



Red Hat OpenShift Container Storage 4.7

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

インストールおよび管理方法

Red Hat OpenShift Container Storage 4.7 Google Cloud を使用した OpenShift Container Storage のデプロイおよび管理

インストールおよび管理方法

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Deploying_and_managing_OpenShift_Container_Storage_using_Google_Cloud.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

Google Cloud で Red Hat OpenShift Container Storage をインストールし、管理する方法については、本書をお読みください。 Deploying and managing OpenShift Container Storage 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	5
RED HAT ドキュメントへのフィードバック (英語のみ)	6
はじめに	7
第1章 OPENSIFT CONTAINER STORAGE のデプロイの準備	8
1.1. VAULT でのキー値のバックエンドパスおよびポリシーの有効化	8
第2章 GOOGLE CLOUD での OPENSIFT CONTAINER STORAGE のデプロイ	10
2.1. RED HAT OPENSIFT CONTAINER STORAGE OPERATOR のインストール	10
2.2. 内部モードでの OPENSIFT CONTAINER STORAGE CLUSTER SERVICE の作成	11
第3章 OPENSIFT CONTAINER STORAGE デプロイメントの検証	15
3.1. POD の状態の確認	15
3.2. OPENSIFT CONTAINER STORAGE クラスターが正常であることの確認	16
3.3. MULTICLOUD OBJECT GATEWAY が正常であることの確認	17
3.4. OPENSIFT CONTAINER STORAGE 固有のストレージクラスが存在することの確認	18
第4章 OPENSIFT CONTAINER STORAGE のアンインストール	19
4.1. 内部モードでの OPENSIFT CONTAINER STORAGE のアンインストール	19
4.2. OPENSIFT CONTAINER STORAGE からのモニタリングスタックの削除	25
4.3. OPENSIFT CONTAINER STORAGE からの OPENSIFT CONTAINER PLATFORM レジストリーの削除	28
4.4. OPENSIFT CONTAINER STORAGE からのクラスターロギング OPERATOR の削除	29
第5章 ストレージクラスおよびストレージプール	31
5.1. ストレージクラスおよびプールの作成	31
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	32
第6章 OPENSIFT CONTAINER PLATFORM サービスのストレージの設定	36
6.1. OPENSIFT CONTAINER STORAGE を使用するためのイメージレジストリーの設定	36
6.2. OPENSIFT CONTAINER STORAGE を使用するためのモニタリングの設定	38
6.3. OPENSIFT CONTAINER STORAGE のクラスターロギング	41
6.3.1. 永続ストレージの設定	41
6.3.2. OpenShift Container Storage を使用するためのクラスターロギングの設定	42
第7章 OPENSIFT CONTAINER STORAGE を使用した OPENSIFT CONTAINER PLATFORM アプリケーションのサポート	46
第8章 RED HAT OPENSIFT CONTAINER STORAGE に専用のワーカーノードを使用する方法	48
8.1. インフラストラクチャーノードの仕組み	48
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	48
8.3. インフラストラクチャーノードの手動作成	49
第9章 ストレージノードのスケールリング	51
9.1. ストレージノードのスケールリングの要件	51
9.2. GOOGLE CLOUD インフラストラクチャーの OPENSIFT CONTAINER STORAGE 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	51
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	53
9.3.1. 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へのノードの追加	53
9.3.2. 新規ノードの追加の確認	54
9.3.3. ストレージ容量のスケールアップ	54

第10章 MULTICLOUD OBJECT GATEWAY	55
10.1. MULTICLOUD OBJECT GATEWAY について	55
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	55
10.2.1. ターミナルから Multicloud Object Gateway へのアクセス	56
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	58
10.3. MULTICLOUD OBJECT GATEWAY コンソールへのユーザーアクセスの許可	60
10.4. 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	61
10.4.1. 新規バックキングストアの作成	61
10.4.2. 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	63
10.4.2.1. AWS でサポートされるバックキングストアの作成	63
10.4.2.2. IBM COS でサポートされるバックキングストアの作成	65
10.4.2.3. Azure でサポートされるバックキングストアの作成	67
10.4.2.4. GCP でサポートされるバックキングストアの作成	68
10.4.2.5. ローカル永続ボリュームでサポートされるバックキングストアの作成	69
10.4.3. s3 と互換性のある Multicloud Object Gateway バックキングストアの作成	71
10.4.4. 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	72
10.4.5. 新規バケットクラスの作成	74
10.4.6. バケットクラスの編集	75
10.4.7. バケットクラスのバックキングストアの編集	75
10.5. NAMESPACE バケットの管理	77
10.5.1. プロバイダー接続の Multicloud Object Gateway への追加	77
10.5.2. Multicloud Object Gateway を使用した namespace リソースの追加	78
10.5.3. Multicloud Object Gateway を使用した namespace バケットへのリソースの追加	79
10.5.4. namespace バケットのオブジェクトの Amazon S3 API エンドポイント	80
10.5.5. Multicloud Object Gateway CLI および YAML を使用した namespace バケットの追加	80
10.5.5.1. YAML を使用した AWS S3 namespace バケットの追加	81
10.5.5.2. YAML を使用した IBM COS namespace バケットの追加	83
10.5.5.3. Multicloud Object Gateway CLI を使用した AWS S3 namespace バケットの追加	85
10.5.5.4. Multicloud Object Gateway CLI を使用した IBM COS namespace バケットの追加	86
10.6. 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	88
10.6.1. 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	88
10.6.2. 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	89
10.6.3. ユーザーインターフェイスを使用したデータミラーリングを行うためのバケットの設定	89
10.7. MULTICLOUD OBJECT GATEWAY のバケットポリシー	90
10.7.1. バケットポリシーについて	90
10.7.2. バケットポリシーの使用	90
10.7.3. Multicloud Object Gateway での AWS S3 ユーザーの作成	92
10.8. OBJECT BUCKET CLAIM(オブジェクトバケット要求)	94
10.8.1. 動的 Object Bucket Claim(オブジェクトバケット要求)	94
10.8.2. 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	97
10.8.3. 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	99
10.8.4. 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	102
10.8.5. 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	103
10.8.6. Object Bucket Claim(オブジェクトバケット要求) の削除	104
10.9. オブジェクトバケットのキャッシュポリシー	105
10.9.1. AWS キャッシュバケットの作成	105
10.9.2. IBM COS キャッシュバケットの作成	107
10.10. 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	109
10.10.1. Multicloud Object Gateway での S3 エンドポイント	109
10.10.2. ストレージノードを使用したスケーリング	109

