。

ii. **認証方法**を選択します。

トークン認証方式の使用

- Vault ('https://<hostname or ip>') サーバーの一意の **Connection Name**、ホストの **Address**、**Port** 番号および **Token** を入力します。
- **Advanced Settings** を展開して、**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。
  - 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。
  - オプション: **TLS Server Name** および **Vault Enterprise Namespace** を入力します。
  - 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 証明書**、**クライアント証明書**、および **クライアントの秘密鍵** を指定します。
  - **Save** をクリックして、手順 iv に進みます。

Kubernetes 認証方式の使用

- Vault ('https://<hostname or ip>') サーバーの一意の **Connection Name**、ホストの **Address**、**Port** 番号および **Role** 名を入力します。
- **Advanced Settings** を展開して、**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。
  - 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。
  - 該当する場合は、**TLS Server Name** および **Authentication Path** を入力します。
  - 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 証明書**、**クライアント証明書**、および **クライアントの秘密鍵** を指定します。
  - **Save** をクリックして、手順 iv に進みます。

iii. **Thales CipherTrust Manager (using KMIP)** を KMS プロバイダーとして使用するには、次の手順に従います。

A. 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。

B. **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。以下に例を示します。

- **Address**: 123.34.3.2
- **Port**: 5696

C. **クライアント証明書、CA 証明書、**および **クライアント秘密鍵** をアップロードします。

D. StorageClass 暗号化が有効になっている場合は、上記で生成された暗号化および復号化に使用する一意の識別子を入力します。

E. **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。

iv. **Network** を選択します。

v. **Next** をクリックします。

6. **Review and create** ページで、設定の詳細を確認します。
設定設定を変更するには、**Back** をクリックします。

7. **Create StorageSystem** をクリックします。

検証手順

- インストールされたストレージクラスターの最終ステータスを確認するには、以下を実行します。

  a. OpenShift Web コンソールで、**Installed Operators → OpenShift Data Foundation → Storage System → ocs-storagecluster-storagesystem → Resources** の順に移動します。

  b. **StorageCluster** の **Status** が **Ready** になっており、それの横に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。

- OpenShift Data Foundation のすべてのコンポーネントが正常にインストールされていることを確認するには、Verifying your OpenShift Data Foundation deployment を参照してください。

関連情報

Overprovision Control アラートを有効にするには、モニタリングガイドの アラート を参照してください。

2.5. OPENSHIFT DATA FOUNDATION デプロイメントの確認

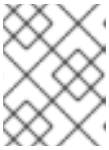
このセクションを使用して、OpenShift Data Foundation が正しくデプロイされていることを確認します。

2.5.1. Pod の状態の確認

手順

1. OpenShift Web コンソールから **Workloads → Pods** をクリックします。

2. **Project** ドロップダウンリストから **openshift-storage** を選択します。



注記

Show default projects オプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

コンポーネントごとに想定される Pod 数や、ノード数に合わせてこの数値がどのように変化するかなどの詳細は、表2.1「OpenShift Data Foundation クラスターに対応する Pod」 を参照してください。

3. 実行中および完了した Pod のフィルターを設定して、次の Pod が **Running** および **Completed** 状態であることを確認します。

表2.1 OpenShift Data Foundation クラスターに対応する Pod

コンポーネント	対応する Pod
OpenShift Data Foundation Operator	• **ocs-operator-*** (任意のストレージノードに 1 Pod) • **ocs-metrics-exporter-*** (任意のストレージノードに 1 Pod) • **odf-operator-controller-manager-*** (任意のストレージノードに 1 Pod) • **odf-console-*** (任意のストレージノードに 1 Pod) • **csi-addons-controller-manager-*** (任意のストレージノードに 1 Pod)
Rook-ceph Operator	**rook-ceph-operator-*** (任意のストレージノードに 1 Pod)
Multicloud Object Gateway	• **noobaa-operator-*** (任意のストレージノードに 1 Pod) • **noobaa-core-*** (任意のストレージノードに 1 Pod) • **noobaa-db-pg-*** (任意のストレージノードに 1 Pod) • **noobaa-endpoint-*** (任意のストレージノードに 1 Pod)
MON	**rook-ceph-mon-*** (ストレージノードに分散する 3 Pod)
MGR	**rook-ceph-mgr-*** (任意のストレージノードに 1 Pod)
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** (ストレージノードに分散する 2 Pod)

コンポーネント	対応する Pod

CSI	• **cephfs**
	  ◦ **csi-cephfsplugin-*** (各ストレージノードに 1 Pod)
	  ◦ **csi-cephfsplugin-provisioner-*** (ストレージノードに分散する 2 Pod)
	• **rbd**
	  ◦ **csi-rbdplugin-*** (各ストレージノードに 1 Pod)
	  ◦ **csi-rbdplugin-provisioner-*** (ストレージノードに分散する 2 Pod)

rook-ceph-crashcollector	**rook-ceph-crashcollector-*** (各ストレージノードに 1 Pod)
OSD	• **rook-ceph-osd-*** (各デバイス用に 1 Pod) • **rook-ceph-osd-prepare-ocs-deviceset-*** (各デバイス用に 1 Pod)

2.5.2. OpenShift Data Foundation クラスターの正常性の確認

手順

1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Overview** タブの **Status** カードで **Storage System** をクリックし、表示されたポップアップからストレージシステムリンクをクリックします。
3. **Block and File** タブの **Status** カードで、**Storage Cluster** に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。
4. **Details** カードで、クラスター情報が表示されていることを確認します。

ブロックおよびファイルダッシュボードを使用した OpenShift Data Foundation クラスターの正常性については、Monitoring OpenShift Data Foundation を参照してください。

2.5.3. Multicloud Object Gateway が正常であることの確認

手順

1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Overview** タブの **Status** カードで **Storage System** をクリックし、表示されたポップアップからストレージシステムリンクをクリックします。
   a. **Object** タブの **Status** card で、**Object Service** と **Data Resiliency** の両方に緑色のチェッ

a. **Object** タブの **Status card** で、**Object Service** と **Data Resiliency** の両方に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。

b. **Details** カードで、MCG 情報が表示されることを確認します。

ブロックおよびファイルダッシュボードを使用した OpenShift Data Foundation クラスターの正常性は、OpenShift Data Foundation の監視 を参照してください。

2.5.4. 特定のストレージクラスが存在することの確認

手順

1. OpenShift Web コンソールの左側のペインから **Storage → Storage Classes**をクリックします。
2. 以下のストレージクラスが OpenShift Data Foundation クラスターの作成時に作成されることを確認します。
   - **ocs-storagecluster-ceph-rbd**
   - **ocs-storagecluster-cephfs**
   - **openshift-storage.noobaa.io**

第3章 スタンドアロンの MULTICLOUD OBJECT GATEWAY のデプロイ

OpenShift Data Foundation で Multicloud Object Gateway コンポーネントのみをデプロイすると、デプロイメントで柔軟性が高まり、リソース消費を減らすことができます。このセクションでは、以下のステップで、スタンドアロンの Multicloud Object Gateway コンポーネントのみをデプロイします。

- Red Hat OpenShift Data Foundation Operator のインストール
- スタンドアロンの Multicloud Object Gateway の作成

3.1. RED HAT OPENSHIFT DATA FOUNDATION OPERATOR のインストール

Red Hat OpenShift Data Foundation Operator は、Red Hat OpenShift Container Platform Operator Hub を使用してインストールできます。

前提条件

- **cluster-admin** および operator インストールのパーミッションを持つアカウントを使用して OpenShift Container Platform クラスターにアクセスできる。
- Red Hat OpenShift Container Platform クラスターにワーカーノードが少なくとも 3 つある。各ノードには 1 つのディスクが含まれ、3 つのディスク(PV)が必要です。ただし、1 つの PV はデフォルトで最終的に使用されません。これは想定される動作です。
- その他のリソース要件については、デプロイメントのプランニング ガイドを参照してください。

重要

- OpenShift Data Foundation のクラスター全体でのデフォルトノードセレクターを上書きする必要がある場合は、以下のコマンドを使用し、**openshift-storage** namespace の空のノードセレクターを指定できます (この場合、**openshift-storage** を作成します)。

```
$ oc annotate namespace openshift-storage openshift.io/node-selector=
```

- ノードに Red Hat OpenShift Data Foundation リソースのみがスケジュールされるように **infra** のテイントを設定します。これにより、サブスクリプションコストを節約できます。詳細は、ストレージリソースの管理および割り当て ガイドの **Red Hat OpenShift Data Foundation に専用のワーカーノードを使用する方法** を参照してください。

手順

1. OpenShift Web コンソールにログインします。
2. **Operators → OperatorHub** をクリックします。
3. スクロールするか、**OpenShift Data Foundation** を **Filter by keyword** ボックスに入力し、**OpenShift Data Foundation** Operator を検索します。
4. **Install** をクリックします。

5. **Install Operator** ページで、以下のオプションを設定します。

    a. Channel を **stable-4.12** として更新します。

    b. Installation Mode オプションに **A specific namespace on the cluster**を選択します。

    c. Installed Namespace に **Operator recommended namespace openshift-storage** を選択します。namespace **openshift-storage** が存在しない場合、これは Operator のインストール時に作成されます。

    d. 承認ストラテジー を **Automatic** または **Manual** として選択します。
    Automatic (自動) 更新を選択した場合、Operator Lifecycle Manager (OLM) は介入なしに、Operator の実行中のインスタンスを自動的にアップグレードします。

    Manual 更新を選択した場合、OLM は更新要求を作成します。クラスター管理者は、Operator を新しいバージョンに更新できるように更新要求を手動で承認する必要があります。

    e. **Console プラグイン** に **Enable** オプションが選択されていることを確認します。

    f. **Install** をクリックします。

検証手順

- Operator が正常にインストールされると、**Web console update is available** メッセージを含むポップアップがユーザーインターフェイスに表示されます。このポップアップから **Refresh web console** をクリックして、反映するコンソールを変更します。
- Web コンソールに移動します。
    - Installed Operators に移動し、**OpenShift Data Foundation** Operator に、インストールが正常に実行された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。
    - **Storage** に移動し、**Data Foundation** ダッシュボードが使用可能かどうかを確認します。

3.2. スタンドアロンの MULTICLOUD OBJECT GATEWAY の作成

OpenShift Data Foundation のデプロイ中には、スタンドアロンの Multicloud Object Gateway コンポーネントのみを作成できます。

前提条件

- OpenShift Data Foundation Operator がインストールされている。

手順

1. 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。
    選択された **Project** が **openshift-storage** であることを確認します。

2. **OpenShift Data Foundation** Operator をクリックした後、**Create StorageSystem** をクリックします。

3. **Backing storage** ページで、以下を選択します。

    a. **Deployment type** の **Multicloud Object Gateway** を選択します。

b. **Use an existing StorageClass** オプションを選択します。

c. **Next** をクリックします。

4. オプション: **Connect to an external key management service**チェックボックスを選択します。これはクラスター全体の暗号化の場合はオプションになります。

a. **Key Management Service Provider**ドロップダウンリストから、**Vault** または **Thales CipherTrust Manager (using KMIP)** を選択します。**Vault** を選択した場合は、次の手順に進みます。**Thales CipherTrust Manager (using KMIP)** を選択した場合は、手順 iii に進みます。

b. **認証方法**を選択します。

トークン認証方式の使用

- Vault ('https://<hostname or ip>') サーバーの一意の **Connection Name**、ホストの **Address**、**Port** 番号および **Token** を入力します。
- **Advanced Settings** を展開して、**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。
  - 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。
  - オプション: **TLS Server Name** および **Vault Enterprise Namespace** を入力します。
  - 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 証明書**、**クライアント証明書**、および **クライアントの秘密鍵** を指定します。
  - **Save** をクリックして、手順 iv に進みます。

Kubernetes 認証方式の使用

- Vault ('https://<hostname or ip>') サーバーの一意の **Connection Name**、ホストの **Address**、**Port** 番号および **Role** 名を入力します。
- **Advanced Settings** を展開して、**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。
  - 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。
  - 該当する場合は、**TLS Server Name** および **Authentication Path** を入力します。
  - 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 証明書**、**クライアント証明書**、および **クライアントの秘密鍵** を指定します。
  - **Save** をクリックして、手順 iv に進みます。

c. **Thales CipherTrust Manager (using KMIP)** を KMS プロバイダーとして使用するには、次の手順に従います。

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. **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。以下に例を示します。

- **Address**: 123.34.3.2
- **Port**: 5696

iii. **クライアント証明書**、**CA 証明書**、および **クライアント秘密鍵** をアップロードします。

iv. StorageClass 暗号化が有効になっている場合は、上記で生成された暗号化および復号化に使用する一意の識別子を入力します。

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。

d. **Network** を選択します。

e. **Next** をクリックします。

5. **Review and create** ページで、設定の詳細を確認します。
設定設定を変更するには、**Back** をクリックします。

6. **Create StorageSystem** 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ることの確認

1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。

2. **Overview** タブの **Status** カードで **Storage System** をクリックし、表示されたポップアップからストレージシステムリンクをクリックします。

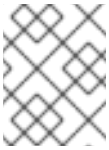
a. **Object** タブの **Status card** で、**Object Service** と **Data Resiliency** の両方に緑色のチェックマークが表示されていることを確認します。

b. **Details** カードで、MCG 情報が表示されることを確認します。

Pod の状態の確認

1. OpenShift Web コンソールから **Workloads → Pods** をクリックします。

2. **Project** ドロップダウンリストから **openshift-storage** を選択し、以下の Pod が **Running** 状態にあることを確認します。



注記

Show default projects オプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

コンポーネント	対応する Pod
OpenShift Data Foundation Operator	• **ocs-operator-*** (任意のストレージノードに 1 Pod) • **ocs-metrics-exporter-*** (任意のストレージノードに 1 Pod) • **odf-operator-controller-manager-*** (任意のストレージノードに 1 Pod) • **odf-console-*** (任意のストレージノードに 1 Pod) • **csi-addons-controller-manager-*** (任意のストレージノードに 1 Pod)
Rook-ceph Operator	**rook-ceph-operator-*** (任意のストレージノードに 1 Pod)
Multicloud Object Gateway	• **noobaa-operator-*** (任意のストレージノードに 1 Pod) • **noobaa-core-*** (任意のストレージノードに 1 Pod) • **noobaa-db-pg-*** (任意のストレージノードに 1 Pod) • **noobaa-endpoint-*** (任意のストレージノードに 1 Pod)

第4章 OPENSHIFT DATA FOUNDATION のアンインストール

4.1. 内部モードでの OPENSHIFT DATA FOUNDATION のアンインストール

OpenShift Data Foundation を内部モードでアンインストールするには、Uninstalling OpenShift Data Foundation のナレッジベース記事を参照してください。

第5章 ストレージクラスおよびストレージプール

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、ストレージクラスの異なる動作が必要な場合は、カスタムストレージクラスを作成できます。

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。

注記

複数のストレージクラスおよび複数のプールは、**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。

注記

単一デバイスセットの最小クラスターで新規作成できるストレージクラスは、2 つだけです。ストレージクラスターを拡張するたびに、新規ストレージクラスを 2 つ追加できます。

5.1. 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に使用されるプラグインである **RBD Provisioner** を選択します。

7. 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。

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。

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

9. **Create** をクリックしてストレージクラスを作成します。

5.2. 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** を使用したアクセス権の設定 の説明に従って、アクセスを確実に設定してください。
- **vaulttenantsa** の使用 (テクノロジープレビュー): **vaulttenantsa** を使用したアクセス権の設定 の説明に従って、アクセスを確実に設定してください。
- 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。

手順

1. OpenShift Web コンソールで、**Storage → Storage Classes** 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 について 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. Immediate または WaitForFirstConsumer を Volume binding モードとして選択します。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ため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。KMS 接続の詳細を設定するオプションは 2 つあります。

   - **既存の KMS 接続の選択**: ドロップダウンリストから既存の KMS 接続を選択します。この一覧は、**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. Key Management Service Provider を選択します。

     b. Key Management Service Provider として **Vault** が選択されている場合は、次の手順に従います。

        i. Vault サーバーの一意の **接続名**、ホスト **アドレス**、ポート番号および **トークン** を入力します。

        ii. Advanced Settings を展開して、**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 証明書**、**クライアント証明書**、および **クライアントの秘密鍵** を指定します。

            D. Save をクリックします。

     c. Key Management Service Provider として **Thales CipherTrust Manager (using KMIP)** が選択されている場合は、次の手順に従います。

        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. **クライアント証明書**、**CA 証明書**、および **クライアント秘密鍵** をアップロードします。

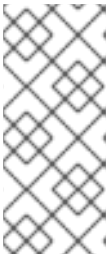
iv. 上記で生成された暗号化および復号化に使用する鍵の **Unique Identifier** を入力します。

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。

d. **Save** をクリックします。

e. **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。一致しない場合には、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。

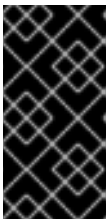
```
kind: ConfigMap
apiVersion: v1
metadata:
```

```
  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、本書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。

第6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、イメージレジストリー、モニタリング、およびロギングなどの OpenShift Container Platform サービスのストレージを提供できます。

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

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、クラスターは動作しなくなり、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 ログ保持時間の設定、および 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。

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。

6.1. 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。Google Cloud 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。

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、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がある。

- 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 Persistent Volume Claim(永続ボリューム要求、PVC) 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一覧から、プロビジョナー **openshift-storage.cephfs.csi.ceph.com** で **Storage Class** を指定します。
      ii. Persistent Volume Claim(永続ボリューム要求、PVC) 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 Persistent Volume Claim(永続ボリューム要求、PVC) のステータスが **Bound** として一覧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 Persistent Volume Claim(永続ボリューム要求、PVC) 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 Persistent Volume Claim(永続ボリューム要求、PVC) 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。

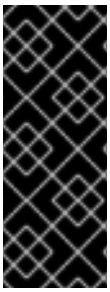
d. 新規の **image-registry-*** Pod をクリックし、Pod の詳細を表示します。

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

6.2. 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 および Alert Manager で設定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 Monitoring guide の Modifying retention time for Prometheus metrics data 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がある。
- 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, e.g. 24h>
      volumeClaimTemplate:
        metadata:
          name: ocs-prometheus-claim
        spec:
          storageClassName: ocs-storagecluster-ceph-rbd
          resources:
            requests:
              storage: <size of claim, e.g. 40Gi>
    alertmanagerMain:
      volumeClaimTemplate:
        metadata:
          name: ocs-alertmanager-claim
        spec:
          storageClassName: ocs-storagecluster-ceph-rbd
          resources:
            requests:
              storage: <size of claim, e.g. 40Gi>
```