10.11. MULTICLOUD OBJECT GATEWAY エンドポイントの自動スケーリング	112
第11章 永続ボリューム要求の管理	113
11.1. OPENSIFT CONTAINER PLATFORM を使用するためのアプリケーション POD の設定	113
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	114
11.3. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) 要求イベントの確認	115
11.4. 動的プロビジョニング	115
11.4.1. 動的プロビジョニングについて	115
11.4.2. OpenShift Container Storage の動的プロビジョニング	116
11.4.3. 利用可能な動的プロビジョニングプラグイン	116
第12章 ボリュームスナップショット	118
12.1. ボリュームスナップショットの作成	118
12.2. ボリュームスナップショットの復元	119
12.3. ボリュームスナップショットの削除	121
第13章 ボリュームのクローン作成	122
13.1. クローンの作成	122
第14章 ストレージノードの置き換え	123
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	123
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	124
第15章 ストレージデバイスの置き換え	127
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	127
第16章 OPENSIFT CONTAINER STORAGE の更新	128
16.1. OPENSIFT CONTAINER STORAGE 更新プロセスの概要	128
16.2. 非接続環境での更新の準備	128
16.2.1. ミラーレジストリーの認証情報の追加	129
16.2.2. Red Hat Operator カタログのビルドおよびミラーリング	130
16.2.3. Operator imageContentSourcePolicy を作成します。	131
16.2.4. redhat-operator CatalogSource の更新	131
16.2.5. 更新の継続	132
16.3. 内部モードでの OPENSIFT CONTAINER STORAGE の更新	133
16.3.1. 内部モードでの OpenShift Container Storage Operator の自動更新の有効化	133
16.3.2. 内部モードでの OpenShift Container Storage Operator の手動による更新	135

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、[弊社の CTO である Chris Wright のメッセージ](#) を参照してください。

RED HAT ドキュメントへのフィードバック (英語のみ)

弊社のドキュメントについての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。フィードバックをお寄せいただくには、以下をご確認ください。

- 特定の部分についての簡単なコメントをお寄せいただく場合は、以下をご確認ください。
 1. ドキュメントの表示が **Multi-page HTML** 形式になっていることを確認してください。ドキュメントの右上隅に **Feedback** ボタンがあることを確認してください。
 2. マウスカーソルを使用して、コメントを追加するテキストの部分を強調表示します。
 3. 強調表示されたテキストの下に表示される **Add Feedback** ポップアップをクリックします。
 4. 表示される指示に従ってください。
- より詳細なフィードバックをお寄せいただく場合は、Bugzilla のチケットを作成してください。
 1. [Bugzilla](#) の Web サイトに移動します。
 2. Component (コンポーネント) として **Documentation** を使用します。
 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 Container Storage 4.7 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 Container Storage 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 Container Storage をデプロイするには、[Deploying OpenShift Container Storage のデプロイの準備](#)についての章の要件を確認し、デプロイメントプロセス [Deploying OpenShift Container Storage on Microsoft Azure](#)を実行します。

第1章 OPENSIFT CONTAINER STORAGE のデプロイの準備

ダイナミックストレージデバイスを使用して OpenShift Container Storage 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可能にすることができます。

OpenShift Container Storage 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: 外部鍵管理システム (KMS) を使用してクラスター全体の暗号化を有効にする場合:
 - トークンのあるポリシーが存在し、Vault のキー値のバックエンドパスが有効にされていることを確認します。[enabled the key value backend path and policy in Vault](#) 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. ノードの最小要件 [テクノロジープレビュー]

OpenShift Container Storage 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。

1.1. VAULT でのキー値のバックエンドパスおよびポリシーの有効化

前提条件

- Vault への管理者アクセス。
- 注: 後に変更することはできないため、命名規則に基づいてバックエンド **path** として一意のパス名を選択します。