5. **Create** をクリックして、設定マップを保存し、作成します。

検証手順

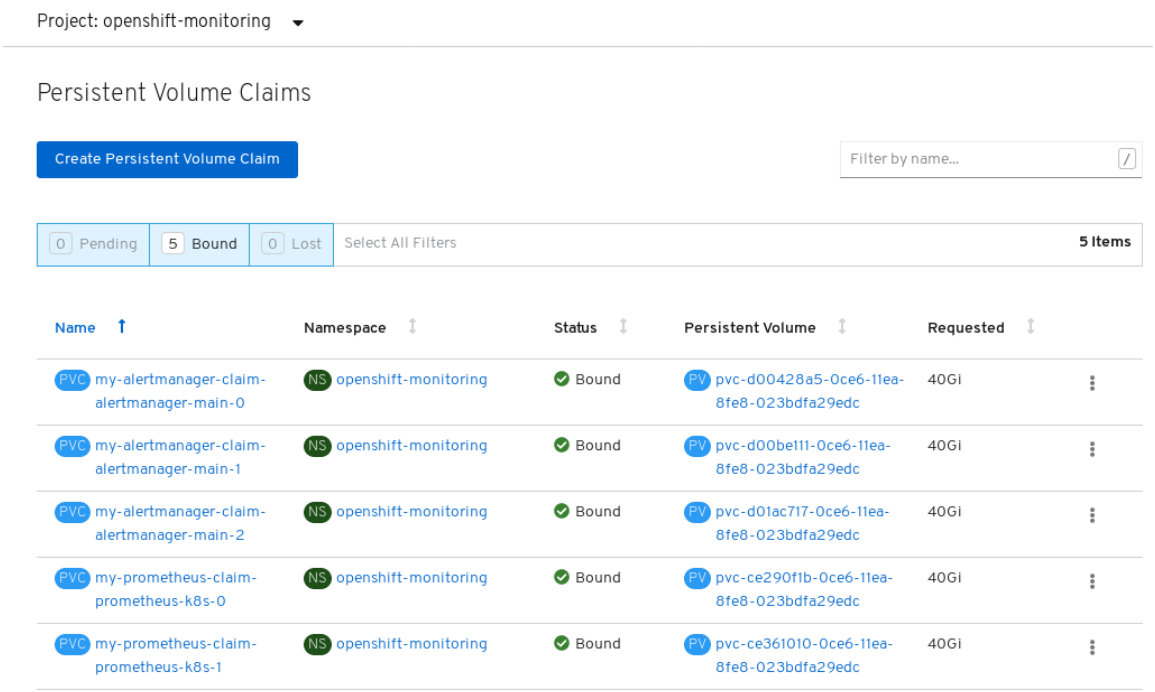
1. Persistent Volume Claim (永続ボリューム要求、PVC) が Pod にバインドされていることを確認します。

   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 Persistent Volume Claim(永続ボリューム要求、PVC) 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6.1 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



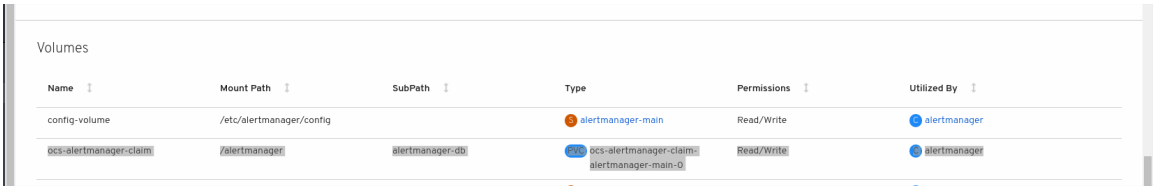
2. 新規の **alertmanager-main-*** Pod が **Running** 状態で表示されることを確認します。

   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 Persistent Volume Claim(永続ボリューム要求、PVC) 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6.2 alertmanager-main-* Pod に割り当てられた Persistent Volume Claim(永続ボリューム要求、PVC)



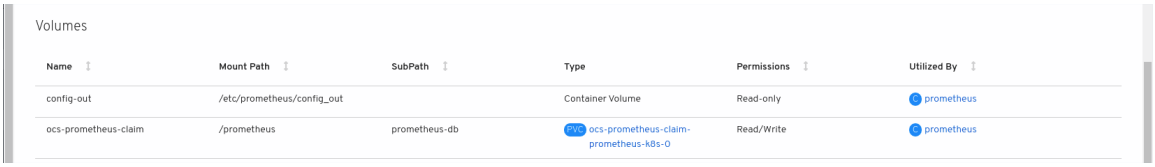
3. 新規 **prometheus-k8s-*** Pod が **Running** 状態で表示されることを確認します。

   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 Persistent Volume Claim (永続ボ

b. **Volumes** までスクロールダウンし、ボリューム に新規の Persistent Volume Claim (永続ボリューム要求、PVC) 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6.3 prometheus-k8s-* Pod に割り当てられた Persistent Volume Claim(永続ボリューム要求、PVC)



6.3. 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 サービスについての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 ドキュメントで ログ保持時間の設定 について参照してください。

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。

6.3.1. 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ついて Persistent Volume Claim (永続ボリューム要求、PVC) 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 Persistent Volume Claim(永続ボリューム要求、PVC) 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、ログストアのレプリケーションポリシーの設定 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、ログストアの永続ストレージの設定 および emptyDir ストレージのログストアの設定 を参照してください。

6.3.2. 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がある。
- 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、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

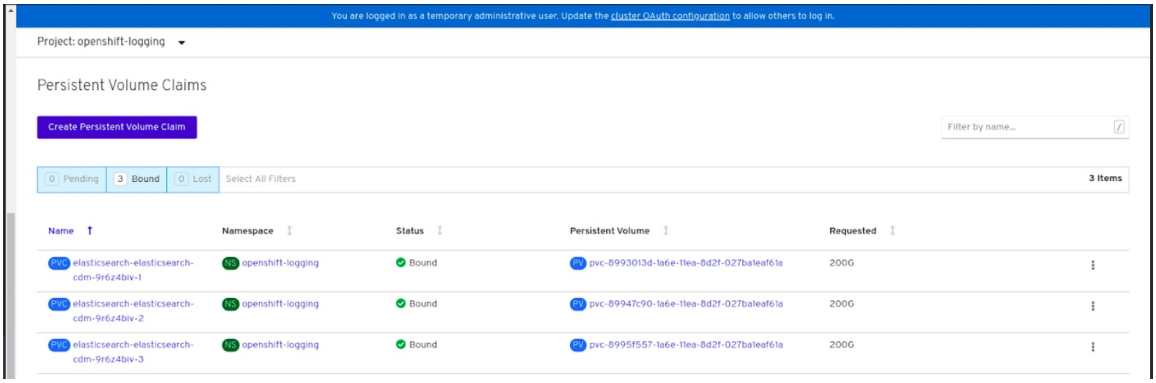
1. Persistent Volume Claim(永続ボリューム要求、PVC) が **elasticsearch** Pod にバインドされていることを確認します。

   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. Persistent Volume Claim(永続ボリューム要求、PVC) 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6.4 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。

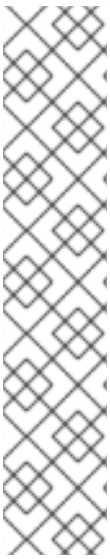
b. プロジェクトを **openshift-logging** に設定します。

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 Persistent Volume Claim (永続ボリューム要求、PVC) に一致する **Type** があることを確認します (例: **elasticsearch-elasticsearch-cdm-9r624biv-3**)。

f. Persistent Volume Claim(永続ボリューム要求、PVC) 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、ログ保持時間の設定 を参照してください。

注記

Persistent Volume Claim(永続ボリューム要求、PVC) 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ついての章に記載されている、クラスターロギング Operator の OpenShift Data Foundation からの削除についての手順を使用します。

第7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. Persistent Volume Claim (永続ボリューム要求、PVC) の名前を指定します。
  c. ReadWriteOnce (RWO) または ReadWriteMany (RWX) アクセスモードを選択します。

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、Persistent Volume Claim (永続ボリューム要求、PVC) 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 Persistent Volume Claim(永続ボリューム要求、PVC) に一致する **Type** があることを確認します。
5. Persistent Volume Claim(永続ボリューム要求、PVC) 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** effect 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

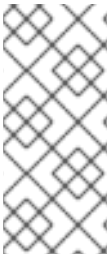
ノードにストレージテイントを追加するには、**openshift-dns daemonset** などの他の **daemonset** Pod の容認処理が必要になる場合があります。容認を管理する方法の詳細は、ナレッジベースの記事 https://access.redhat.com/solutions/6592171 を参照してください。

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には、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。ラベルをマシン API によって作成されるノードに手動で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/ノードが失敗する場合、置き換え用の 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について説明します。

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

関連情報

詳細については、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 ストレージノードのスケーリング

OpenShift Data Foundation のストレージ容量をスケーリングするには、以下のいずれかを実行できます。

- **ストレージノードのスケールアップ**: 既存の OpenShift Data Foundation ワーカーノードに対してストレージ容量を追加します。
- **ストレージノードのスケールアウト**: ストレージ容量を含む新規ワーカーノードを追加します。

9.1. ストレージノードのスケーリングの要件

ストレージノードをスケーリングする前に、以下のセクションを参照して、特定の Red Hat OpenShift Data Foundation インスタンスのノード要件を把握してください。

- プラットフォーム要件
- ストレージデバイスの要件
  - 動的ストレージデバイス
  - 容量のプランニング

警告

常にストレージ容量が十分にあることを確認してください。

ストレージが完全に一杯になると、容量を追加したり、ストレージからコンテンツを削除したり、コンテンツを移動して領域を解放することはできません。完全なストレージを復元することは非常に困難です。

容量アラートは、クラスターストレージ容量が合計容量の 75% (ほぼ一杯) および 85% (一杯) になると発行されます。容量についての警告に常に迅速に対応し、ストレージを定期的に確認して、ストレージ領域が不足しないようにします。

ストレージ領域が不足する場合は、Red Hat カスタマーポータルにお問い合わせください。

9.2. GOOGLE CLOUD インフラストラクチャーの OPENSHIFT DATA FOUNDATION ノードへの容量の追加によるストレージのスケールアップ

ユーザーがプロビジョニングしたインフラストラクチャー上に動的に作成したストレージクラスターのストレージ容量を増やすために、設定済みの Red Hat OpenShift Data Foundation ワーカーノードにストレージ容量およびパフォーマンスを追加できます。

前提条件

- OpenShift Container Platform に対する管理者権限がある。
- 実行中の OpenShift Data Foundation ストレージクラスターがある。

- ディスクは、最初のデプロイメント時に使用したものと同じサイズおよびタイプである必要があります。

手順

1. OpenShift Web コンソールにログインします。
2. **Operators → Installed Operators** をクリックします。
3. **OpenShift Data Foundation** Operator をクリックします。
4. **Storage Systems** タブをクリックします。
   a. ストレージシステム名の右側にある **Action Menu (⋮)** をクリックし、オプションメニューを拡張します。
   b. オプションメニューから **Add Capacity** を選択します。
   c. **Storage Class** を選択します。新しいストレージデバイスのプロビジョニングに使用するストレージクラスを選択します。
   HDD を使用するデフォルトのストレージクラスを使用している場合は、ストレージクラスを **standard** に設定します。ただし、パフォーマンスを向上させるために SSD ベースのディスクを使用するようにストレージクラスを作成している場合は、ストレージクラスを選択する必要があります。

   Raw Capacity フィールドには、ストレージクラスの作成時に設定されるサイズが表示されます。OpenShift Data Foundation はレプリカ数 3 を使用するため、消費されるストレージの合計量はこの量の 3 倍になります。
   d. **Add** をクリックします。
5. ステータスを確認するには、**Storage → Data Foundation** に移動し、Status カードの **Storage System** に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。

検証手順

- **Raw Capacity** カードを確認します。
  a. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
  b. **Overview** タブの **Status** カードで **Storage System** をクリックし、表示されたポップアップからストレージシステムリンクをクリックします。
  c. **Block and File** タブで、**Raw Capacity** カードを確認します。
  容量は選択に応じて増大することに注意してください。

  注記

  Raw 容量はレプリケーションを考慮せず、フル容量を表示します。

- 新しい OSD およびそれらの対応する新規 Persistent Volume Claims (PVC) が作成されていることを確認します。
  - 新規作成された OSD の状態を表示するには、以下を実行します。
    a. OpenShift Web コンソールから **Workloads → Pods** をクリックします。

b. **Project** ドロップダウンリストから **openshift-storage** を選択します。

注記

Show default projectsオプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

- Pod の状態を確認します。

  a. OpenShift Web コンソールで、**Storage → Persistent Volume Claims**をクリックします。

  b. **Project** ドロップダウンリストから **openshift-storage** を選択します。

注記

Show default projectsオプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

- (オプション) クラスターでクラスター全体の暗号化が有効な場合は、新規 OSD デバイスが暗号化されていることを確認します。

  a. 新規 OSD Pod が実行しているノードを特定します。

```
$ oc get -n openshift-storage -o=custom-columns=NODE:.spec.nodeName pod/<OSD-pod-name>
```

<OSD-pod-name>

これは OSD Pod の名前です。
以下に例を示します。

```
$ oc get -n openshift-storage -o=custom-columns=NODE:.spec.nodeName pod/rook-ceph-osd-0-544db49d7f-qrgqm
```

出力例:

```
NODE
compute-1
```

  b. 直前の手順で特定された各ノードに以下を実行します。

    i. デバッグ Pod を作成し、選択したホストの chroot 環境を開きます。

```
$ oc debug node/<node-name>
```

<node-name>

ノードの名前。

```
$ chroot /host
```

ii. **ocs-deviceset** 名の横にある **crypt** キーワードを確認します。

```
$ lsblk
```

重要

クラスターの削減は、Red Hat サポートチーム のサポートがある場合にのみサポートされます。

9.3. 新規ノードの追加によるストレージ容量のスケールアウト

ストレージ容量をスケールアウトするには、以下を実行する必要があります。

- 既存のワーカーノードがサポートされる最大 OSD (初期設定で選択される容量の 3 OSD の増分) で実行されている場合には、ストレージの容量を増やすために新規ノードを追加します。
- 新規ノードが正常に追加されたことを確認します。
- ノードが追加された後にストレージ容量をスケールアップします。

9.3.1. インストーラーでプロビジョニングされるインフラストラクチャーへのノードの追加

前提条件

- OpenShift Container Platform に対する管理者権限がある。
- 実行中の OpenShift Data Foundation ストレージクラスターがある。

手順

1. **Compute → Machine Sets** に移動します。
2. ノードを追加する必要のあるマシンセットで、**Edit Machine Count** を選択します。
   a. ノード数を追加し、**Save** をクリックします。
   b. **Compute → Nodes** をクリックし、新規ノードが **Ready** 状態にあることを確認します。
3. OpenShift Data Foundation ラベルを新規ノードに適用します。
   a. 新規ノードについて、**Action menu (⋮) → Edit Labels** をクリックします。
   b. **cluster.ocs.openshift.io/openshift-storage** を追加し、**Save** をクリックします。

注記

異なるゾーンのそれぞれに 3 つのノードを追加することが推奨されます。3 つのノードを追加して、それらすべてのノードに対してこの手順を実行する必要があります。

検証手順

1. 端末で次のコマンドを実行し、新しいノードが出力に存在することを確認します。

```
$ oc get nodes --show-labels | grep cluster.ocs.openshift.io/openshift-storage= |cut -d' ' -f1
```

2. OpenShift Web コンソールで、**Workloads → Pods** をクリックし、新しいノードの少なくとも以下の Pod が **Running** 状態になっていることを確認します。
   - **csi-cephfsplugin-***
   - **csi-rbdplugin-***

9.3.2. ストレージ容量のスケールアップ

OpenShift Data Foundation に新しいノードを追加した後、新しいノードの追加によるストレージ容量のスケールアウト の説明に従ってストレージ容量をスケールアップする必要があります。

第10章 MULTICLOUD OBJECT GATEWAY

10.1. MULTICLOUD OBJECT GATEWAY について

Multicloud Object Gateway (MCG) は OpenShift の軽量オブジェクトストレージサービスであり、ユーザーは必要に応じて、複数のクラスター、およびクラウドネイティブストレージを使用して、オンプレミスで小規模に開始し、その後にスケーリングできます。

10.2. アプリケーションの使用による MULTICLOUD OBJECT GATEWAY へのアクセス

AWS S3 を対象とするアプリケーションまたは AWS S3 Software Development Kit(SDK) を使用するコードを使用して、オブジェクトサービスにアクセスできます。アプリケーションは、Multicloud Object Gateway (MCG) エンドポイント、アクセスキー、およびシークレットアクセスキーを指定する必要があります。ターミナルまたは MCG CLI を使用して、この情報を取得できます。

前提条件

- 実行中の OpenShift Data Foundation Platform。
- MCG コマンドラインインターフェイスをダウンロードして、管理を容易にします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。

- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- または、MCG パッケージを、Download RedHat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

関連するエンドポイント、アクセスキー、およびシークレットアクセスキーには、以下の 2 つの方法でアクセスできます。

- 「ターミナルから Multicloud Object Gateway へのアクセス」
- 「MCG コマンドラインインターフェイスからの Multicloud Object Gateway へのアクセス」

以下に例を示します。

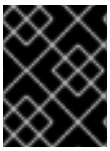
仮想ホストのスタイルを使用した MCG バケットへのアクセス

クライアントアプリケーションが https://**<bucket-name>**.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com にアクセスしようとする場合