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 を使用します。

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

Vault KV シークレットエンジン API の場合は、バージョン 2 です。

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

2. 以下のコマンドを使用して、シークレットでの書き込み操作または削除操作の実行をユーザーを制限するポリシーを作成します。

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

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=ocs -format json
```

第2章 GOOGLE CLOUD での OPENSIFT CONTAINER STORAGE 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 (IPI) 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 Container Storage 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できます。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可能にすることができます。



注記

Google Cloud では、内部の OpenShift Container Storage 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 Container Storage のデプロイの準備](#) について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ついて以下の手順を実行してください。

1. [Red Hat OpenShift Container Storage Operator をインストールする](#)
2. [OpenShift Container Storage Cluster Service を作成する](#)

2.1. RED HAT OPENSIFT CONTAINER STORAGE OPERATOR のインストール

Red Hat OpenShift Container Storage 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こと。
- RHOCV クラスターにワーカーノードが少なくとも 3 つある。
- 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](#) を参照してください。



注記

- OpenShift Container Storage のクラスター全体でのデフォルトノードセクターを上書きする必要がある場合は、コマンドラインインターフェイスで以下のコマンドを使用し、**openshift-storage** namespace の空のノードセクターを指定できます (この場合、openshift-storage namespace を作成します)。

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

- ノードに Red Hat OpenShift Container Storage リソースのみがスケジュールされるように、そのノードに **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 Container Storage に専用のワーカーノードを使用する方法](#) の章を参照してください。

手順

1. Web コンソールで **Operators → OperatorHub** ページに移動し、クリックします。
2. スクロールするか、またはキーワードを Filter by keyword ボックスに入力し、OpenShift Container Storage Operator を検索します。
3. OpenShift Container Storage Operator ページで、**Install** をクリックします。
4. **Install Operator** ページで、以下の必須オプションがデフォルトで選択されます。
 - a. Channel を **stable-4.7** 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。
 - e. **Install** をクリックします。

Automatic (自動) 更新を選択している場合、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 (手動) 更新を選択している場合、OLM は更新要求を作成します。クラスター管理者は、Operator が新規バージョンに更新されるように更新要求を手動で承認する必要があります。

検証手順

OpenShift Container Storage 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。

次のステップ

- OpenShift Container Storage クラスターを作成します。
詳細は、[内部モードでの OpenShift Container Storage Cluster Service の作成](#) について参照してください。

2.2. 内部モードでの OPENSIFT CONTAINER STORAGE CLUSTER SERVICE の作成

以下の手順を使用して、OpenShift Container Storage Operator のインストール後に OpenShift Container Storage Cluster Service を作成します。

前提条件

- OpenShift Container Storage は Operator Hub からインストールする必要があります。詳細は、[Operator Hub を使用した OpenShift Container Storage Operator のインストール](#) について参照してください。
- Google Cloud のデフォルトのストレージクラスはハードドライブ (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-storageclass.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 コンソールにログインします。
2. **Operators** → **Installed Operators** をクリックし、インストールされた Operator をすべて表示します。
選択された **Project** が **openshift-storage** であることを確認します。
3. Storage Cluster の **OpenShift Container Storage** > **Create Instance** リンクをクリックします。
4. **Mode** がデフォルトで **Internal** に設定されます。
5. **Select capacity and nodes** で、以下を実行します。
 - a. **Storage Class** を選択します。デフォルトでは **standard** に設定されます。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。
 - b. ドロップダウンリストから **Requested Capacity** を選択します。デフォルトで、これは **2 TiB** に設定されます。ドロップダウンを使用して容量の値を変更できます。



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)。

- c. **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。
複数のアベイラビリティ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 Container Storage 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。
6. (オプション) セキュリティー設定
 - a. **Enable encryption** チェックボックスを選択して、ブロックおよびファイルストレージを暗号化します。
 - b. 1 つまたは両方の **Encryption level** を選択します。
 - **クラスター全体の暗号化** クラスター全体 (ブロックおよびファイル) を暗号化します。

- **Storage class encryption**(ストレージクラスの暗号化): 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。



重要

ストレージクラスの暗号化は、RBD PV でのみ利用可能な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 の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。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。

- c. **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** はデフォルトで **Vault** に設定されます。
 - ii. **Vault Service Name**、Vault サーバーのホスト **Address** ('https://<hostname または ip>'), **Port number** および **Token** を入力します。
 - iii. **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 Container Storage 専用かつ特有のキー値のシークレットパスを **Backend Path** に入力します。
 - B. **TLS Server Name** および **Vault Enterprise Namespace** を入力します。
 - C. それぞれの PEM でエンコードされた証明書ファイルをアップロードして、**CA Certificate**、**Client Certificate** および **Client Private Key** を指定します。
 - D. **Save** をクリックします。
 - d. **Next** をクリックします。
7. 設定の詳細を確認します。設定を変更するには、**Back** をクリックして以前の設定ページに戻ります。
8. **Create** をクリックします。
9. Vault Key/Value (KV) シークレットエンジン API の場合に configmap を編集します。バージョン 2 は、鍵管理システム (KMS) のクラスター全体の暗号化に使用されます。
 - a. OpenShift Web コンソールで **Workloads** → **ConfigMaps** に移動します。
 - b. KMS 接続の詳細を表示するには、**ocs -kms-connection-details** をクリックします。
 - c. configmap を編集します。
 - i. **Action menu**(**:**) → **Edit ConfigMap** をクリックします。
 - ii. **VAULT_BACKEND** パラメーターを **v2** に設定します。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ocs-kms-connection-details
[...]
data:
  KMS_PROVIDER: vault
  KMS_SERVICE_NAME: vault
[...]
  VAULT_BACKEND: v2
[...]
```

iii. **Save** をクリックします。

検証手順

1. ストレージクラスターの詳細ページで、ストレージクラスター名の横に緑色のチェックマークが表示され、クラスターが正常に作成されたことを示します。
2. インストールされたストレージクラスターの最後の **Status** が緑色のチェックマークと共に **Phase: Ready** と表示されていることを確認します。
 - **Operators → Installed Operators → Storage Cluster**のリンクをクリックして、ストレージクラスターのインストールのステータスを表示します。
 - または、Operator **Details** タブで、**Storage Cluster** タブをクリックすると、ステータスを表示できます。
3. OpenShift Container Storage 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、[OpenShift Container Storage インストールの確認](#) を参照してください。

第3章 OPENSIFT CONTAINER STORAGE デプロイメントの検証

このセクションを使用して、OpenShift Container Storage が正常にデプロイされていることを確認します。

3.1. POD の状態の確認

OpenShift Container Storage が正常にデプロイされているかどうかを判別するために、Pod の状態が **Running** であることを確認できます。

手順

1. OpenShift Web コンソールの左側のペインから **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。
各コンポーネントについて予想される Pod 数や、これがノード数によってどのように異なるかについての詳細は、[表3.1「OpenShift Container Storage クラスターに対応する Pod」](#) を参照してください。
3. **Running** および **Completed** タブをクリックして、以下の Pod が実行中および完了状態にあることを確認します。

表3.1 OpenShift Container Storage クラスターに対応する Pod

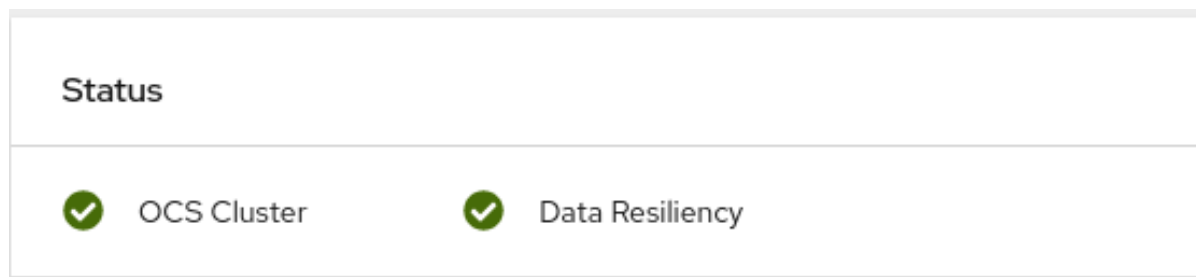
コンポーネント	対応する Pod
OpenShift Container Storage Operator	<ul style="list-style-type: none">● ocs-operator-* (任意のワーカーノードに 1 Pod)● ocs-metrics-exporter-*
Rook-ceph Operator	rook-ceph-operator-* (任意のワーカーノードに 1 Pod)
Multicloud Object Gateway	<ul style="list-style-type: none">● noobaa-operator-* (任意のワーカーノードに 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)

コンポーネント	対応する 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)
CSI	<ul style="list-style-type: none"> ● cephfs <ul style="list-style-type: none"> ○ csi-cephfsplugin-* (各ワーカーノードに 1 Pod) ○ csi-cephfsplugin-provisioner-* (ワーカーノードに分散する 2 Pod) ● rbd <ul style="list-style-type: none"> ○ csi-rbdplugin-* (各ワーカーノードに 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	<ul style="list-style-type: none"> ● rook-ceph-osd-* (各デバイス用に 1 Pod) ● rook-ceph-osd-prepare-ocs-device-* (各デバイス用に 1 Pod)

3.2. OPENSIFT CONTAINER STORAGE クラスターが正常であることの確認

- OpenShift Web コンソールの左側のペインから **Home → Overview** をクリックし、**Persistent Storage** タブをクリックします。
- **Status** カードで、以下のイメージのように **OCS Cluster** および **Data Resiliency** 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。

図3.1 Persistent Storage Overview ダッシュボードの Health status カード



- **Details カード** で、以下のようにクラスター情報が表示されていることを確認します。

サービス名

OpenShift Container Storage

クラスター名

ocs-storagecluster

プロバイダー

GCP

モード

内部

バージョン

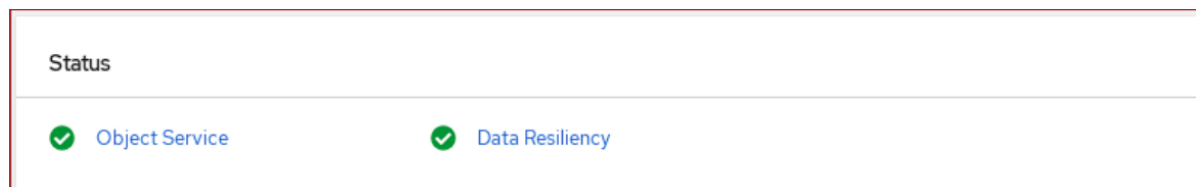
ocs-operator-4.7.0

永続ストレージダッシュボードを使用して OpenShift Container Storage クラスターの正常性に関する詳細は、[OpenShift Container Storage のモニターリング](#) を参照してください。

3.3. MULTICLOUD OBJECT GATEWAY が正常であることの確認

- OpenShift Web コンソールの左側のペインから **Home → Overview** をクリックし、**Object Service** タブをクリックします。
- **Status card** で、**Object Service** と **Data Resiliency** の両方が **Ready** 状態 (緑のチェックマーク) にあることを確認します。

図3.2 Object Service Overview ダッシュボードの Health status カード



- **Details カード** で、MCG 情報が以下のように表示されることを確認します。

サービス名

OpenShift Container Storage

システム名

Multicloud Object Gateway

プロバイダー

GCP

バージョン

ocs-operator-4.7.0

オブジェクトサービスダッシュボードを使用した OpenShift Container Storage クラスターの正常性については、[OpenShift Container Storage のモニターリング](#) を参照してください。