<bucket-name>

MCG バケットの名前です。
たとえば、https://mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com になります。

DNS エントリーは、S3 サービスを参照するように、**mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com** が必要です。



重要

仮想ホストスタイルを使用してクライアントアプリケーションを MCG バケットを参照するように、DNS エントリーがあることを確認します。

10.2.1. ターミナルから Multicloud Object Gateway へのアクセス

手順

describe コマンドを実行し、アクセスキー (**AWS_ACCESS_KEY_ID** 値) およびシークレットアクセスキー (**AWS_SECRET_ACCESS_KEY** 値) を含む Multicloud Object Gateway (MCG) エンドポイントについての情報を表示します。

```
# oc describe noobaa -n openshift-storage
```

出力は以下のようになります。

```
Name:         noobaa
Namespace:    openshift-storage
Labels:       <none>
Annotations:  <none>
API Version:  noobaa.io/v1alpha1
Kind:         NooBaa
Metadata:
  Creation Timestamp:  2019-07-29T16:22:06Z
  Generation:          1
  Resource Version:    6718822
  Self Link:           /apis/noobaa.io/v1alpha1/namespaces/openshift-storage/noobaas/noobaa
  UID:                 019cfb4a-b21d-11e9-9a02-06c8de012f9e
Spec:
Status:
  Accounts:
    Admin:
      Secret Ref:
        Name:          noobaa-admin
        Namespace:     openshift-storage
  Actual Image:        noobaa/noobaa-core:4.0
  Observed Generation:  1
  Phase:               Ready
  Readme:
```

```
Welcome to NooBaa!
-----------------

Welcome to NooBaa!
-----------------
NooBaa Core Version:
NooBaa Operator Version:

Lets get started:

1. Connect to Management console:

   Read your mgmt console login information (email & password) from secret: "noobaa-admin".

     kubectl get secret noobaa-admin -n openshift-storage -o json | jq '.data|map_values(@base64d)'

   Open the management console service - take External IP/DNS or Node Port or use port forwarding:

     kubectl port-forward -n openshift-storage service/noobaa-mgmt 11443:443 &
     open https://localhost:11443

2. Test S3 client:

   kubectl port-forward -n openshift-storage service/s3 10443:443 &
(1)
   NOOBAA_ACCESS_KEY=$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r '.data.AWS_ACCESS_KEY_ID|@base64d')
(2)
   NOOBAA_SECRET_KEY=$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r '.data.AWS_SECRET_ACCESS_KEY|@base64d')
   alias s3='AWS_ACCESS_KEY_ID=$NOOBAA_ACCESS_KEY AWS_SECRET_ACCESS_KEY=$NOOBAA_SECRET_KEY aws --endpoint https://localhost:10443 --no-verify-ssl s3'
   s3 ls


Services:
  Service Mgmt:
    External DNS:
      https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
      https://a3406079515be11eaa3b70683061451e-1194613580.us-east-2.elb.amazonaws.com:443
    Internal DNS:
      https://noobaa-mgmt.openshift-storage.svc:443
    Internal IP:
      https://172.30.235.12:443
    Node Ports:
      https://10.0.142.103:31385
    Pod Ports:
      https://10.131.0.19:8443
  serviceS3:
    External DNS: (3)
      https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
      https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443
```

```
    Internal DNS:
      https://s3.openshift-storage.svc:443
    Internal IP:
      https://172.30.86.41:443
    Node Ports:
      https://10.0.142.103:31011
    Pod Ports:
      https://10.131.0.19:6443
```

1. アクセスキー (**AWS_ACCESS_KEY_ID** 値)
2. シークレットアクセスキー (**AWS_SECRET_ACCESS_KEY** 値)
3. MCG エンドポイント

注記

oc describe noobaa コマンドには、利用可能な内部および外部 DNS 名が一覧表示されます。内部 DNS を使用する場合、トラフィックは無料になります。外部 DNS はロードバランシングを使用してトラフィックを処理するため、1時間あたりのコストがかかります。

10.2.2. MCG コマンドラインインターフェイスからの Multicloud Object Gateway へのアクセス

前提条件

- MCG コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。

- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

手順

status コマンドを実行して、エンドポイント、アクセスキー、およびシークレットアクセスキーにアクセスします。

```
noobaa status -n openshift-storage
```

出力は以下のようになります。

```
INFO[0000] Namespace: openshift-storage
INFO[0000]
INFO[0000] CRD Status:
INFO[0003]   Exists: CustomResourceDefinition "noobaas.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "backingstores.noobaa.io"
INFO[0003]   Exists: CustomResourceDefinition "bucketclasses.noobaa.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbucketclaims.objectbucket.io"
INFO[0004]   Exists: CustomResourceDefinition "objectbuckets.objectbucket.io"
INFO[0004]
INFO[0004] Operator Status:
INFO[0004]   Exists: Namespace "openshift-storage"
INFO[0004]   Exists: ServiceAccount "noobaa"
INFO[0005]   Exists: Role "ocs-operator.v0.0.271-6g45f"
INFO[0005]   Exists: RoleBinding "ocs-operator.v0.0.271-6g45f-noobaa-f9vpj"
INFO[0006]   Exists: ClusterRole "ocs-operator.v0.0.271-fjhgh"
INFO[0006]   Exists: ClusterRoleBinding "ocs-operator.v0.0.271-fjhgh-noobaa-pdxn5"
INFO[0006]   Exists: Deployment "noobaa-operator"
INFO[0006]
INFO[0006] System Status:
INFO[0007]   Exists: NooBaa "noobaa"
INFO[0007]   Exists: StatefulSet "noobaa-core"
INFO[0007]   Exists: Service "noobaa-mgmt"
INFO[0008]   Exists: Service "s3"
INFO[0008]   Exists: Secret "noobaa-server"
INFO[0008]   Exists: Secret "noobaa-operator"
INFO[0008]   Exists: Secret "noobaa-admin"
INFO[0009]   Exists: StorageClass "openshift-storage.noobaa.io"
INFO[0009]   Exists: BucketClass "noobaa-default-bucket-class"
INFO[0009]   (Optional) Exists: BackingStore "noobaa-default-backing-store"
INFO[0010]   (Optional) Exists: CredentialsRequest "noobaa-cloud-creds"
INFO[0010]   (Optional) Exists: PrometheusRule "noobaa-prometheus-rules"
INFO[0010]   (Optional) Exists: ServiceMonitor "noobaa-service-monitor"
INFO[0011]   (Optional) Exists: Route "noobaa-mgmt"
INFO[0011]   (Optional) Exists: Route "s3"
INFO[0011]   Exists: PersistentVolumeClaim "db-noobaa-core-0"
INFO[0011]   System Phase is "Ready"
INFO[0011]   Exists:  "noobaa-admin"

#------------------#
#- Mgmt Addresses -#
#------------------#

ExternalDNS : [https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a3406079515be11eaa3b70683061451e-1194613580.us-east-2.elb.amazonaws.com:443]
ExternalIP  : []
NodePorts   : [https://10.0.142.103:31385]
InternalDNS : [https://noobaa-mgmt.openshift-storage.svc:443]
InternalIP  : [https://172.30.235.12:443]
PodPorts    : [https://10.131.0.19:8443]

#--------------------#
#- Mgmt Credentials -#
#--------------------#
```

```
email    : admin@noobaa.io
password : HKLbH1rSuVU0I/souIkSiA==

#----------------#
#- S3 Addresses -#
#----------------#

(1)
ExternalDNS : [https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443]
ExternalIP  : []
NodePorts   : [https://10.0.142.103:31011]
InternalDNS : [https://s3.openshift-storage.svc:443]
InternalIP  : [https://172.30.86.41:443]
PodPorts    : [https://10.131.0.19:6443]

#------------------#
#- S3 Credentials -#
#------------------#

(2)
AWS_ACCESS_KEY_ID     : jVmAsu9FsvRHYmfjTiHV
(3)
AWS_SECRET_ACCESS_KEY : E//420VNedJfATvVSmDz6FMtsSAzuBv6z180PT5c

#------------------#
#- Backing Stores -#
#------------------#

NAME                          TYPE     TARGET-BUCKET                                      PHASE   AGE
noobaa-default-backing-store   aws-s3   noobaa-backing-store-15dc896d-7fe0-4bed-9349-
5942211b93c9   Ready   141h35m32s

#------------------#
#- Bucket Classes -#
#------------------#

NAME                          PLACEMENT                                                  PHASE   AGE
noobaa-default-bucket-class   {Tiers:[{Placement: BackingStores:[noobaa-default-backing-store]}]}
Ready   141h35m33s

#----------------#
#- Bucket Claims -#
#----------------#

No OBC's found.
```

(1) endpoint

(2) アクセスキー

(3) シークレットアクセスキー

これで、アプリケーションに接続するための関連するエンドポイント、アクセスキー、およびシークレットアクセスキーを使用できます。

以下に例を示します。

AWS S3 CLI がアプリケーションである場合、以下のコマンドは OpenShift Data Foundation のバケットを一覧表示します。

```
AWS_ACCESS_KEY_ID=<AWS_ACCESS_KEY_ID>
AWS_SECRET_ACCESS_KEY=<AWS_SECRET_ACCESS_KEY>
aws --endpoint <ENDPOINT> --no-verify-ssl s3 ls
```

10.3. ハイブリッドまたはマルチクラウド用のストレージリソースの追加

10.3.1. 新規バッキングストアの作成

以下の手順を使用して、OpenShift Data Foundation で新規のバッキングストアを作成します。

前提条件

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

手順

1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Backing Store** タブをクリックします。
3. **Create Backing Store** をクリックします。
4. **Create New Backing Store** ページで、以下を実行します。
   a. **Backing Store Name** を入力します。
   b. **Provider** を選択します。
   c. **Region** を選択します。
   d. **Endpoint** を入力します。これは任意です。
   e. ドロップダウンリストから **Secret** を選択するか、独自のシークレットを作成します。オプションで、**Switch to Credentials** ビューを選択すると、必要なシークレットを入力できます。
   OCP シークレットの作成に関する詳細は、**Openshift Container Platform** ドキュメントの シークレットの作成 を参照してください。

   バッキングストアごとに異なるシークレットが必要です。特定のバッキングストアのシークレット作成についての詳細は 「MCG コマンドラインインターフェイスを使用したハイブリッドまたはマルチクラウドのストレージリソースの追加」 を参照して、YAML を使用したストレージリソースの追加についての手順を実行します。

注記

このメニューは、Google Cloud およびローカル PVC 以外のすべてのプロバイダーに関連します。

f. **Target bucket** を入力します。ターゲットバケットは、リモートクラウドサービスでホストされるコンテナーストレージです。MCG に対してシステム用にこのバケットを使用できることを通知する接続を作成できます。

5. **Create Backing Store** をクリックします。

検証手順

1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Backing Store** タブをクリックして、すべてのバッキングストアを表示します。

10.3.2. MCG コマンドラインインターフェイスを使用したハイブリッドまたはマルチクラウドのストレージリソースの追加

Multicloud Object Gateway (MCG) は、クラウドプロバイダーおよびクラスター全体にまたがるデータの処理を単純化します。

MCG で使用できるバッキングストレージを追加する必要があります。

デプロイメントのタイプに応じて、以下のいずれかの手順を選択してバッキングストレージを作成できます。

- AWS でサポートされるバッキングストアを作成する方法については、「AWS でサポートされるバッキングストアの作成」 を参照してください。
- IBM COS でサポートされるバッキングストアを作成する方法については、「IBM COS でサポートされるバッキングストアの作成」 を参照してください。
- Azure でサポートされるバッキングストアを作成する方法については、「Azure でサポートされるバッキングストアの作成」 を参照してください。
- GCP でサポートされるバッキングストアを作成する方法については、「GCP でサポートされるバッキングストアの作成」 を参照してください。
- ローカルの永続ボリュームでサポートされるバッキングストアを作成する方法については、「ローカル永続ボリュームでサポートされるバッキングストアの作成」 を参照してください。

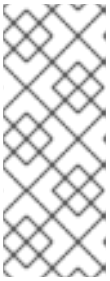
VMware デプロイメントの場合、「s3 と互換性のある Multicloud Object Gateway バッキングストアの作成」 に進み、詳細の手順を確認します。

10.3.2.1. AWS でサポートされるバッキングストアの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

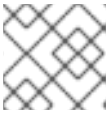


注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。たとえば、IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- または、MCG パッケージを、Download Red Hat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。



注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

MCG コマンドラインインターフェイスの使用

- MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

```
noobaa backingstore create aws-s3 <backingstore_name> --access-key=<AWS ACCESS KEY> --secret-key=<AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

<backingstore_name>

バッキングストアの名前。

<AWS ACCESS KEY> と <AWS SECRET ACCESS KEY>

この目的のために作成した AWS アクセスキー ID とシークレットアクセスキー。

<bucket-name>

既存の AWS バケット名。この引数は、バッキングストアのターゲットバケットとして使用するバケットを MCG に示し、その後、データストレージと管理を行います。
出力は次のようになります。

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "aws-resource"
INFO[0002]   Created: Secret "backing-store-secret-aws-resource"
```

YAML を使用してストレージリソースを追加する

1. 認証情報でシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
  namespace: openshift-storage
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

<AWS ACCESS KEY> および <AWS SECRET ACCESS KEY>

Base64 を使用して独自の AWS アクセスキー ID とシークレットアクセスキーを指定してエンコードし、その結果を **<AWS ACCESS KEY ID ENCODED IN BASE64>** と **<AWS SECRET ACCESS KEY ENCODED IN BASE64>** に使用します。

<backingstore-secret-name>

前のステップで作成された backingstore シークレットの名前。

2. 特定のバッキングストアについて以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
  type: aws-s3
```

<bucket-name>

既存の AWS バケット名。

<backingstore-secret-name>

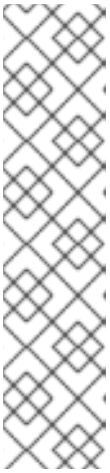
前のステップで作成された backingstore シークレットの名前。

10.3.2.2. IBM COS でサポートされるバッキングストアの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```



注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。以下に例を示します。

- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- または、MCG パッケージを、Download Red Hat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

コマンドラインインターフェイスの使用

1. MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

```
noobaa backingstore create ibm-cos <backingstore_name> --access-key=<IBM ACCESS KEY> --secret-key=<IBM SECRET ACCESS KEY> --endpoint=<IBM COS ENDPOINT> --target-bucket <bucket-name> -n openshift-storage
```

<backingstore_name>

バッキングストアの名前。

<IBM ACCESS KEY>、<IBM SECRET ACCESS KEY>、および <IBM COS ENDPOINT>

IBM アクセスキー ID、シークレットアクセスキー、および既存の IBM バケットの場所に対応する該当する地域エンドポイント
IBM クラウドで上記のキーを生成するには、ターゲットバケットのサービス認証情報を作成する際に HMAC 認証情報を含める必要があります。

<bucket-name>

既存の IBM バケット名。この引数は、バッキングストア、およびその後のデータストレージと管理のターゲットバケットとして使用するバケットに関する MCG を示します。
出力は次のようになります。

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "ibm-resource"
INFO[0002]   Created: Secret "backing-store-secret-ibm-resource"
```

YAML を使用してストレージリソースを追加する

1. 認証情報でシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
  namespace: openshift-storage
type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN
BASE64>
```

<IBM COS ACCESS KEY ID ENCODED IN BASE64> と <IBM COS SECRET ACCESS KEY ENCODED IN BASE64>

Base64 を使用して独自の IBM COS アクセスキー ID とシークレットアクセスキーを提供およびエンコードし、これらの属性の代わりに結果をそれぞれ使用します。

<backingstore-secret-name>

backingstore シークレットの名前。

2. 特定のバッキングストアについて以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  ibmCos:
    endpoint: <endpoint>
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
  type: ibm-cos
```

<bucket-name>

既存の IBM COS バケット名。この引数は、バッキングストアのターゲットバケットとして使用するバケットについて MCG に指示し、その後、データストレージと管理を行います。

<endpoint>

既存の IBM バケット名の場所に対応する地域エンドポイント。この引数は、バッキングストアに使用するエンドポイントを MCG に示し、その後、データストレージと管理を行います。

<backingstore-secret-name>

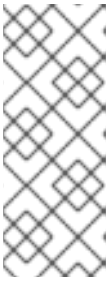
前のステップで作成されたシークレットの名前。

10.3.2.3. Azure でサポートされるバッキングストアの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```



注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。たとえば、IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- または、MCG パッケージを、Download Red Hat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

MCG コマンドラインインターフェイスの使用

- MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

```
noobaa backingstore create azure-blob <backingstore_name> --account-key=<AZURE
ACCOUNT KEY> --account-name=<AZURE ACCOUNT NAME> --target-blob-container
<blob container name> -n openshift-storage
```

<backingstore_name>

バッキングストアの名前。

<AZURE ACCOUNT KEY> および <AZURE ACCOUNT NAME>

この目的のために作成した AZURE アカウントキーとアカウント名。

<blob container name>

既存の Azure BLOB コンテナー名。この引数は、バッキングストアのターゲットバケットとして使用するバケットについて MCG に指示し、その後、データストレージと管理を行います。
出力は次のようになります。

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "azure-resource"
INFO[0002]   Created: Secret "backing-store-secret-azure-resource"
```

YAML を使用してストレージリソースを追加する

1. 認証情報でシークレットを作成します。

```
apiVersion: v1
kind: Secret
```

```
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  AccountName: <AZURE ACCOUNT NAME ENCODED IN BASE64>
  AccountKey: <AZURE ACCOUNT KEY ENCODED IN BASE64>
```

<AZURE ACCOUNT NAME ENCODED IN BASE64> および <AZURE ACCOUNT KEY ENCODED IN BASE64>

Base64 を使用して独自の Azure アカウント名とアカウントキーを指定してエンコードし、これらの属性の代わりに結果をそれぞれ使用します。

<backingstore-secret-name>

backingstore シークレットの一意の名前。

2. 特定のバッキングストアについて以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  azureBlob:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBlobContainer: <blob-container-name>
  type: azure-blob
```

<blob-container-name>

既存の Azure BLOB コンテナー名。この引数は、バッキングストアのターゲットバケットとして使用するバケットについて MCG に指示し、その後、データストレージと管理を行います。

<backingstore-secret-name>

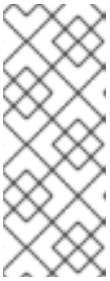
前の手順で作成したシークレットの名前に置き換えます。

10.3.2.4. GCP でサポートされるバッキングストアの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```



注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。たとえば、IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- または、MCG パッケージを、Download Red Hat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

MCG コマンドラインインターフェイスの使用

- MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

```
noobaa backingstore create google-cloud-storage <backingstore_name> --private-key-json-file=<PATH TO GCP PRIVATE KEY JSON FILE> --target-bucket <GCP bucket name> -n openshift-storage
```

<backingstore_name>

バッキングストアの名前。

<PATH TO GCP PRIVATE KEY JSON FILE>

この目的のために作成された GCP 秘密鍵へのパス。

<GCP bucket name>

既存の GCP オブジェクトストレージバケット名。この引数は、MCG に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。
出力は次のようになります。

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "google-gcp"
INFO[0002]   Created: Secret "backing-store-google-cloud-storage-gcp"
```

YAML を使用してストレージリソースを追加する

1. 認証情報でシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  GoogleServiceAccountPrivateKeyJson: <GCP PRIVATE KEY ENCODED IN BASE64>
```

<GCP PRIVATE KEY ENCODED IN BASE64>

Base64 を使用して独自の GCP サービスアカウントの秘密鍵を提供およびエンコードし、この属性の結果を使用します。

<backingstore-secret-name>

backingstore シークレットの一意の名前。

2. 特定のバッキングストアについて以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  googleCloudStorage:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <target bucket>
  type: google-cloud-storage
```

<target bucket>

既存の Google ストレージバケット。この引数は、バッキングストアのターゲットバケットとして使用するバケットについて MCG に指示し、その後、データストレージ dfdand 管理を行います。

<backingstore-secret-name>

前のステップで作成されたシークレットの名前。

10.3.2.5. ローカル永続ボリュームでサポートされるバッキングストアの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。

- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- または、MCG パッケージを、Download Red Hat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

MCG コマンドラインインターフェイスを使用したストレージリソースの追加

- MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

注記

このコマンドは、**openshift-storage** namespace 内から実行する必要があります。

```
$ noobaa -n openshift-storage backingstore create pv-pool <backingstore_name> --num-volumes <NUMBER OF VOLUMES>  --pv-size-gb <VOLUME SIZE> --request-cpu <CPU REQUEST> --request-memory <MEMORY REQUEST> --limit-cpu <CPU LIMIT> --limit-memory <MEMORY LIMIT> --storage-class <LOCAL STORAGE CLASS>
```

YAML を使用してストレージリソースの追加

- 特定のバッキングストアについて以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore_name>
  namespace: openshift-storage
spec:
   pvPool:
    numVolumes: <NUMBER OF VOLUMES>
    resources:
```

```
      requests:
        storage: <VOLUME SIZE>
        cpu: <CPU REQUEST>
        memory: <MEMORY REQUEST>
      limits:
        cpu: <CPU LIMIT>
        memory: <MEMORY LIMIT>
    storageClass: <LOCAL STORAGE CLASS>
  type: pv-pool
```

<backingstore_name>

バッキングストアの名前。

<NUMBER OF VOLUMES>

作成するボリュームの数。ボリュームの数を増やすと、ストレージが拡大することに注意してください。

<VOLUME SIZE>

各ボリュームの必要なサイズ (GB)。

<CPU REQUEST>

CPU ユニット **m** で要求された CPU の保証量。

<MEMORY REQUEST>

要求されたメモリー量の保証。

<CPU LIMIT>

CPU ユニット **m** で消費できる CPU の最大量。

<MEMORY LIMIT>

消費できるメモリーの最大量。

<LOCAL STORAGE CLASS>

ocs-storagecluster-ceph-rbd の使用を推奨するローカルストレージクラス名。
出力は次のようになります。

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Exists: BackingStore "local-mcg-storage"
```

10.3.3. s3 と互換性のある Multicloud Object Gateway バッキングストアの作成

Multicloud Object Gateway (MCG) は、任意の S3 と互換性のあるオブジェクトストレージをバッキングストアとして使用できます (例: Red Hat Ceph Storage の RADOS Object Gateway (RGW))。以下の手順では、Red Hat Ceph Storage の RGW 用の S3 と互換性のある MCG バッキングストアを作成する方法を説明します。RGW がデプロイされると、OpenShift Data Foundation operator は MCG の S3 と互換性のあるバッキングストアを自動的に作成することに注意してください。

手順

1. MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

注記

このコマンドは、**openshift-storage** namespace 内から実行する必要があります。

```
noobaa backingstore create s3-compatible rgw-resource --access-key=<RGW ACCESS KEY> --secret-key=<RGW SECRET KEY> --target-bucket=<bucket-name> --endpoint=<RGW endpoint> -n openshift-storage
```

a. **<RGW ACCESS KEY>** および **<RGW SECRET KEY>** を取得するには、RGW ユーザーシークレット名を使用して以下のコマンドを実行します。

```
oc get secret <RGW USER SECRET NAME> -o yaml -n openshift-storage
```

b. Base64 からアクセスキー ID とアクセスキーをデコードし、それらのキーを保持します。

c. **<RGW USER ACCESS KEY>** と **<RGW USER SECRET ACCESS KEY>** を、直前の手順でデコードした適切なデータに置き換えます。

d. **<bucket-name>** を既存の RGW バケット名に置き換えます。この引数は、MCG に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。

e. **<RGW endpoint>** を取得するには、RADOS Object Gateway S3 エンドポイントへのアクセス を参照してください。
出力は次のようになります。

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "rgw-resource"
INFO[0002]   Created: Secret "backing-store-secret-rgw-resource"
```

YAML を使用してバッキングストアを作成することもできます。

1. **CephObjectStore** ユーザーを作成します。これにより、RGW 認証情報が含まれるシークレットも作成されます。

```
apiVersion: ceph.rook.io/v1
kind: CephObjectStoreUser
metadata:
  name: <RGW-Username>
  namespace: openshift-storage
spec:
  store: ocs-storagecluster-cephobjectstore
  displayName: "<Display-name>"
```

a. **<RGW-Username>** と **<Display-name>** を、一意のユーザー名および表示名に置き換えます。

2. 以下の YAML を S3 と互換性のあるバッキングストアについて適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore-name>
  namespace: openshift-storage
spec:
```

```
  s3Compatible:
    endpoint: <RGW endpoint>
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    signatureVersion: v4
    targetBucket: <RGW-bucket-name>
  type: s3-compatible
```

a. **<backingstore-secret-name>** を、直前の手順で **CephObjectStore** で作成したシークレットの名前に置き換えます。

b. **<bucket-name>** を既存の RGW バケット名に置き換えます。この引数は、MCG に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。

c. **<RGW endpoint>** を取得するには、RADOS Object Gateway S3 エンドポイントへのアクセス を参照してください。

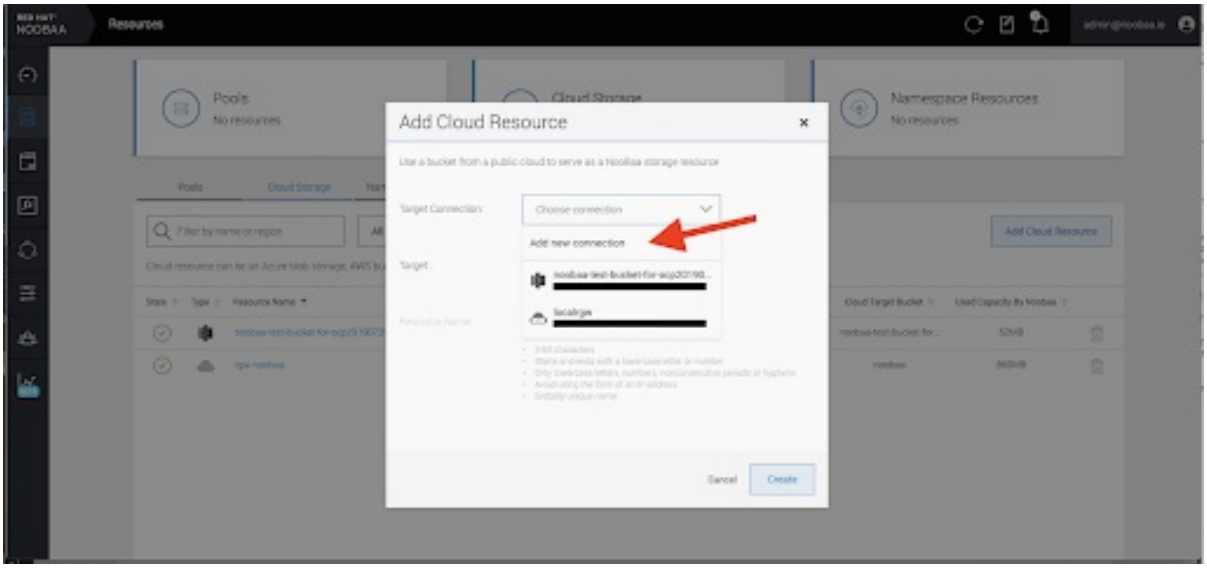
10.3.4. ユーザーインターフェイスを使用したハイブリッドおよびマルチクラウドのストレージリソースの追加

手順

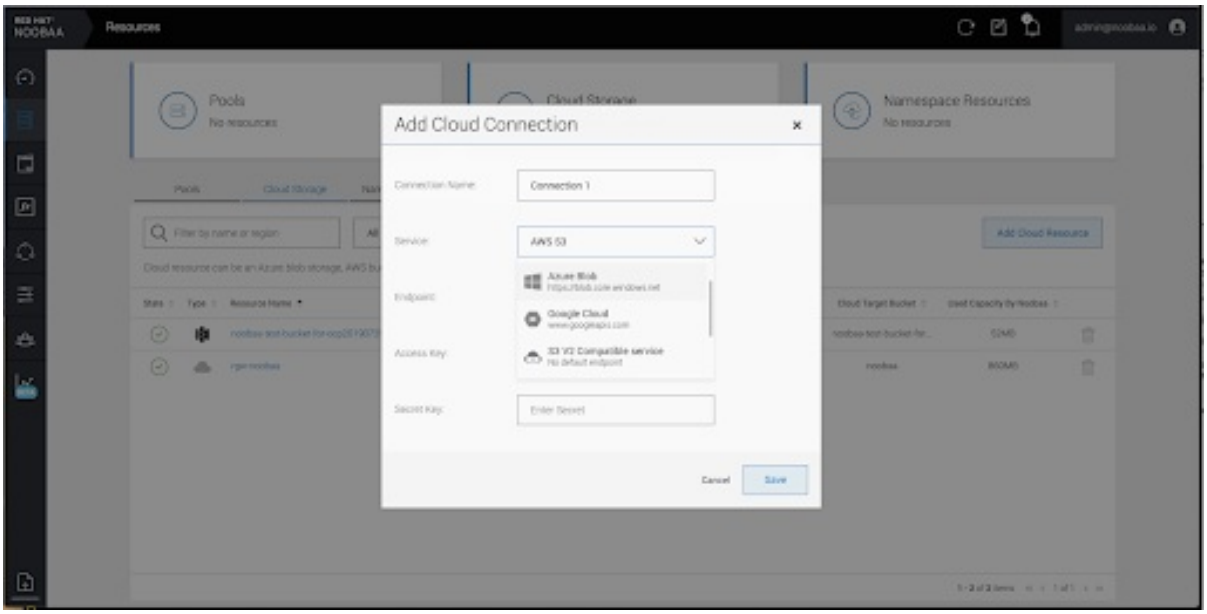
1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Storage Systems** タブでストレージシステムを選択し、**Overview → Object** タブをクリックします。
3. **Multicloud Object Gateway** のリンクをクリックします。

1. 以下に強調表示されているように左側にある **Resources** タブを選択します。設定する一覧から、**Add Cloud Resource** を選択します。

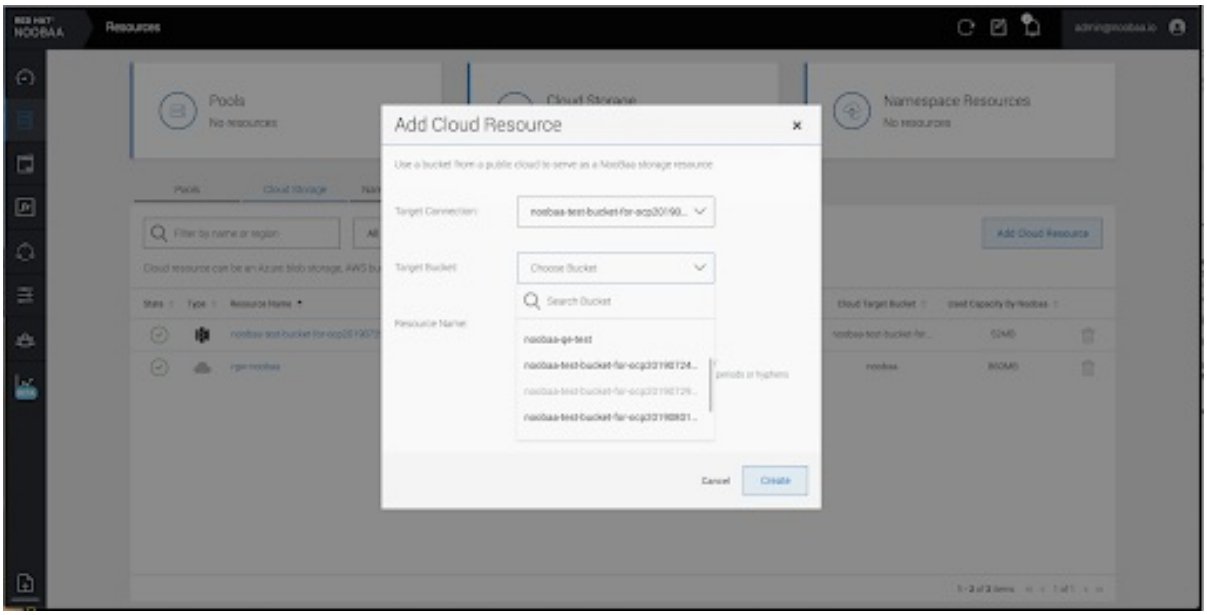
2. **Add new connection** を選択します。



3. 関連するネイティブクラウドプロバイダーまたは S3 互換オプションを選択し、詳細を入力します。



4. 新規に作成された接続を選択し、これを既存バケットにマップします。



5. これらの手順を繰り返して、必要な数のバッキングストアを作成します。

注記

NooBaa UI で作成されたリソースは、OpenShift UI または MCG CLI では使用できません。

10.3.5. 新規バケットクラスの作成

バケットクラスは、OBC (Object Bucket Class) の階層ポリシーおよびデータ配置を定義するバケットのクラスを表す CRD です。

以下の手順を使用して、OpenShift Data Foundation でバケットクラスを作成します。

手順

1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Bucket Class** タブをクリックします。
3. **Create Bucket Class** をクリックします。
4. Create new Bucket Class ページで、以下を実行します。
   a. バケットクラスタイプを選択し、バケットクラス名を入力します。
      i. **BucketClass** タイプを選択します。以下のいずれかのオプションを選択します。
         - **Standard**: データは Multicloud Object Gateway (MCG) に使用され、重複排除、圧縮、および暗号化されます。
         - **Namespace**: データは、重複排除、圧縮、または暗号化を実行せずに NamespaceStores に保存されます。
           デフォルトでは、**Standard** が選択されます。
      ii. **Bucket Class Name** 名を入力します。
      iii. **Next** をクリックします。
   b. **Placement Policy** で **Tier 1 - Policy Type** を選択し、**Next** をクリックします。要件に応じて、いずれかのオプションを選択できます。
      - **Spread** により、選択したリソース全体にデータを分散できます。
      - **Mirror** により、選択したリソース全体でデータを完全に複製できます。
      - **Add Tier** をクリックし、別のポリシー階層を追加します。
   c. **Tier 1 - Policy Type** で **Spread** を選択した場合は、利用可能な一覧から 1 つ以上の **Backing Store** リソースを選択してから、**Next** をクリックします。また、新しいバッキングストアを作成 することも可能です。

注記

直前の手順で Policy Type に Mirror を選択する場合は、2 つ以上のバッキングストアを選択する必要があります。

d. Bucket Class 設定を確認し、確認します。

e. **Create Bucket Class** をクリックします。

検証手順

1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Bucket Class** タブをクリックし、新しい Bucket Class を検索します。

10.3.6. バケットクラスの編集

以下の手順に従って、Openshift Web コンソールの **edit** ボタンをクリックし、YAML ファイルを使用してバケットクラスコンポーネントを編集します。

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 コンソールで、**Storage → Data Foundation** をクリックします。
2. **Bucket Class** タブをクリックします。
3. 編集する Bucket クラスの横にあるアクションメニュー **(⋮)** をクリックします。
4. **Edit Bucket Class** をクリックします。
5. **YAML** ファイルにリダイレクトされ、このファイルで必要な変更を加え、**Save** をクリックします。

10.3.7. バケットクラスのバッキングストアの編集

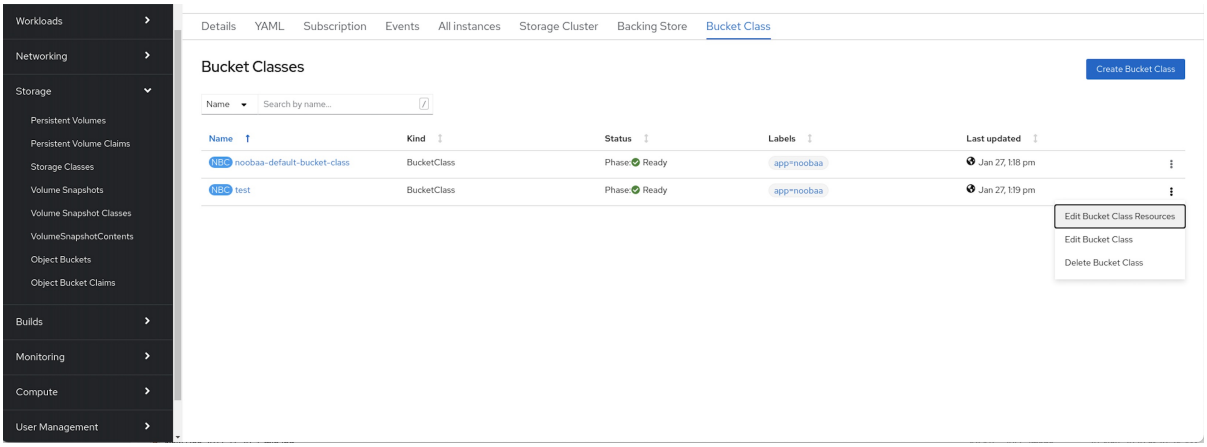
以下の手順を使用して、既存の Multicloud Object Gateway (MCG) バケットクラスを編集し、バケットクラスで使用される基礎となるバッキングストアを変更します。

前提条件

- OpenShift Web コンソールへの管理者アクセス。
- バケットクラス。
- バッキングストア。

手順

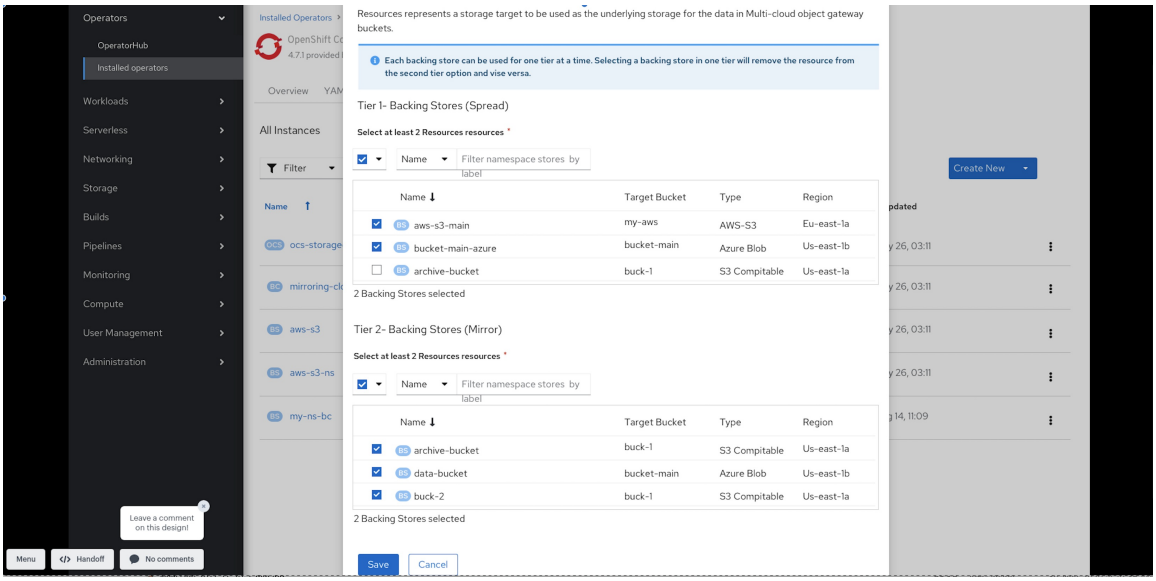
1. OpenShift Web コンソールで、**Storage → Data Foundation** をクリックします。
2. **Bucket Class** タブをクリックします。
3. 編集する Bucket クラスの横にあるアクションメニュー **(⋮)** をクリックします。



4. **Edit Bucket Class Resources** をクリックします。

5. **Edit Bucket Class Resources** ページで、バッキングストアをバケットクラスに追加するか、バケットクラスからバッキングストアを削除してバケットクラスリソースを編集します。1 つまたは 2 つの層を使用して作成されたバケットクラスリソースや、異なる配置ポリシーが指定されたバケットクラスリソースを編集することもできます。

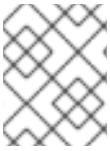
   - バッキングストアをバケットクラスに追加するには、バッキングストアの名前を選択します。
   - バケットクラスからバッキングストアを削除するには、バッキングストアの名前を消去します。



6. **Save** をクリックします。

10.4. NAMESPACE バケットの管理

namespace バケットを使用すると、異なるプロバイダーのデータリポジトリーを接続できるため、単一の統合ビューを使用してすべてのデータと対話できます。各プロバイダーに関連付けられたオブジェクトバケットを namespace バケットに追加し、namespace バケット経由でデータにアクセスし、一度にすべてのオブジェクトバケットを表示します。これにより、他の複数のストレージプロバイダーから読み込む間に、希望するストレージプロバイダーへの書き込みを行うことができ、新規ストレージプロバイダーへの移行コストが大幅に削減されます。



注記

namespace バケットは、このバケットの書き込みターゲットが利用可能で機能している場合にのみ使用できます。

10.4.1. namespace バケットのオブジェクトの Amazon S3 API エンドポイント

Amazon Simple Storage Service (S3) API を使用して namespace バケットのオブジェクトと対話できます。

Red Hat OpenShift Data Foundation 4.6 以降では、以下の namespace バケット操作をサポートします。

- ListObjectVersions
- ListObjects
- PutObject
- CopyObject
- ListParts
- CreateMultipartUpload
- CompleteMultipartUpload
- UploadPart
- UploadPartCopy
- AbortMultipartUpload
- GetObjectAcl
- GetObject
- HeadObject
- DeleteObject
- DeleteObjects

これらの操作および使用方法に関する最新情報は、Amazon S3 API リファレンスのドキュメントを参照してください。

関連情報

- Amazon S3 REST API Reference
- Amazon S3 CLI Reference

10.4.2. Multicloud Object Gateway CLI および YAML を使用した namespace バケットの追加

namespace バケットの詳細は、namespace バケットの管理 を参照してください。

デプロイメントのタイプに応じて、また YAML または Multicloud Object Gateway CLI を使用するかどうかに応じて、以下の手順のいずれかを選択して namespace バケットを追加します。

- YAML を使用した AWS S3 namespace バケットの追加
- YAML を使用した IBM COS namespace バケットの追加
- Multicloud Object Gateway CLI を使用した AWS S3 namespace バケットの追加
- Multicloud Object Gateway CLI を使用した IBM COS namespace バケットの追加

10.4.2.1. YAML を使用した AWS S3 namespace バケットの追加

前提条件

- OpenShift Data Foundation Operator を使用した OpenShift Container Platform のインストール
- Multicloud Object Gateway (MCG) へのアクセス。
  詳細は、第 2 章 アプリケーションによるマルチクラウドオブジェクトゲートウェイへのアクセス を参照してください。

手順

1. 認証情報でシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
  type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

   ここで、**<namespacestore-secret-name>** は一意の NamespaceStore 名になります。

   Base64 を使用して独自の AWS アクセスキー ID およびシークレットアクセスキーを指定してエンコードし、その結果を **<AWS ACCESS KEY ID ENCODED IN BASE64>** および **<AWS SECRET ACCESS KEY ENCODED IN BASE64>** の代わりに使用する必要があります。

2. OpenShift カスタムリソース定義 (CRD) を使用して NamespaceStore リソースを作成します。NamespaceStore は、MCG namespace バケットでデータの **read** または **write** ターゲットとして使用される基礎となるストレージを表します。

   NamespaceStore リソースを作成するには、以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <resource-name>
```

```
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    targetBucket: <target-bucket>
  type: aws-s3
```