3.4. OPENSIFT CONTAINER STORAGE 固有のストレージクラスが存在することの確認

ストレージクラスがクラスターに存在することを確認するには、以下を実行します。

- 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。
- 以下のストレージクラスが OpenShift Container Storage 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4章 OPENSIFT CONTAINER STORAGE のアンインストール

4.1. 内部モードでの OPENSIFT CONTAINER STORAGE のアンインストール

このセクションの手順に従って OpenShift Container Storage をアンインストールします。

アノテーションのアンインストール

Storage Cluster のアノテーションは、アンインストールプロセスの動作を変更するために使用されます。アンインストールの動作を定義するために、ストレージクラスターに以下の2つのアノテーションが導入されました。

- `uninstall.ocs.openshift.io/cleanup-policy: delete`
- `uninstall.ocs.openshift.io/mode: graceful`

以下の表は、これらのアノテーションで使用できる各種値に関する情報を示しています。

表4.1 `uninstall.ocs.openshift.io` でアノテーションの説明をアンインストールする

Annotation	値	デフォルト	動作
<code>cleanup-policy</code>	<code>delete</code>	はい	Rook は物理ドライブおよび DataDirHostPath をクリーンアップします。
<code>cleanup-policy</code>	<code>Retain</code>	いいえ	Rook は物理ドライブおよび DataDirHostPath をクリーンアップ しません 。
<code>mode</code>	<code>graceful</code>	はい	Rook および NooBaa は PVC および OBC が管理者/ユーザーによって削除されるまでアンインストールプロセスを一時停止します。
<code>mode</code>	<code>forced</code>	いいえ	Rook および NooBaa は、Rook および NooBaa を使用してプロビジョニングされた PVC/OBC がそれぞれ存在している場合でもアンインストールを続行します。

以下のコマンドを使用してアノテーションの値を編集し、クリーンアップポリシーまたはアンインストールモードを変更できます。

```
$ oc annotate storagecluster -n openshift-storage ocs-storagecluster
uninstall.ocs.openshift.io/cleanup-policy="retain" --overwrite
storagecluster.ocs.openshift.io/ocs-storagecluster annotated
```

```
$ oc annotate storagecluster -n openshift-storage ocs-storagecluster
uninstall.ocs.openshift.io/mode="forced" --overwrite
storagecluster.ocs.openshift.io/ocs-storagecluster annotated
```

前提条件

- OpenShift Container Storage クラスターの状態が正常であることを確認します。リソースまたはノードの不足により一部の Pod が正常に終了されないと、アンインストールプロセスに失敗する可能性があります。クラスターが状態が正常でない場合は、OpenShift Container Storage をアンインストールする前に Red Hat カスタマーサポートにお問い合わせください。
- アプリケーションが OpenShift Container Storage によって提供されるストレージクラスを使用して永続ボリューム要求 (PVC) またはオブジェクトバケット要求 (OBC) を使用していないことを確認します。
- カスタムリソース (カスタムストレージクラス、cephblockpools など) が管理者によって作成された場合、それらを消費したリソースを削除した後に管理者によって削除される必要があります。

手順

1. OpenShift Container Storage を使用しているボリュームスナップショットを削除します。

- a. すべての namespace からボリュームスナップショットを一覧表示します。

```
$ oc get volumesnapshot --all-namespaces
```

- b. 直前のコマンドの出力から、OpenShift Container Storage を使用しているボリュームスナップショットを特定し、削除します。

```
$ oc delete volumesnapshot <VOLUME-SNAPSHOT-NAME> -n <NAMESPACE>
```

2. OpenShift Container Storage を使用している PVC および OBC を削除します。
デフォルトのアンインストールモード (graceful) では、アンインストーラーは OpenShift Container Storage を使用するすべての PVC および OBC が削除されるまで待機します。

PVC を事前に削除せずに Storage Cluster を削除する場合は、アンインストールモードのアノテーションを forced に設定し、この手順を省略できます。これを実行すると、孤立した PVC および OBC がシステムに作成されます。

- a. OpenShift Container Storage を使用して、OpenShift Container Platform モニターリングスタック PVC を削除します。
[「OpenShift Container Storage からのモニターリングスタックの削除」](#) を参照してください。
- b. OpenShift Container Storage を使用して、OpenShift Container Platform レジストリー PVC を削除します。
[「OpenShift Container Storage からの OpenShift Container Platform レジストリーの削除」](#) を参照してください。

- c. OpenShift Container Storage を使用して、OpenShift Container Platform ロギング PVC を削除します。
「[OpenShift Container Storage からのクラスターロギング Operator の削除](#)」を参照してください。
- d. OpenShift Container Storage を使用してプロビジョニングした PVC および OBC を削除します。
 - 以下に、OpenShift Container Storage を使用してプロビジョニングされる PVC および OBC を特定するサンプルスクリプトを示します。このスクリプトは、OpenShift Container Storage によって内部で使用される PVC を無視します。

```
#!/bin/bash

RBD_PROVISIONER="openshift-storage.rbd.csi.ceph.com"
CEPHFS_PROVISIONER="openshift-storage.cephfs.csi.ceph.com"
NOOBAA_PROVISIONER="openshift-storage.noobaa.io/obc"
RGW_PROVISIONER="openshift-storage.ceph.rook.io/bucket"

NOOBAA_DB_PVC="noobaa-db"
NOOBAA_BACKINGSTORE_PVC="noobaa-default-backing-store-noobaa-pvc"

# Find all the OCS StorageClasses
OCS_STORAGECLASSES=$(oc get storageclasses | grep -e
"$RBD_PROVISIONER" -e "$CEPHFS_PROVISIONER" -e
"$NOOBAA_PROVISIONER" -e "$RGW_PROVISIONER" | awk '{print $1}')

# List PVCs in each of the StorageClasses
for SC in $OCS_STORAGECLASSES
do
    echo
    "=====
    ==
    echo "$SC StorageClass PVCs and OBCs"
    echo
    "=====
    ==
    oc get pvc --all-namespaces --no-headers 2>/dev/null | grep $SC | grep -v -e
"$NOOBAA_DB_PVC" -e "$NOOBAA_BACKINGSTORE_PVC"
    oc get obc --all-namespaces --no-headers 2>/dev/null | grep $SC
    echo
done
```