<resource-name>

リソースに指定する名前。

<namespacestore-secret-name>

前の手順で作成されたシークレット

<namespace-secret>

シークレットが見つかる namespace。

<target-bucket>

NamespaceStore 用に作成したターゲットバケット。

3. namespace バケットの namespace ポリシーを定義する namespace バケットクラスを作成します。namespace ポリシーには、**single** または **multi** のタイプが必要です。

- タイプ **single** の namespace ポリシーには、以下の設定が必要です。

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type:
    single:
      resource: <resource>
```

<my-bucket-class>

一意の namespace バケットクラス名。

<resource>

namespace バケットの読み取りおよび書き込みターゲットを定義する単一の NamespaceStore の名前。

- タイプが **multi** の namespace ポリシーには、以下の設定が必要です。

```
apiVersion: noobaa.io/v1alpha1

kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
```

```
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
      - <read-resources>
      - <read-resources>
```

<my-bucket-class>

一意のバケットクラス名。

<write-resource>

namespace バケットの **write** ターゲットを定義する単一の NamespaceStore の名前。

<read-resources>

namespace バケットの **read** ターゲットを定義する NamespaceStores の名前のリスト。

4. 以下の YAML を使用して前の手順に定義されたバケットクラスを使用する Object Bucket Class (OBC) リソースを使用して、バケットを作成します。

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
  namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

<resource-name>

リソースに指定する名前。

<my-bucket>

バケットに指定したい名前。

<my-bucket-class>

前のステップで作成されたバケットクラス。

OBC が Operator によってプロビジョニングされると、バケットが MCG で作成され、Operator は OBC と同じ namespace 上に同じ名前で **Secret** および **ConfigMap** を作成します。

10.4.2.2. YAML を使用した IBM COS namespace バケットの追加

前提条件

- OpenShift Data Foundation Operator を使用した OpenShift Container Platform のインストール
- Multicloud Object Gateway (MCG) へのアクセスについては、第 2 章の Accessing the Multicloud Object Gateway with your applications を参照してください。

手順

1. 認証情報でシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
  type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN
BASE64>
```

<namespacestore-secret-name>

一意の NamespaceStore 名。
Base64 を使用して独自の IBM COS アクセスキー ID およびシークレットアクセスキーを指定してエンコードし、その結果を **<IBM COS ACCESS KEY ID ENCODED IN BASE64>** および **<IBM COS SECRET ACCESS KEY ENCODED IN BASE64>** の代わりに使用する必要があります。

2. OpenShift カスタムリソース定義 (CRD) を使用して NamespaceStore リソースを作成します。NamespaceStore は、MCG namespace バケットでデータの **read** または **write** ターゲットとして使用される基礎となるストレージを表します。

NamespaceStore リソースを作成するには、以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    signatureVersion: v2
    targetBucket: <target-bucket>
  type: ibm-cos
```

<IBM COS ENDPOINT>

適切な IBM COS エンドポイント。

<namespacestore-secret-name>

前の手順で作成されたシークレット

<namespace-secret>

シークレットが見つかる namespace。

<target-bucket>

NamespaceStore 用に作成したターゲットバケット。

3. namespace バケットの namespace ポリシーを定義する namespace バケットクラスを作成します。namespace ポリシーには、**single** または **multi** のタイプが必要です。

- タイプ **single** の namespace ポリシーには、以下の設定が必要です。

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type:
    single:
      resource: <resource>
```

<my-bucket-class>

一意の namespace バケットクラス名。

<resource>

namespace バケットの **read** および **write** ターゲットを定義する単一の NamespaceStore の名前。

- タイプが **multi** の namespace ポリシーには、以下の設定が必要です。

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
      - <read-resources>
      - <read-resources>
```

<my-bucket-class>

一意のバケットクラス名。

<write-resource>

namespace バケットの書き込みターゲットを定義する単一の NamespaceStore の名前。

<read-resources>

namespace バケットの **read** ターゲットを定義する NamespaceStores 名のリスト。

4. 以下の YAML を適用して、前の手順で定義されたバケットクラスを使用する Object Bucket Class (OBC) リソースを使用してバケットを作成します。

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
  namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: openshift-storage.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

<resource-name>

リソースに指定する名前。

<my-bucket>

バケットに指定したい名前。

<my-bucket-class>

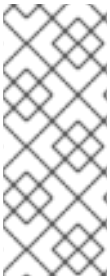
前のステップで作成されたバケットクラス。
OBC が Operator によってプロビジョニングされると、バケットが MCG で作成され、Operator は OBC と同じ namespace 上に同じ名前で **Secret** および **ConfigMap** を作成します。

10.4.2.3. Multicloud Object Gateway CLI を使用した AWS S3 namespace バケットの追加

前提条件

- OpenShift Data Foundation Operator を使用した OpenShift Container Platform のインストール
- Multicloud Object Gateway (MCG) へのアクセスについては、第 2 章の Accessing the Multicloud Object Gateway with your applications を参照してください。
- MCG コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

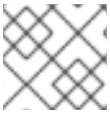


注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。たとえば、IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

または、MCG パッケージを、https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package にある OpenShift Data Foundation RPM からインストールできます。



注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

1. MCG コマンドラインインターフェイスで、NamespaceStore リソースを作成します。NamespaceStore は、MCG namespace バケットでデータの **read** または **write** ターゲットとして使用される基礎となるストレージを表します。

```
$ noobaa namespacestore create aws-s3 <namespacestore> --access-key <AWS ACCESS KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

<namespacestore>

NamespaceStore の名前。

<AWS ACCESS KEY> と <AWS SECRET ACCESS KEY>

この目的のために作成した AWS アクセスキー ID とシークレットアクセスキー。

<bucket-name>

既存の AWS バケット名。この引数は、MCG に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。

2. namespace バケットの namespace ポリシーを定義する namespace バケットクラスを作成します。namespace ポリシーは、**single** または **multi** のいずれかになります。

- タイプが **single** の namespace ポリシーを使用して namespace バケットクラスを作成するには、以下を実行します。

```
$ noobaa bucketclass create namespace-bucketclass single <my-bucket-class> --resource <resource> -n openshift-storage
```

<resource-name>

リソースに指定する名前。

<my-bucket-class>

一意のバケットクラス名。

<resource>

namespace バケットの **read** および **write** ターゲットを定義する単一の namespace-store。

- タイプ **multi** の namespace ポリシーを使用して namespace バケットクラスを作成するには、以下を実行します。

```
$ noobaa bucketclass create namespace-bucketclass multi <my-bucket-class> --write-resource <write-resource> --read-resources <read-resources> -n openshift-storage
```

<resource-name>

リソースに指定する名前。

<my-bucket-class>

一意のバケットクラス名。

<write-resource>

namespace バケットの **write** ターゲットを定義する単一の namespace-store。

<read-resources>s

namespace バケットの **read** ターゲットを定義する、コンマで区切られた namespace-store の一覧。

3. 前の手順で定義したバケットクラスを使用する Object Bucket Class (OBC) リソースを使用して、バケットを作成します。

```
$ noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --bucketclass <custom-bucket-class>
```

<bucket-name>

選択したバケット名。

<custom-bucket-class>

前の手順で作成したバケットクラスの名前。

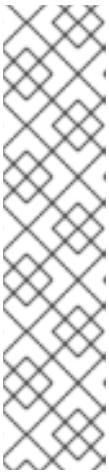
OBC が Operator によってプロビジョニングされると、バケットが MCG で作成され、Operator は OBC と同じ namespace 上に同じ名前で **Secret** および **ConfigMap** を作成します。

10.4.2.4. Multicloud Object Gateway CLI を使用した IBM COS namespace バケットの追加

前提条件

- OpenShift Data Foundation Operator を使用した OpenShift Container Platform のインストール
- Multicloud Object Gateway (MCG) へのアクセスについては、第 2 章の Accessing the Multicloud Object Gateway with your applications を参照してください。
- MCG コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```



注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。

- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

または、MCG パッケージを、https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package にある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

1. MCG コマンドラインインターフェイスで、NamespaceStore リソースを作成します。NamespaceStore は、MCG namespace バケットでデータの **read** または **write** ターゲットとして使用される基礎となるストレージを表します。

```
$ noobaa namespacestore create ibm-cos <namespacestore> --endpoint <IBM COS ENDPOINT> --access-key <IBM ACCESS KEY> --secret-key <IBM SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

<namespacestore>

NamespaceStore の名前。

<IBM ACCESS KEY>、<IBM SECRET ACCESS KEY>、<IBM COS ENDPOINT>

IBM アクセスキー ID、シークレットアクセスキー、および既存の IBM バケットの場所に対応する該当する地域エンドポイント

<bucket-name>

既存の IBM バケット名。この引数は、MCG に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。

2. namespace バケットの namespace ポリシーを定義する namespace バケットクラスを作成します。namespace ポリシーには、**single** または **multi** のタイプが必要です。

- タイプが **single** の namespace ポリシーを使用して namespace バケットクラスを作成するには、以下を実行します。

```
$ noobaa bucketclass create namespace-bucketclass single <my-bucket-class> --resource <resource> -n openshift-storage
```

<resource-name>

リソースに指定する名前。

<my-bucket-class>

一意のバケットクラス名。

<resource>

namespace バケットの **read** および **write** ターゲットを定義する単一の NamespaceStore。

- タイプ **multi** の namespace ポリシーを使用して namespace バケットクラスを作成するには、以下を実行します。

```
$ noobaa bucketclass create namespace-bucketclass multi <my-bucket-class> --write-resource <write-resource> --read-resources <read-resources> -n openshift-storage
```

<resource-name>

リソースに指定する名前。

<my-bucket-class>

一意のバケットクラス名。

<write-resource>

namespace バケットの **write** ターゲットを定義する単一の NamespaceStore。

<read-resources>

namespace バケットの **read** ターゲットを定義する NamespaceStores のコンマ区切りリスト。

3. 前の手順で定義したバケットクラスを使用する Object Bucket Class (OBC) リソースを使用して、バケットを作成します。

```
$ noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --bucketclass <custom-bucket-class>
```

<bucket-name>

選択したバケット名。

<custom-bucket-class>

前の手順で作成したバケットクラスの名前。

OBC が Operator によってプロビジョニングされると、バケットが MCG で作成され、Operator は OBC と同じ namespace 上に同じ名前で **Secret** および **ConfigMap** を作成します。

10.4.3. OpenShift Container Platform ユーザーインターフェイスを使用した namespace バケットの追加

OpenShift Container Platform ユーザーインターフェイスを使用して namespace バケットを追加できます。namespace バケットの詳細については、namespace バケットの管理 を参照してください。

前提条件

- OpenShift Data Foundation Operator を使用した OpenShift Container Platform のインストール
- Multicloud Object Gateway (MCG) へのアクセス。

手順

1. OpenShift Web コンソールにログインします。
2. **Storage → Data Foundation** をクリックします。
3. **Namespace Store** タブをクリックして、namespace バケットで使用される **namespacestore** リソースを作成します。
   a. **Create namespace store**をクリックします。
   b. namespacestore 名を入力します。
   c. プロバイダーを選択します。
   d. リージョンを選択します。
   e. 既存のシークレットを選択するか、**Switch to credentials** をクリックして、シークレットキーおよびシークレットアクセスキーを入力してシークレットを作成します。
   f. ターゲットバケットを選択します。

g. **Create** をクリックします。

h. namespacestore が **Ready** 状態にあることを確認します。

i. 必要なリソースが得られるまで、これらの手順を繰り返します。

4. **Bucket Class** タブ → **Create a new Bucket Class**をクリックします。

a. **Namespace** ラジオボタンを選択します。

b. Bucket Class 名を入力します。

c. (オプション) 説明を追加します。

d. **Next** をクリックします。

5. namespace バケットの namespace ポリシータイプを選択し、**Next** をクリックします。

6. ターゲットリソースを選択します。

- namespace ポリシータイプが **Single** の場合、読み取りリソースを選択する必要があります。
- namespace ポリシータイプが **Multi** の場合、読み取りリソースおよび書き込みリソースを選択する必要があります。
- namespace ポリシータイプが **Cache** の場合は、namespace バケットの読み取りおよび書き込みターゲットを定義する Hub namespace ストアを選択する必要があります。

7. **Next** をクリックします。

8. 新しいバケットクラスを確認してから **Create Bucketclass** をクリックします。

9. **BucketClass** ページで、新たに作成されたリソースが **Created** フェーズにあることを確認します。

10. OpenShift Web コンソールで、**Storage** → **Data Foundation** をクリックします。

11. **Status** カードで **Storage System** をクリックし、表示されるポップアップからストレージシステムリンクをクリックします。

12. **Object** タブで、**Multicloud Object Gateway** → **Buckets** → **Namespace Buckets** タブをクリックします。

13. **Create Namespace Bucket**をクリックします。

a. **Choose Name** タブで、namespace バケットの名前を指定し、**Next** をクリックします。

b. **Set Placement** タブで、以下を実行します。

i. **Read Policy** で、namespace バケットがデータの読み取りに使用する、前の手順で作成した各 namespace リソースのチェックボックスを選択します。

ii. 使用している namespace ポリシータイプが **Multi** の場合、**Write Policy** の場合は、namespace バケットがデータを書き込む namespace リソースを指定します。

iii. **Next** をクリックします。

c. **Create** をクリックします。

検証手順

- namespace バケットが **State** 列の緑色のチェックマークと、予想される読み取りリソースの数、および予想される書き込みリソース名と共に一覧表示されていることを確認します。

10.5. ハイブリッドおよびマルチクラウドバケットのデータのミラーリング

Multicloud Object Gateway (MCG) の簡素化されたプロセスを使用して、クラウドプロバイダーとクラスター全体にデータを広げることができます。データ管理ポリシーとミラーリングを反映するバケットクラスを作成する前に、MCG で使用できるバッキングストレージを追加する必要があります。詳細は、第 4 章 「ハイブリッドまたはマルチクラウド用のストレージリソースの追加」 を参照してください。

OpenShift UI、YAML、または MCG コマンドラインインターフェイスを使用して、ミラーリングデータを設定できます。

以下のセクションを参照してください。

- セクション 6.2、「MCG コマンドラインインターフェイスを使用したデータのミラーリング用のバケットクラスの作成」
- セクション 6.3、「YAML を使用したデータのミラーリング用のバケットクラスの作成」

10.5.1. MCG コマンドラインインターフェイスを使用したデータのミラーリング用のバケットクラスの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスを必ずダウンロードしてください。

手順

1. Multicloud Object Gateway (MCG) コマンドラインインターフェイスから以下のコマンドを実行し、ミラーリングポリシーでバケットクラスを作成します。

```
$ noobaa bucketclass create placement-bucketclass mirror-to-aws --backingstores=azure-resource,aws-resource --placement Mirror
```

2. 新たに作成されたバケットクラスを新規のバケット要求に設定し、2 つのロケーション間でミラーリングされる新規バケットを生成します。

```
$ noobaa obc create  mirrored-bucket --bucketclass=mirror-to-aws
```

10.5.2. YAML を使用したデータのミラーリング用のバケットクラスの作成

1. 以下の YAML を適用します。この YAML は、ローカル Ceph ストレージと AWS 間でデータをミラーリングするハイブリッドの例です。

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
```

```
metadata:
  labels:
    app: noobaa
  name: <bucket-class-name>
  namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
    - backingStores:
      - <backing-store-1>
      - <backing-store-2>
      placement: Mirror
```

2. 以下の行を標準の Object Bucket Claim (オブジェクトバケット要求、OBC) に追加します。

```
additionalConfig:
  bucketclass: mirror-to-aws
```

OBC についての詳細は、「Object Bucket Claim(オブジェクトバケット要求)」を参照してください。

10.6. MULTICLOUD OBJECT GATEWAY のバケットポリシー

OpenShift Data Foundation は AWS S3 バケットポリシーをサポートします。バケットポリシーにより、ユーザーにバケットとそれらのオブジェクトのアクセスパーミッションを付与することができます。

10.6.1. バケットポリシーの概要

バケットポリシーは、AWS S3 バケットおよびオブジェクトにパーミッションを付与するために利用できるアクセスポリシーオプションです。バケットポリシーは JSON ベースのアクセスポリシー言語を使用します。アクセスポリシー言語についての詳細は、AWS Access Policy Language Overview を参照してください。

10.6.2. Multicloud Object Gateway でのバケットポリシーの使用

前提条件

- 実行中の OpenShift Data Foundation Platform。
- Multicloud Object Gateway (MCG) へのアクセス。「アプリケーションの使用による Multicloud Object Gateway へのアクセス」を参照してください。

手順

MCG でバケットポリシーを使用するには、以下を実行します。

1. JSON 形式でバケットポリシーを作成します。
以下に例を示します。

```
{
    "Version": "NewVersion",
    "Statement": [
        {
```

```
            "Sid": "Example",
            "Effect": "Allow",
            "Principal": [
                    "john.doe@example.com"
            ],
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::john_bucket"
            ]
        }
    ]
}
```

2. AWS S3 クライアントを使用して **put-bucket-policy** コマンドを使用してバケットポリシーを S3 バケットに適用します。

```
# aws --endpoint ENDPOINT --no-verify-ssl s3api put-bucket-policy --bucket MyBucket --policy BucketPolicy
```

   a. **ENDPOINT** を S3 エンドポイントに置き換えます。

   b. **MyBucket** を、ポリシーを設定するバケットに置き換えます。

   c. **BucketPolicy** をバケットポリシー JSON ファイルに置き換えます。

   d. デフォルトの自己署名証明書を使用している場合は、**--no-verify-ssl** を追加します。
   以下に例を示します。

```
# aws --endpoint https://s3-openshift-storage.apps.gogo44.noobaa.org --no-verify-ssl s3api put-bucket-policy -bucket MyBucket --policy file://BucketPolicy
```

   put-bucket-policy コマンドについての詳細は、AWS CLI Command Reference for put-bucket-policy を参照してください。

注記

主となる要素では、リソース (バケットなど) へのアクセスを許可または拒否されるユーザーを指定します。現在、NooBaa アカウントのみがプリンシパルとして使用できます。Object Bucket Claim (オブジェクトバケット要求) の場合、NooBaa はアカウント **obc-account.<generated bucket name>@noobaa.io** を自動的に作成します。

注記

バケットポリシー条件はサポートされていません。

関連情報

- アクセスパーミッションに関して、バケットポリシーには数多くの利用可能な要素があります。

- これらの要素の詳細と、それらを使用してアクセスパーミッションを制御する方法の例は、AWS Access Policy Language Overview を参照してください。
- バケットポリシーの他の例については、AWS Bucket Policy Examples を参照してください。

10.6.3. Multicloud Object Gateway でのユーザーの作成

前提条件

- 実行中の OpenShift Data Foundation Platform。
- MCG コマンドラインインターフェイスをダウンロードして、管理を容易にします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。

- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

- または、MCG パッケージを、Download RedHat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

次のコマンドを実行して、MCG ユーザーアカウントを作成します。

```
noobaa account create <noobaa-account-name> [--allow_bucket_create=true] [--allowed_buckets=[]]
[--default_resource=''] [--full_permission=false]
```

<noobaa-account-name>
新しい MCG ユーザーアカウントの名前を指定します。

--allow_bucket_create
ユーザーが新しいバケットを作成できるようにします。

--allowed_buckets
ユーザーの許可されたバケットリストを設定します (コンマまたは複数のフラグを使用)。

--default_resource

デフォルトのリソースを設定します。新しいバケットは、このデフォルトのリソース (将来のリソースを含む) で作成されます。

--full_permission

このアカウントが既存および将来のすべてのバケットにアクセスできるようにします。

重要

少なくとも 1 つのバケットにアクセスするためのアクセス許可、またはすべてのバケットにアクセスするための完全なアクセス許可を提供する必要があります。

10.7. OBJECT BUCKET CLAIM(オブジェクトバケット要求)

Object Bucket Claim(オブジェクトバケット要求) は、ワークロードの S3 と互換性のあるバケットバックエンドを要求するために使用できます。

Object Bucket Claim(オブジェクトバケット要求) は 3 つの方法で作成できます。

- 「動的 Object Bucket Claim(オブジェクトバケット要求)」
- 「コマンドラインインターフェイスを使用した Object Bucket Claim(オブジェクトバケット要求) の作成」
- 「OpenShift Web コンソールを使用した Object Bucket Claim(オブジェクトバケット要求) の作成」

Object Bucket Claim(オブジェクトバケット要求) は、新しいアクセスキーおよびシークレットアクセスキーを含む、バケットのパーミッションのある NooBaa の新しいバケットとアプリケーションアカウントを作成します。アプリケーションアカウントは単一バケットにのみアクセスでき、デフォルトで新しいバケットを作成することはできません。

10.7.1. 動的 Object Bucket Claim(オブジェクトバケット要求)

永続ボリュームと同様に、Object Bucket Claim (OBC) の詳細をアプリケーションの YAML に追加し、設定マップおよびシークレットで利用可能なオブジェクトサービスエンドポイント、アクセスキー、およびシークレットアクセスキーを取得できます。この情報をアプリケーションの環境変数に動的に読み込むことは容易に実行できます。

注記

Multicloud Object Gateway エンドポイントは、OpenShift が自己署名証明書を使用する場合にのみ、自己署名証明書を使用します。OpenShift で署名付き証明書を使用すると、Multicloud Object Gateway エンドポイント証明書が署名付き証明書に自動的に置き換えられます。ブラウザーを介してエンドポイントにアクセスし、Multicloud Object Gateway で現在使用されている証明書を取得します。詳細は、アプリケーションの使用による Multicloud Object Gateway へのアクセス を参照してください。

手順

1. 以下の行をアプリケーション YAML に追加します。

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <obc-name>
```

```
spec:
  generateBucketName: <obc-bucket-name>
  storageClassName: openshift-storage.noobaa.io
```

これらの行は OBC 自体になります。

a. **<obc-name>** を、一意の OBC の名前に置き換えます。

b. **<obc-bucket-name>** を、OBC の一意のバケット名に置き換えます。

2. YAML ファイルにさらに行を追加して、OBC の使用を自動化します。
以下に例を示します。

```
apiVersion: batch/v1
kind: Job
metadata:
  name: testjob
spec:
  template:
    spec:
      restartPolicy: OnFailure
      containers:
        - image: <your application image>
          name: test
          env:
            - name: BUCKET_NAME
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_NAME
            - name: BUCKET_HOST
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_HOST
            - name: BUCKET_PORT
              valueFrom:
                configMapKeyRef:
                  name: <obc-name>
                  key: BUCKET_PORT
            - name: AWS_ACCESS_KEY_ID
              valueFrom:
                secretKeyRef:
                  name: <obc-name>
                  key: AWS_ACCESS_KEY_ID
            - name: AWS_SECRET_ACCESS_KEY
              valueFrom:
                secretKeyRef:
                  name: <obc-name>
                  key: AWS_SECRET_ACCESS_KEY
```

以下は、バケット要求の結果のマッピングの例になります。これは、データを含む設定マップおよび認証情報を含むシークレットです。この特定のジョブは NooBaa からオブジェクトバケットを要求し、バケットとアカウントを作成します。

a. **<obc-name>** のすべてのインスタンスを、OBC の名前に置き換えます。

b. **<your application image>** をアプリケーションイメージに置き換えます。

3. 更新された YAML ファイルを適用します。

```
# oc apply -f <yaml.file>
```

<yaml.file> を YAML ファイルの名前に置き換えます。

4. 新しい設定マップを表示するには、以下を実行します。

```
# oc get cm <obc-name> -o yaml
```

obc-name を OBC の名前に置き換えます。

出力には、以下の環境変数が表示されることが予想されます。

- **BUCKET_HOST**: アプリケーションで使用するエンドポイント
- **BUCKET_PORT**: アプリケーションで利用できるポート
  - ポートは **BUCKET_HOST** に関連します。たとえば、**BUCKET_HOST** が https://my.example.com で、**BUCKET_PORT** が 443 の場合、オブジェクトサービスのエンドポイントは https://my.example.com:443 になります。
- **BUCKET_NAME**: 要求されるか、生成されるバケット名
- **AWS_ACCESS_KEY_ID**: 認証情報の一部であるアクセスキー
- **AWS_SECRET_ACCESS_KEY**: 認証情報の一部であるシークレットのアクセスキー

重要

AWS_ACCESS_KEY_ID と **AWS_SECRET_ACCESS_KEY** を取得します。名前は、AWS S3 と互換性があるように使用されます。S3 操作の実行中、特に Multicloud Object Gateway (MCG) バケットから読み取り、書き込み、または一覧表示する場合は、キーを指定する必要があります。キーは Base64 でエンコードされています。キーを使用する前に、キーをデコードしてください。

```
# oc get secret <obc_name> -o yaml
```

<obc_name>

オブジェクトバケットクレームの名前を指定します。

10.7.2. コマンドラインインターフェイスを使用した Object Bucket Claim(オブジェクトバケット要求) の作成

コマンドラインインターフェイスを使用して Object Bucket Claim (OBC) を作成する場合、設定マップとシークレットを取得します。これらには、アプリケーションがオブジェクトストレージサービスを使用するために必要なすべての情報が含まれます。

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。

- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

手順

1. コマンドラインインターフェイスを使用して、新規バケットおよび認証情報の詳細を生成します。
以下のコマンドを実行します。

```
# noobaa obc create <obc-name> -n openshift-storage
```

<obc-name> を一意の OBC 名に置き換えます (例: **myappobc**)。

さらに、**--app-namespace** オプションを使用して、OBC 設定マップおよびシークレットが作成される namespace を指定できます (例: **myapp-namespace**)。

以下に例を示します。

```
INFO[0001]   Created: ObjectBucketClaim "test21obc"
```

MCG コマンドラインインターフェイスが必要な設定を作成し、新規 OBC について OpenShift に通知します。

2. 以下のコマンドを実行して OBC を表示します。

```
# oc get obc -n openshift-storage
```

以下に例を示します。

```
NAME       STORAGE-CLASS               PHASE   AGE
test21obc  openshift-storage.noobaa.io  Bound   38s
```

3. 以下のコマンドを実行して、新規 OBC の YAML ファイルを表示します。

```
# oc get obc test21obc -o yaml -n openshift-storage
```

以下に例を示します。

```
apiVersion: objectbucket.io/v1alpha1
```

```
kind: ObjectBucketClaim
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  generation: 2
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  resourceVersion: "40756"
  selfLink: /apis/objectbucket.io/v1alpha1/namespaces/openshift-
storage/objectbucketclaims/test21obc
  uid: 64f04cba-f662-11e9-bc3c-0295250841af
spec:
  ObjectBucketName: obc-openshift-storage-test21obc
  bucketName: test21obc-933348a6-e267-4f82-82f1-e59bf4fe3bb4
  generateBucketName: test21obc
  storageClassName: openshift-storage.noobaa.io
status:
  phase: Bound
```

4. **openshift-storage** namespace 内で、設定マップおよびシークレットを見つけ、この OBC を使用することができます。CM とシークレットの名前はこの OBC の名前と同じです。
以下のコマンドを実行してシークレットを表示します。

```
# oc get -n openshift-storage secret test21obc -o yaml
```

以下に例を示します。

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: c0M0R2xVanF3ODR3bHBkVW94cmY=
  AWS_SECRET_ACCESS_KEY:
Wi9kcFluSWxHRzlWaFlzNk1hc0xma2JXcjM1MVhqa051SlBleXpmOQ==
kind: Secret
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  ownerReferences:
  - apiVersion: objectbucket.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ObjectBucketClaim
    name: test21obc
    uid: 64f04cba-f662-11e9-bc3c-0295250841af
```

```
  resourceVersion: "40751"
  selfLink: /api/v1/namespaces/openshift-storage/secrets/test21obc
  uid: 65117c1c-f662-11e9-9094-0a5305de57bb
type: Opaque
```

シークレットは S3 アクセス認証情報を提供します。

5. 以下のコマンドを実行して設定マップを表示します。

```
# oc get -n openshift-storage cm test21obc -o yaml
```

以下に例を示します。

```
apiVersion: v1
data:
  BUCKET_HOST: 10.0.171.35
  BUCKET_NAME: test21obc-933348a6-e267-4f82-82f1-e59bf4fe3bb4
  BUCKET_PORT: "31242"
  BUCKET_REGION: ""
  BUCKET_SUBREGION: ""
kind: ConfigMap
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  ownerReferences:
  - apiVersion: objectbucket.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ObjectBucketClaim
    name: test21obc
    uid: 64f04cba-f662-11e9-bc3c-0295250841af
  resourceVersion: "40752"
  selfLink: /api/v1/namespaces/openshift-storage/configmaps/test21obc
  uid: 651c6501-f662-11e9-9094-0a5305de57bb
```