注記

クラウドプラットフォームの **RGW_PROVISIONER** を省略します。

- OBC を削除します。

```
$ oc delete obc <obc name> -n <project name>
```

- PVC を削除します。

```
$ oc delete pvc <pvc name> -n <project-name>
```



注記

クラスターに作成されているカスタムバックリングストア、バケットクラスなどを削除していることを確認します。

- Storage Cluster オブジェクトを削除し、関連付けられたリソースが削除されるのを待機します。

```
$ oc delete -n openshift-storage storagecluster --all --wait=true
```

- uninstall.ocs.openshift.io/cleanup-policy** が **delete** (default) に設定されている場合にクリーンアップ Pod の有無を確認し、それらのステータスが **Completed** していることを確認します。

```
$ oc get pods -n openshift-storage | grep -i cleanup
NAME                                READY STATUS RESTARTS AGE
cluster-cleanup-job-<xx>            0/1   Completed 0      8m35s
cluster-cleanup-job-<yy>            0/1   Completed 0      8m35s
cluster-cleanup-job-<zz>            0/1   Completed 0      8m35s
```

- /var/lib/rook** ディレクトリーが空であることを確認します。このディレクトリーは空になるのは、**uninstall.ocs.openshift.io/cleanup-policy** アノテーションが **delete** (デフォルト) に設定されている場合に限られます。

```
$ for i in $(oc get node -l cluster.ocs.openshift.io/openshift-storage= -o jsonpath='{.items[*].metadata.name}'); do oc debug node/${i} -- chroot /host ls -l /var/lib/rook; done
```

- 暗号化がインストール時に有効にされている場合は、すべての OpenShift Container Storage ノードの OSD デバイスから **dm-crypt** で管理される **device-mapper** マッピングを削除します。

- デバッグ** Pod を作成し、ストレージノードのホストに対して **chroot** を作成します。

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

- デバイス名を取得し、OpenShift Container Storage デバイスについてメモします。

```
$ dmsetup ls
ocs-deviceset-0-data-0-57snx-block-dmccrypt (253:1)
```

- マップ済みデバイスを削除します。

```
$ cryptsetup luksClose --debug --verbose ocs-deviceset-0-data-0-57snx-block-dmccrypt
```



注記

権限が十分でないため、コマンドがスタックした場合には、以下のコマンドを実行します。

- **CTRL+Z** を押して上記のコマンドを終了します。

- スタックしたプロセスの PID を検索します。

```
$ ps -ef | grep crypt
```

- **kill** コマンドを使用してプロセスを終了します。

```
$ kill -9 <PID>
```

- デバイス名が削除されていることを確認します。

```
$ dmsetup ls
```

7. namespace を削除し、削除が完了するまで待機します。**openshift-storage** がアクティブなプロジェクトである場合は、別のプロジェクトに切り替える必要があります。以下に例を示します。

```
$ oc project default
$ oc delete project openshift-storage --wait=true --timeout=5m
```

以下のコマンドが NotFound エラーを返すと、プロジェクトが削除されます。

```
$ oc get project openshift-storage
```



注記

OpenShift Container Storage のアンインストール時に、**namespace** が完全に削除されず、**Terminating** 状態のままである場合は、[Troubleshooting and deleting remaining resources during Uninstall](#) の記事に記載の手順を実行して namespace の終了をブロックしているオブジェクトを特定します。

8. ストレージノードのラベルを解除します。

```
$ oc label nodes --all cluster.ocs.openshift.io/openshift-storage-
$ oc label nodes --all topology.rook.io/rack-
```

9. ノードにテイントのマークが付けられている場合に OpenShift Container Storage テイントを削除します。

```
$ oc adm taint nodes --all node.ocs.openshift.io/storage-
```

10. OpenShift Container Storage を使用してプロビジョニングした PV がすべて削除されていることを確認します。**Released** 状態のままの PV がある場合は、これを削除します。

```
$ oc get pv
$ oc delete pv <pv name>
```

11. Multicloud Object Gateway storageclass を削除します。

```
$ oc delete storageclass openshift-storage.noobaa.io --wait=true --timeout=5m
```

12. **CustomResourceDefinitions** を削除します。

```
$ oc delete crd backingstores.noobaa.io bucketclasses.noobaa.io
cephblockpools.ceph.rook.io cephclusters.ceph.rook.io cephfilesystems.ceph.rook.io
cephnfses.ceph.rook.io cephobjectstores.ceph.rook.io cephobjectstoreusers.ceph.rook.io
noobaas.noobaa.io ocsinitializations.ocs.openshift.io storageclusters.ocs.openshift.io
cephclients.ceph.rook.io cephobjectrealms.ceph.rook.io cephobjectzonegroups.ceph.rook.io
cephobjectzones.ceph.rook.io cephrobdmirrors.ceph.rook.io --wait=true --timeout=5m
```

13. オプション: vault キーが完全に削除されるようにするには、vault キーに関連付けられているメタデータを手動で削除する必要があります。



注記

このステップは、Vault Key/Value (KV) シークレットエンジン API バージョン 2 が、OpenShift Container Storage のアンインストール中に Vault キーが削除済みとしてマークされ、完全に削除されないため、Key Management System (KMS) によるクラスター全体の暗号化に使用される場合にのみ実行してください。必要に応じて、後でいつでも復元できます。

- a. Vault 内のキーを一覧表示します。

```
$ vault kv list <backend_path>
```

<backend_path>

暗号化キーが保存されている Vault 内のパスです。
以下に例を示します。

```
$ vault kv list kv-v2
```

出力例:

```
Keys
-----
NOOBAA_ROOT_SECRET_PATH/
rook-ceph-osd-encryption-key-ocs-deviceset-thin-0-data-0m27q8
rook-ceph-osd-encryption-key-ocs-deviceset-thin-1-data-0sq227
rook-ceph-osd-encryption-key-ocs-deviceset-thin-2-data-0xzszb
```

- b. Vault キーに関連付けられているメタデータを一覧表示します。

```
$ vault kv get kv-v2/<key>
```

Multicloud Object Gateway (MCG) キーの場合:

```
$ vault kv get kv-v2/NOOBAA_ROOT_SECRET_PATH/<key>
```

<key>

暗号化キーです。
以下に例を示します。

```
$ vault kv get kv-v2/rook-ceph-osd-encryption-key-ocs-deviceset-thin-0-data-0m27q8
```

出力例:

```
===== Metadata =====
Key          Value
---          -
created_time  2021-06-23T10:06:30.650103555Z
deletion_time 2021-06-23T11:46:35.045328495Z
destroyed     false
version       1
```

- c. メタデータを削除します。

```
$ vault kv metadata delete kv-v2/<key>
```

MCG キーの場合:

```
$ vault kv metadata delete kv-v2/NOOBAA_ROOT_SECRET_PATH/<key>
```

<key>

暗号化キーです。
以下に例を示します。

```
$ vault kv metadata delete kv-v2/rook-ceph-osd-encryption-key-ocs-deviceset-thin-0-
data-0m27q8
```

出力例:

```
Success! Data deleted (if it existed) at: kv-v2/metadata/rook-ceph-osd-encryption-key-
ocs-deviceset-thin-0-data-0m27q8
```

- d. これらの手順を繰り返して、すべての Vault キーに関連付けられているメタデータを削除します。
14. OpenShift Container Platform Web コンソールで、OpenShift Container Storage が完全にアンインストールされていることを確認するには、以下を実行します。
- Home → Overview** をクリックし、ダッシュボードにアクセスします。
 - Persistent Storage および Object Service タブが **Cluster** タブの横に表示されなくなることを確認します。

4.2. OPENSIFT CONTAINER STORAGE からのモニターリングスタックの削除

このセクションでは、モニタリングスタックを OpenShift Container Storage からクリーンアップします。

モニタリングスタックの設定の一部として作成される PVC は **openshift-monitoring** namespace に置かれます。

前提条件

- PVC は OpenShift Container Platform モニタリングスタックを使用できるように設定されます。
詳細は、[モニタリングスタックの設定](#) を参照してください。

手順

- openshift-monitoring** namespace で現在実行されている Pod および PVC を一覧表示します。

```
$ oc get pod,pvc -n openshift-monitoring
```

NAME	READY	STATUS	RESTARTS	AGE
pod/alertmanager-main-0	3/3	Running	0	8d
pod/alertmanager-main-1	3/3	Running	0	8d
pod/alertmanager-main-2	3/3	Running	0	8d
pod/cluster-monitoring-operator-84457656d-pkrxm	1/1	Running	0	8d
pod/grafana-79ccf6689f-2ll28	2/2	Running	0	8d
pod/kube-state-metrics-7d86fb966-rvd9w	3/3	Running	0	8d
pod/node-exporter-25894	2/2	Running	0	8d
pod/node-exporter-4dsd7	2/2	Running	0	8d
pod/node-exporter-6p4zc	2/2	Running	0	8d
pod/node-exporter-jbjvg	2/2	Running	0	8d
pod/node-exporter-jj4t5	2/2	Running	0	6d18h
pod/node-exporter-k856s	2/2	Running	0	6d18h
pod/node-exporter-rf8gn	2/2	Running	0	8d
pod/node-exporter-rmb5m	2/2	Running	0	6d18h
pod/node-exporter-zj7kx	2/2	Running	0	8d
pod/openshift-state-metrics-59dbd4f654-4clng	3/3	Running	0	8d
pod/prometheus-adapter-5df5865596-k8dzn	1/1	Running	0	7d23h
pod/prometheus-adapter-5df5865596-n2gj9	1/1	Running	0	7d23h
pod/prometheus-k8s-0	6/6	Running	1	8d
pod/prometheus-k8s-1	6/6	Running	1	8d
pod/prometheus-operator-55cfb858c9-c4zd9	1/1	Running	0	6d21h
pod/telemeter-client-78fc8fc97d-2rgfp	3/3	Running	0	8d