設定マップには、アプリケーションの S3 エンドポイント情報が含まれます。

10.7.3. OpenShift Web コンソールを使用した Object Bucket Claim(オブジェクトバケット要求) の作成

OpenShift Web コンソールを使用して Object Bucket Claim (オブジェクトバケット要求) を作成できます。

前提条件

- OpenShift Web コンソールへの管理者アクセス。

- アプリケーションが OBC と通信できるようにするには、configmap およびシークレットを使用する必要があります。これに関する詳細情報は、「動的 Object Bucket Claim(オブジェクトバケット要求)」 を参照してください。

手順

1. OpenShift Web コンソールにログインします。
2. 左側のナビゲーションバーで **Storage → Object Bucket Claims → Create Object Bucket Claim** をクリックします。
   a. Object Bucket Claim(オブジェクトバケット要求) の名前を入力し、ドロップダウンメニューから、内部または外部かのデプロイメントに応じて適切なストレージクラスとバケットクラスを選択します。

   内部モード

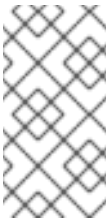
   デプロイメント後に作成された以下のストレージクラスを使用できます。

   - **ocs-storagecluster-ceph-rgw** は Ceph Object Gateway (RGW) を使用します。
   - **openshift-storage.noobaa.io** は Multicloud Object Gateway (MCG) を使用します。

   外部モード

   デプロイメント後に作成された以下のストレージクラスを使用できます。

   - **ocs-external-storagecluster-ceph-rgw** は RGW を使用します。
   - **openshift-storage.noobaa.io** は MCG を使用します。

   注記

   RGW OBC ストレージクラスは、OpenShift Data Foundation バージョン 4.5 の新規インストールでのみ利用できます。これは、以前の OpenShift Data Foundation リリースからアップグレードされたクラスターには適用されません。

   b. **Create** をクリックします。
   OBC を作成すると、その詳細ページにリダイレクトされます。

10.7.4. Object Bucket Claim(オブジェクトバケット要求) のデプロイメントへの割り当て

Object Bucket Claim(オブジェクトバケット要求、OBC) は作成後に、特定のデプロイメントに割り当てることができます。

前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

1. 左側のナビゲーションバーで **Storage → Object Bucket Claims** をクリックします。

2. 作成した OBC の横にあるアクションメニュー (⋮) をクリックします。

    a. ドロップダウンメニューで、**Attach to Deployment** を選択します。

    b. Deployment Name 一覧から必要なデプロイメントを選択し、**Attach** をクリックします。

10.7.5. OpenShift Web コンソールを使用したオブジェクトバケットの表示

OpenShift Web コンソールを使用して、Object Bucket Claim(オブジェクトバケット要求、OBC) 用に作成されたオブジェクトバケットの詳細を表示できます。

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 コンソールにログインします。
2. 左側のナビゲーションバーで **Storage → Object Buckets** をクリックします。
   オプション: 特定の OBC の詳細ページに移動し、**Resource** リンクをクリックして、その OBC のオブジェクトバケットを表示することもできます。
3. 詳細を表示するオブジェクトバケットを選択します。選択すると、**Object Bucket Details** ページに移動します。

10.7.6. Object Bucket Claim(オブジェクトバケット要求) の削除

前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

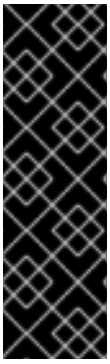
1. 左側のナビゲーションバーで **Storage → Object Bucket Claims** をクリックします。
2. 削除する Object Bucket Claim(オブジェクトバケット要求) の横にあるアクションメニュー (⋮) をクリックします。

    a. **Delete Object Bucket Claim** を選択します。

    b. **Delete** をクリックします。

10.8. オブジェクトバケットのキャッシュポリシー

キャッシュバケットは、ハブのターゲットとキャッシュターゲットが指定された namespace バケットです。ハブのターゲットは、S3 と互換性のある大規模なオブジェクトストレージバケットです。
キャッシュのバケットは、ローカルの Multicloud Object Gateway バケットです。AWS バケットまたは IBM COS バケットをキャッシュするキャッシュバケットを作成できます。



重要

CPU バケット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。

- AWS S3
- IBM COS

10.8.1. AWS キャッシュバケットの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

または、MCG パッケージを、Download Red Hat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。

注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

1. NamespaceStore リソースを作成します。NamespaceStore は、MCG namespace バケットでデータの読み取りおよび書き込みターゲットとして使用される基礎となるストレージを表します。MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

```
noobaa namespacestore create aws-s3 <namespacestore> --access-key <AWS ACCESS KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name>
```

   a. **<namespacestore>** を namespacestore の名前に置き換えます。

   b. **<AWS ACCESS KEY>** および **<AWS SECRET ACCESS KEY>** を、作成した AWS アクセスキー ID およびシークレットアクセスキーに置き換えます。

c. **<bucket-name>** を既存の AWS バケット名に置き換えます。この引数は、MCG に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。
YAML を適用してストレージリソースを追加することもできます。まず、認証情報を使用してシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN
BASE64>
```

Base64 を使用して独自の AWS アクセスキー ID およびシークレットアクセスキーを指定し、エンコードし、その結果を **<AWS ACCESS KEY ID ENCODED IN BASE64>** および **<AWS SECRET ACCESS KEY ENCODED IN BASE64>** に使用する必要があります。

<namespacestore-secret-name> を一意の名前に置き換えます。

次に、以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <namespacestore>
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    targetBucket: <target-bucket>
  type: aws-s3
```

d. **<namespacestore>** を一意の名前に置き換えます。

e. **<namespacestore-secret-name>** を、直前の手順で作成されたシークレットに置き換えます。

f. **<namespace-secret>** を、直前の手順でシークレットを作成するために使用された namespace に置き換えます。

g. **<target-bucket>** を namespacestore 用に作成した AWS S3 バケットに置き換えます。

2. 以下のコマンドを実行してバケットクラスを作成します。

```
noobaa bucketclass create namespace-bucketclass cache <my-cache-bucket-class> --
backingstores <backing-store> --hub-resource <namespacestore>
```

a. **<my-cache-bucket-class>** を一意のバケットクラス名に置き換えます。

b. **<backing-store>** を関連するバッキングストアに置き換えます。コンマで区切られた 1 つ以上のバッキングストアを一覧表示できます。

c. **<namespacestore>** を、直前の手順で作成された namespacestore に置き換えます。

3. 以下のコマンドを実行して、手順 2 に定義されたバケットクラスを使用する Object Bucket Claim (OBC) リソースを使用してバケットを作成します。

```
noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
```

a. **<my-bucket-claim>** を一意の名前に置き換えます。

b. **<custom-bucket-class>** を、手順 2 で作成したバケットクラスの名前に置き換えます。

10.8.2. IBM COS キャッシュバケットの作成

前提条件

- Multicloud Object Gateway (MCG) コマンドラインインターフェイスをダウンロードします。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

注記

サブスクリプションマネージャーを使用してリポジトリーを有効にするための適切なアーキテクチャーを指定します。

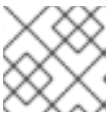
- IBM Power の場合は、次のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-ppc64le-rpms
```

- IBM Z インフラストラクチャーの場合は、以下のコマンドを使用します。

```
# subscription-manager repos --enable=rh-odf-4-for-rhel-8-s390x-rpms
```

または、MCG パッケージを、Download Red Hat OpenShift Data Foundation ページにある OpenShift Data Foundation RPM からインストールできます。



注記

お使いのアーキテクチャーに応じて、正しい製品バリアントを選択します。

手順

1. NamespaceStore リソースを作成します。NamespaceStore は、MCG namespace バケットでデータの読み取りおよび書き込みターゲットとして使用される基礎となるストレージを表します。MCG コマンドラインインターフェイスから、以下のコマンドを実行します。

```
noobaa namespacestore create ibm-cos <namespacestore> --endpoint <IBM COS
ENDPOINT> --access-key <IBM ACCESS KEY> --secret-key <IBM SECRET ACCESS
KEY> --target-bucket <bucket-name>
```

a. **<namespacestore>** を NamespaceStore の名前に置き換えます。

b. **<IBM ACCESS KEY>**, **<IBM SECRET ACCESS KEY>**, **<IBM COS ENDPOINT>** を IBM アクセスキー ID、シークレットアクセスキー、および既存の IBM バケットの場所に対応する地域のエンドポイントに置き換えます。

c. **<bucket-name>** を既存の IBM バケット名に置き換えます。この引数は、MCG に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。
YAML を適用してストレージリソースを追加することもできます。まず、認証情報を使用してシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED
IN BASE64>
```

Base64 を使用して独自の IBM COS アクセスキー ID およびシークレットアクセスキーを指定し、エンコードし、その結果を **<IBM COS ACCESS KEY ID ENCODED IN BASE64>** および **<IBM COS SECRET ACCESS KEY ENCODED IN BASE64>** に使用する必要があります。

<namespacestore-secret-name> を一意の名前に置き換えます。

次に、以下の YAML を適用します。

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <namespacestore>
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
    secret:
      name: <backingstore-secret-name>
      namespace: <namespace-secret>
    signatureVersion: v2
    targetBucket: <target-bucket>
  type: ibm-cos
```

d. **<namespacestore>** を一意の名前に置き換えます。

e. **<IBM COS ENDPOINT>** を適切な IBM COS エンドポイントに置き換えます。

f. **<backingstore-secret-name>** を、直前の手順で作成されたシークレットに置き換えます。

g. **<namespace-secret>** を、直前の手順でシークレットを作成するために使用された namespace に置き換えます。

h. **<target-bucket>** を namespacestore 用に作成した AWS S3 バケットに置き換えます。

2. 以下のコマンドを実行してバケットクラスを作成します。

```
noobaa bucketclass create namespace-bucketclass cache <my-bucket-class> --
backingstores <backing-store> --hubResource <namespacestore>
```

a. **<my-bucket-class>** を一意のバケットクラス名に置き換えます。

b. **<backing-store>** を関連するバッキングストアに置き換えます。コンマで区切られた 1 つ以上のバッキングストアを一覧表示できます。

c. **<namespacestore>** を、直前の手順で作成された namespacestore に置き換えます。

3. 以下のコマンドを実行して、手順 2 に定義されたバケットクラスを使用する Object Bucket Class リソースを使用してバケットを作成します。

```
noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
```

a. **<my-bucket-claim>** を一意の名前に置き換えます。

b. **<custom-bucket-class>** を、手順 2 で作成したバケットクラスの名前に置き換えます。

10.9. エンドポイントの追加による MULTICLOUD OBJECT GATEWAY パフォーマンスのスケーリング

Multicloud Object Gateway のパフォーマンスは環境によって異なる場合があります。特定のアプリケーションでは、高速なパフォーマンスを必要とする場合があり、これは S3 エンドポイントをスケーリングして簡単に対応できます。

Multicloud Object Gateway リソースプールは、デフォルトで有効にされる 2 種類のサービスを提供する NooBaa デーモンコンテナーのグループです。

- ストレージサービス
- S3 エンドポイントサービス

10.9.1. ストレージノードを使用した Multicloud Object Gateway のスケーリング

前提条件

- Multicloud Object Gateway (MCG) にアクセスできる OpenShift Container Platform で実行中の OpenShift Data Foundation クラスター。

MCG のストレージノードは 1 つ以上の永続ボリューム (PV) に割り当てられた NooBaa デーモンコンテナーであり、ローカルオブジェクトサービスデータストレージに使用されます。NooBaa デーモンは Kubernetes ノードにデプロイできます。これは、StatefulSet Pod で設定される Kubernetes プールを作成して実行できます。

手順

1. **OpenShift Web Console** にログインします。
2. MCG ユーザーインターフェイスから **Overview → Add Storage Resources** をクリックします。
3. ウィンドウから **Deploy Kubernetes Pool** をクリックします。
4. **Create Pool** 手順で、今後インストールされるノードのターゲットプールを作成します。
5. **Configure** 手順で、要求される Pod 数と各 PV のサイズを設定します。新規 Pod ごとに、1 つの PV が作成されます。
6. **Review** 手順で、新規プールの詳細を検索し、ローカルまたは外部デプロイメントのいずれかの使用するデプロイメント方法を選択します。ローカルデプロイメントが選択されている場合、Kubernetes ノードはクラスター内にデプロイされます。外部デプロイメントが選択されている場合、外部で実行するための YAML ファイルが提供されます。
7. すべてのノードは最初の手順で選択したプールに割り当てられ、**Resources → Storage resources → Resource name** の下で確認できます。

10.10. MULTICLOUD OBJECT GATEWAY エンドポイントの自動スケーリング

MultiCloud Object Gateway (MCG) の S3 サービスの負荷が増減すると、MCG エンドポイントの数が自動的にスケーリングされます。OpenShift Data Foundation クラスターは、アクティブな MCG エンドポイントを 1 つ使用してデプロイされます。デフォルトでは、MCG エンドポイント Pod はそれぞれ、CPU 1 つ、メモリー要求 2 Gi、要求に一致する制限で設定されます。エンドポイントの CPU 負荷が一貫した期間、使用率 80% のしきい値を超えると、2 番目のエンドポイントがデプロイされ、最初のエンドポイントの負荷を軽減します。両方のエンドポイントの平均 CPU 負荷が、一貫した期間 80% のしきい値を下回ると、エンドポイントの 1 つが削除されます。この機能により、MCG のパフォーマンスおよび保守性が向上します。

第11章 永続ボリューム要求の管理

重要

PVC の拡張は OpenShift Data Foundation がサポートする PVC ではサポートされません。

11.1. 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がある。
- 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 Persistent Volume Claim(永続ボリューム要求、PVC) 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、**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、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 Persistent Vocume Claim (永続ボリューム要求、PVC) に一致する **Type** があることを確認します (例: **myclaim**)。

11.2. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) 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、一覧を絞り込むために Name または Label で PVC の一覧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。

11.3. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) 要求イベントの確認

以下の手順を使用して、Persistent Volume Claim(永続ボリューム要求、PVC) 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。

11.4. 動的プロビジョニング

11.4.1. 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。

11.4.2. 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、CSI ドライバーでは以下のオプションを使用できます。

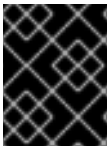
- ボリュームモードが **Block** の Ceph RBD をベースとする PVC (ReadWriteOnce (RWO) および ReadWriteMany (RWX) アクセス) を作成します。
- ボリュームモードが **Filesystem** の Ceph RBD をベースとする PVC (ReadWriteOnce (RWO) アクセス) を作成します。
- ボリュームモードが **Filesystem** の CephFS をベースとする PVC (ReadWriteOnce (RWO) および ReadWriteMany (RWX) アクセス) を作成します。

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。

11.4.3. 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、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。
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ので、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。

同じ永続ボリューム要求 (PVC) の複数のスナップショットを作成できます。CephFS の場合、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、PVC ごとに最大 512 スナップショットを作成できます。

注記

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。

12.1. ボリュームスナップショットの作成

Persistent Volume Claim(永続ボリューム要求、PVC) 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

Pod が使用している場合、OpenShift Data Foundation は PVC のボリュームスナップショットのクラッシュの一貫性だけを提供します。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一覧を表示します。新規スナップショットが一覧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一覧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 Persistent Volume Claim(永続ボリューム要求、PVC) 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。

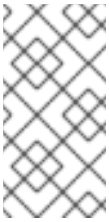
Persistent Volume Claim ページまたは Volume Snapshots ページのいずれかからボリュームスナップショットを復元できます。

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。



注記

Rados Block Device (RBD) の場合、親 PVC と同じプールが指定されるストレージクラスを選択する必要があります。暗号化が有効でないストレージクラスを使用して暗号化された PVC のスナップショットを復元することや、その逆はサポートされていません。

7. 任意の **Access Mode** を選択します。

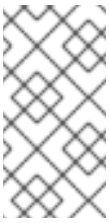
重要

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には、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 について参照してください。

8. オプション: RBD の場合、**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。



注記

Rados Block Device (RBD) の場合、親 PVC と同じプールが指定されるストレージクラスを選択する必要があります。暗号化が有効でないストレージクラスを使用して暗号化された PVC のスナップショットを復元することや、その逆はサポートされていません。

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には、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 について参照してください。

7. オプション: RBD の場合、**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** ページに一覧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一覧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 (PVC) 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

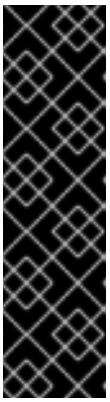
- ソース PVC は **Bound** 状態にある必要があり、使用中の状態にすることはできません。

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には、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 について参照してください。

5. **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。
6. 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 ストレージノードの置き換え

以下のいずれかの手順を選択して、ストレージノードを置き換えることができます。

- 「Google Cloud のインストーラーでプロビジョニングされるインフラストラクチャーで動作するノードの置き換え」
- 「Google Cloud のインストーラーでプロビジョニングされるインフラストラクチャーでの失敗したノードの置き換え」

14.1. GOOGLE CLOUD のインストーラーでプロビジョニングされるインフラストラクチャーで動作するノードの置き換え

以下の手順を使用して、Google Cloud のインストーラーでプロビジョニングされるインフラストラクチャー (IPI) で動作するノードを置き換えます。

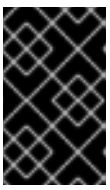
手順

1. OpenShift Web コンソールにログインし、**Compute → Nodes** をクリックします。
2. 置き換える必要のあるノードを特定します。その **マシン名** をメモします。
3. 以下のコマンドを実行して、ノードにスケジュール対象外 (unschedulable) のマークを付けます。

```
$ oc adm cordon <node_name>
```

4. 以下のコマンドを使用してノードをドレイン (解放) します。

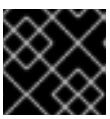
```
$ oc adm drain <node_name> --force --delete-emptydir-data=true --ignore-daemonsets
```



重要

このアクティビティーには少なくとも 5-10 分以上かかる場合があります。この期間に生成される Ceph のエラーは一時的なもので、新規ノードにラベルが付けられ、これが機能すると自動的に解決されます。

5. **Compute → Machines** をクリックします。必要なマシンを検索します。
6. 必要なマシンの横にある **Action menu (⋮) → Delete Machine** をクリックします。
7. **Delete** をクリックしてマシンの削除を確認します。新しいマシンが自動的に作成されます。
8. 新規マシンが起動し、**Running** 状態に移行するまで待機します。



重要

このアクティビティーには少なくとも 5-10 分以上かかる場合があります。

9. **Compute → Nodes** をクリックし、新規ノードが **Ready** 状態にあることを確認します。
10. 以下のいずれかを使用して、OpenShift Data Foundation ラベルを新規ノードに適用します。

ユーザーインターフェイスを使用する場合

a. 新規ノードについて、**Action Menu (⋮) → Edit Labels** をクリックします。

b. **cluster.ocs.openshift.io/openshift-storage** を追加し、**Save** をクリックします。

コマンドラインインターフェイスの使用

- 以下のコマンドを実行して、OpenShift Data Foundation ラベルを新規ノードに適用します。

```
$ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
```

検証手順

1. 以下のコマンドを実行して、出力で新規ノードが表示されていることを確認します。

```
$ oc get nodes --show-labels | grep cluster.ocs.openshift.io/openshift-storage= |cut -d' ' -f1
```

2. **Workloads → Pods** をクリックし、新規ノード上の少なくとも以下の Pod が **Running** 状態にあることを確認します。
   - **csi-cephfsplugin-***
   - **csi-rbdplugin-***
3. 他の必要なすべての OpenShift Data Foundation Pod が **Running** 状態にあることを確認します。
4. 新規 OSD Pod が交換後のノードで実行されていることを確認します。

```
$ oc get pods -o wide -n openshift-storage| egrep -i new-node-name | egrep osd
```

5. (オプション) クラスターでクラスター全体の暗号化が有効な場合は、新規 OSD デバイスが暗号化されていることを確認します。
   直前の手順で特定された新規ノードごとに、以下を実行します。
   a. デバッグ Pod を作成し、選択したホストの chroot 環境を開きます。

```
$ oc debug node/<node name>
$ chroot /host
```

   b. lsblk を実行し、**ocs-deviceset** 名の横にある crypt キーワードを確認します。

```
$ lsblk
```

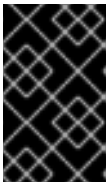
6. 検証手順が失敗した場合は、Red Hat サポート にお問い合わせください。

14.2. GOOGLE CLOUD のインストーラーでプロビジョニングされるインフラストラクチャーでの失敗したノードの置き換え

以下の手順に従って、OpenShift Data Foundation の Google Cloud のインストーラーでプロビジョニングされるインフラストラクチャー (IPI) で動作しない障害のあるノードを置き換えます。

手順

1. OpenShift Web コンソールにログインし、**Compute → Nodes** をクリックします。
2. 障害のあるノードを特定し、その **Machine Name** をクリックします。
3. **Actions → Edit Annotations** をクリックし、**Add More** をクリックします。
4. **machine.openshift.io/exclude-node-draining** を追加し、**Save** をクリックします。
5. **Actions → Delete Machine** をクリックしてから、**Delete** をクリックします。
6. 新しいマシンが自動的に作成されます。新規マシンが起動するのを待機します。



重要

このアクティビティーには少なくとも 5-10 分以上かかる場合があります。この期間に生成される Ceph のエラーは一時的なもので、新規ノードにラベルが付けられ、これが機能すると自動的に解決されます。

7. **Compute → Nodes** をクリックし、新規ノードが **Ready** 状態にあることを確認します。
8. 以下のいずれかを使用して、OpenShift Data Foundation ラベルを新規ノードに適用します。

   Web ユーザーインターフェイスの使用

   a. 新規ノードについて、**Action Menu (⋮) → Edit Labels** をクリックします。

   b. **cluster.ocs.openshift.io/openshift-storage** を追加し、**Save** をクリックします。

   コマンドラインインターフェイスの使用

   - 以下のコマンドを実行して、OpenShift Data Foundation ラベルを新規ノードに適用します。

     ```
     $ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
     ```

9. [オプション]: 失敗した Google Cloud インスタンスが自動的に削除されない場合、インスタンスを Google Cloud コンソールで終了します。

検証手順

1. 以下のコマンドを実行して、出力で新規ノードが表示されていることを確認します。

   ```
   $ oc get nodes --show-labels | grep cluster.ocs.openshift.io/openshift-storage= |cut -d' ' -f1
   ```

2. **Workloads → Pods** をクリックし、新規ノード上の少なくとも以下の Pod が **Running** 状態にあることを確認します。

   - **csi-cephfsplugin-***

- **csi-rbdplugin-***

3. 他の必要なすべての OpenShift Data Foundation Pod が **Running** 状態にあることを確認します。

4. 新規 OSD Pod が交換後のノードで実行されていることを確認します。

```
$ oc get pods -o wide -n openshift-storage| egrep -i new-node-name | egrep osd
```

5. (オプション) クラスターでクラスター全体の暗号化が有効な場合は、新規 OSD デバイスが暗号化されていることを確認します。
直前の手順で特定された新規ノードごとに、以下を実行します。

   a. デバッグ Pod を作成し、選択したホストの chroot 環境を開きます。

```
$ oc debug node/<node name>
$ chroot /host
```

   b. lsblk を実行し、**ocs-deviceset** 名の横にある crypt キーワードを確認します。

```
$ lsblk
```

6. 検証手順が失敗した場合は、Red Hat サポート にお問い合わせください。

第15章 ストレージデバイスの置き換え

15.1. GOOGLE CLOUD のインストーラーでプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え

Google Cloud のインストーラーでプロビジョニングされるインフラストラクチャーの動的に作成されたストレージクラスターのデバイスを置き換える必要がある場合は、ストレージノードを置き換える必要があります。ノードを置き換える方法は、以下を参照してください。

- Replacing operational nodes on Google Cloud installer-provisioned infrastructure
- Google Cloud のインストーラーでプロビジョニングされるインフラストラクチャーでの失敗したノードの置き換え

第16章 OPENSHIFT DATA FOUNDATION へのアップグレード

16.1. OPENSHIFT DATA FOUNDATION 更新プロセスの概要

この章では、すべての Red Hat OpenShift Data Foundation デプロイメント (Internal、Internal-Attached、および External) のマイナーリリースおよび z-stream リリース間でアップグレードする方法を説明します。アップグレードプロセスは、すべてのデプロイメントで引き続き同じとなります。

OpenShift Data Foundation とそのコンポーネントは、4.11 と 4.12 のようなマイナーリリース間、または 4.12.0 と 4.12.1 のような z-stream 間更新で、自動更新を有効にするか (Operator のインストール時に行っていない場合)、手動で更新を行うことでアップグレードできます。z-stream の新規リリースが利用可能になると、更新ストラテジーが Automatic に設定されている場合、アップグレードプロセスが自動的にトリガーされます。

また、内部および外部モードのデプロイメントの両方で、以下の順序で Red Hat OpenShift Data Foundation のさまざまな部分をアップグレードする必要もあります。

1. OpenShift Container Platform の クラスターの更新 ドキュメントに従って **OpenShift Container Platform を更新します。**
2. **Red Hat OpenShift Data Foundation を更新します。**
    a. **更新で使用する非接続環境を準備する** には、Operator Lifecycle Manager を制限されたネットワークで使用するための Operator ガイドを参照し、OpenShift Data Foundation およびローカルストレージ Operator を使用している場合はこれらを更新できるようにします。
    b. **マイナーリリース間の更新** は、Red Hat OpenShift Data Foundation 4.11 から 4.12 への更新を参照してください。
    c. **z-stream リリース間の更新** は、Red Hat OpenShift Data Foundation 4.12.x の 4.12.y への更新 を参照してください。
    d. **外部モードのデプロイメントを更新する場合は、**Red Hat OpenShift Data Foundation 外部シークレットの更新 のセクションにある手順も実行する必要があります。
    e. **ローカルストレージ演算子を使用している場合は、Local Storage operator を更新します。**不明な場合は、ローカルストレージ Operator デプロイメントの確認 を参照してください。

更新に関する考慮事項

開始する前に、以下の重要な考慮事項を確認してください。

- Red Hat Open Shift Container Platform のバージョンは、Red Hat Open Shift Data Foundation と同じです。
  OpenShift Container Platform および Red Hat OpenShift Data Foundation のサポートされる組み合わせについての詳細は、Interoperability Matrix を参照してください。
- クラスターが内部モードまたは外部モードのどちらでデプロイされたかを確認するには、**ODF クラスターに内部モードまたは外部モードのストレージがあるかどうかを判別する方法** に関する ナレッジベースの記事 を参照してください。
- ローカルストレージ Operator は、ローカルストレージ Operator のバージョンが Red Hat OpenShift Container Platform のバージョンと一致する場合にのみ完全にサポートされます。

- フレキシブルスケーリング機能は、OpenShift Data Foundation の新しいデプロイメントでのみ利用できます。詳細は、Scaling storage guide を参照してください。

16.2. RED HAT OPENSHIFT DATA FOUNDATION 4.11 から 4.12 への更新

この章では、すべての Red Hat OpenShift Data Foundation デプロイメント (Internal、Internal-Attached、および External) のマイナーリリース間でアップグレードする方法を説明します。アップグレードプロセスは、すべてのデプロイメントで引き続き同じとなります。唯一の違いは、アップグレードされるものとアップグレードされないものがあることです。

- Internal および Internal-attached のデプロイメントの場合、OpenShift Data Foundation をアップグレードすると、バックエンド Ceph Storage クラスターを含むすべての OpenShift Data Foundation サービスがアップグレードされます。

- 外部モードのデプロイメントの場合、OpenShift Data Foundation をアップグレードすると、OpenShift Data Foundation サービスのみがアップグレードされ、バックエンド Ceph ストレージクラスターは変更されないままとなり、個別にアップグレードする必要があります。
新機能のサポート、セキュリティー修正、およびその他のバグ修正を取得するために、RHCS を OpenShift Data Foundation と共にアップグレードすることが推奨されます。RHCS アップグレードに強く依存していないため、最初に OpenShift Data Foundation Operator をアップグレードしてから、RHCS をアップグレードするか、その逆を行うことができます。Red Hat Ceph Storage リリースの詳細は、solution を参照してください。

重要

4.11 よりも古いバージョンから 4.12 への直接のアップグレードはサポートされていません。

前提条件

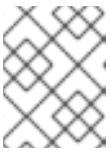
- **OpenShift Container Platform** クラスターがバージョン 4.12.X の最新の安定したリリースに更新されていることを確認します。クラスターの更新 を参照してください。

- **OpenShift Data Foundation** クラスターが正常であり、データに回復性があることを確認する。

  - **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** タブおよび **Object** タブのステータスカードの緑色のチェックマークを確認します。緑色のチェックマークは、**ストレージクラスター**、**オブジェクトサービス**、および **データ回復性** がすべて正常であることを示します。

- Operator Pod を含むすべての **OpenShift Data Foundation** Pod が **openshift-storage** namespace で **Running** 状態にあることを確認する。
Pod の状態を表示するには、OpenShift Web コンソールで **Workloads → Pods**をクリックします。**Project** ドロップダウンリストから **openshift-storage** を選択します。



注記

Show default projectsオプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

- 更新時間はクラスターで実行される OSD の数によって異なるため、OpenShift Data Foundation 更新プロセスを完了するのに十分な時間を確保してください。

手順

1. OpenShift Web コンソールで、**Operators → Installed Operators** に移動します。
2. **openshift-storage** プロジェクトを選択します。
3. OpenShift Data Foundation Operator 名をクリックします。
4. **Subscription** タブをクリックしてから、**Update Channel** の下にあるリンクをクリックします。
5. **Stable-4.12** 更新チャンネルを選択して、**保存** します。
6. **Upgrade status** に **requires approval** が表示される場合は、**requires approval** をクリックします。
   a. Install Plan Details ページで、**Preview Install Plan** をクリックします。
   b. インストール計画を確認し、**Approve** をクリックします。
   Status が **Unknown** から **Created** に変更されるまで待機します。
7. **Operators → Installed Operators** に移動します。
8. **openshift-storage** プロジェクトを選択します。
   OpenShift Data Foundation Operator の **Status** が**Up to date** に変わるのを待ちます。

検証手順

- OpenShift Data Foundation 名の下にある **Version** を確認し、演算子の状態を確認します。
  - **Operators → Installed Operators** に移動し、**openshift-storage** プロジェクトを選択します。
  - アップグレードが完了すると、バージョンは OpenShift Data Foundation の新規バージョン番号に更新され、ステータスは緑色のチェックマークが付いて **Succeeded** に変わります。
- **OpenShift Data Foundation** クラスターが正常であること、およびデータに回復性があることを確認します。
  - **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** および **Object** タブのステータスカードの緑色のチェックマークを確認します。緑色のチェックマークは、ストレージクラスター、オブジェクトサービス、およびデータの回復性が正常であることを示します。
- 検証手順が失敗した場合は、Red Hat サポート にお問い合わせください。

重要

外部モードのデプロイメントを更新したら、外部シークレットも更新する必要があります。手順については、OpenShift Data Foundation 外部シークレットの更新 を参照してください。

関連情報

OpenShift Data Foundation の更新中に問題が発生した場合は、トラブルシューティングガイド の **トラブルシューティングに共通して必要になるログ** セクションを参照してください。

16.3. RED HAT OPENSHIFT DATA FOUNDATION 4.12.X の 4.12.Y への更新

この章では、すべての Red Hat OpenShift Data Foundation デプロイメント (Internal、Internal-Attached、および External) の z-stream リリース間でアップグレードする方法を説明します。アップグレードプロセスは、すべてのデプロイメントで引き続き同じとなります。唯一の違いは、アップグレードされるものとアップグレードされないものがあることです。

- Internal および Internal-attached のデプロイメントの場合、OpenShift Data Foundation をアップグレードすると、バックエンド Ceph Storage クラスターを含むすべての OpenShift Data Foundation サービスがアップグレードされます。

- 外部モードのデプロイメントの場合、OpenShift Data Foundation をアップグレードすると、OpenShift Data Foundation サービスのみがアップグレードされ、バックエンド Ceph ストレージクラスターは変更されないままとなり、個別にアップグレードする必要があります。
  したがって、新機能のサポート、セキュリティー修正、およびその他のバグ修正を取得するために、RHCS を OpenShift Data Foundation と共にアップグレードすることが推奨されます。RHCS アップグレードに強く依存していないため、最初に OpenShift Data Foundation Operator をアップグレードしてから、RHCS をアップグレードするか、その逆を行うことができます。Red Hat Ceph Storage リリースの詳細は、solution を参照してください。

z-stream の新規リリースが利用可能になると、更新ストラテジーが **Automatic** に設定されている場合、アップグレードプロセスが自動的にトリガーされます。更新ストラテジーが **Manual** に設定されている場合には、以下の手順を使用します。

前提条件

- **OpenShift Container Platform** クラスターがバージョン 4.12.X の最新の安定したリリースに更新されていることを確認します。クラスターの更新 を参照してください。

- **OpenShift Data Foundation** クラスターが正常であり、データに回復性があることを確認する。

  - **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** および **Object** タブのステータスカードの緑色のチェックマークを確認します。緑色のチェックマークは、ストレージクラスター、オブジェクトサービス、およびデータの回復性が正常であることを示します。

- Operator Pod を含むすべての **OpenShift Data Foundation** Pod が **openshift-storage** namespace で **Running** 状態にあることを確認する。
  Pod の状態を表示するには、OpenShift Web コンソールで **Workloads → Pods**をクリックします。**Project** ドロップダウンリストから **openshift-storage** を選択します。

  注記

  Show default projectsオプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

- 更新時間はクラスターで実行される OSD の数によって異なるため、OpenShift Data Foundation 更新プロセスを完了するのに十分な時間を確保してください。

手順

1. OpenShift Web コンソールで、**Operators → Installed Operators** に移動します。
2. **openshift-storage** プロジェクトを選択します。

注記

Show default projects オプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

3. **OpenShift Data Foundation** Operator 名をクリックします。
4. **Subscription** タブをクリックします。
5. **Upgrade Status** に **require approval** が表示される場合は、**requires approval** リンクをクリックします。
6. **InstallPlan Details** ページで、**Preview Install Plan** をクリックします。
7. インストール計画を確認し、**Approve** をクリックします。
8. Status が **Unknown** から **Created** に変更されるまで待機します。

検証手順

- OpenShift Data Foundation 名の下にある **Version** を確認し、演算子の状態を確認します。
  - **Operators → Installed Operators** に移動し、**openshift-storage** プロジェクトを選択します。
  - アップグレードが完了すると、バージョンは OpenShift Data Foundation の新規バージョン番号に更新され、ステータスは緑色のチェックマークが付いて **Succeeded** に変わります。
- OpenShift Data Foundation クラスターが正常であること、およびデータに回復性があることを確認します。
  - **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** および **Object** タブのステータスカードの緑色のチェックマークを確認します。緑色のチェックマークは、ストレージクラスター、オブジェクトサービス、およびデータ回復性が正常であることを示します。
- 検証手順が失敗した場合は、Red Hat サポート にお問い合わせください。

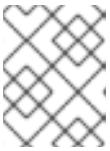
16.4. 更新承認ストラテジーの変更

同じチャネルで新しい更新が利用可能になったときにストレージシステムが自動的に更新されるようにするには、更新承認ストラテジーを **Automatic** のままにしておくことをお勧めします。更新承認ストラテジーを **Manual** に変更すると、アップグレードごとに手動承認が必要になります。

手順

1. **Operators → Installed Operators** に移動します。

2. Project ドロップダウンリストから **openshift-storage** を選択します。



注記

Show default projectsオプションが無効になっている場合は、切り替えボタンを使用して、すべてのデフォルトプロジェクトを一覧表示します。

3. **OpenShift Data Foundation** Operator 名をクリックします。
4. **Subscription** タブに移動します。
5. **Update approval** を変更するには、**鉛筆** アイコンをクリックします。
6. 更新承認ストラテジーを選択し、**Save** をクリックします。

検証手順

- 更新承認で、その下に新しく選択された承認ストラテジーが表示されていることを確認します。