NAME	CAPACITY	ACCESS MODES	STATUS	VOLUME STORAGECLASS	AGE
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-0	40Gi	RWO	Bound	pvc-0d519c4f-15a5-11ea-baa0-026d231574aa	8d
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-1	40Gi	RWO	Bound	pvc-0d5a9825-15a5-11ea-baa0-026d231574aa	8d
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-2			Bound	pvc-	

```
0d6413dc-15a5-11ea-baa0-026d231574aa 40Gi RWO ocs-storagecluster-ceph-
rbd 8d
persistentvolumeclaim/my-prometheus-claim-prometheus-k8s-0 Bound pvc-0b7c19b0-
15a5-11ea-baa0-026d231574aa 40Gi RWO ocs-storagecluster-ceph-rbd 8d
persistentvolumeclaim/my-prometheus-claim-prometheus-k8s-1 Bound pvc-0b8aed3f-
15a5-11ea-baa0-026d231574aa 40Gi RWO ocs-storagecluster-ceph-rbd 8d
```

2. モニタリング **configmap** を編集します。

```
$ oc -n openshift-monitoring edit configmap cluster-monitoring-config
```

3. 以下の例が示すように、OpenShift Container Storage ストレージクラスを参照する **config** セクションを削除し、これを保存します。

編集前

```
.
.
.
apiVersion: v1
data:
  config.yaml: |
    alertmanagerMain:
      volumeClaimTemplate:
        metadata:
          name: my-alertmanager-claim
        spec:
          resources:
            requests:
              storage: 40Gi
          storageClassName: ocs-storagecluster-ceph-rbd
    prometheusK8s:
      volumeClaimTemplate:
        metadata:
          name: my-prometheus-claim
        spec:
          resources:
            requests:
              storage: 40Gi
          storageClassName: ocs-storagecluster-ceph-rbd
kind: ConfigMap
metadata:
  creationTimestamp: "2019-12-02T07:47:29Z"
  name: cluster-monitoring-config
  namespace: openshift-monitoring
  resourceVersion: "22110"
  selfLink: /api/v1/namespaces/openshift-monitoring/configmaps/cluster-monitoring-config
  uid: fd6d988b-14d7-11ea-84ff-066035b9efa8
.
.
.
```

編集後

```

.
.
.
apiVersion: v1
data:
  config.yaml: |
kind: ConfigMap
metadata:
  creationTimestamp: "2019-11-21T13:07:05Z"
  name: cluster-monitoring-config
  namespace: openshift-monitoring
  resourceVersion: "404352"
  selfLink: /api/v1/namespaces/openshift-monitoring/configmaps/cluster-monitoring-config
  uid: d12c796a-0c5f-11ea-9832-063cd735b81c
.
.
.

```

この例では、**alertmanagerMain** および **prometheusK8s** モニターリングコンポーネントは OpenShift Container Storage PVC を使用しています。

4. 関連する PVC を削除します。ストレージクラスを使用するすべての PVC を削除してください。

```
$ oc delete -n openshift-monitoring pvc <pvc-name> --wait=true --timeout=5m
```

4.3. OPENSIFT CONTAINER STORAGE からの OPENSIFT CONTAINER PLATFORM レジストリーの削除

このセクションを使用して、OpenShift Container Storage から OpenShift Container Platform レジストリーをクリーンアップします。代替ストレージを設定する必要がある場合、[イメージレジストリー](#)を参照してください。

OpenShift Container Platform レジストリーの設定の一部として作成される PVC は **openshift-image-registry** namespace に置かれます。

前提条件

- イメージレジストリーは OpenShift Container Storage PVC を使用するように設定されている必要があります。

手順

1. **configs.imageregistry.operator.openshift.io** オブジェクトを編集し、**storage** セクションのコンテンツを削除します。

```
$ oc edit configs.imageregistry.operator.openshift.io
```

編集前


```

.
.
.
storage:
  pvc:
    claim: registry-cephfs-rwx-pvc
.
.
.

```

編集後

```

.
.
.
storage:
.
.
.

```

この例では、PVC は **registry-cephfs-rwx-pvc** と呼ばれ、これは安全に削除できます。

2. PVC を削除します。

```
$ oc delete pvc <pvc-name> -n openshift-image-registry --wait=true --timeout=5m
```

4.4. OPENSIFT CONTAINER STORAGE からのクラスターロギング OPERATOR の削除

このセクションでは、クラスターロギング Operator を OpenShift Container Storage からクリーンアップします。

クラスターロギング Operator の設定の一部として作成される PVC は **openshift-logging** namespace にあります。

前提条件

- クラスターロギングインスタンスは、OpenShift Container Storage PVC を使用するよう設定されている必要があります。

手順

1. namespace の **ClusterLogging** インスタンスを削除します。

```
$ oc delete clusterlogging instance -n openshift-logging --wait=true --timeout=5m
```

openshift-logging namespace の PVC は安全に削除できます。

2. PVC を削除します。

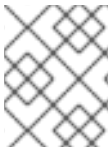
```
$ oc delete pvc <pvc-name> -n openshift-logging --wait=true --timeout=5m
```

第5章 ストレージクラスおよびストレージプール

OpenShift Container Storage 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 Container Storage 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、または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 Storage クラスターが **Ready** 状態にあることを確認します。

手順

1. OpenShift Web コンソールにログインします。
2. **Storage** → **Storage Classes** をクリックします。
3. **Create Storage Class** をクリックします。
4. ストレージクラスの **Name** および **Description** を入力します。
5. 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。
6. 永続ボリュームのプロビジョニングに使用されるプラグインである RBD Provisioner を選択します。
7. 新規プールを作成するか、または既存プールを使用できます。

新規プールを作成します。

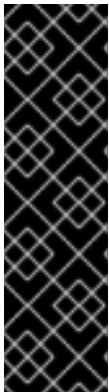
- a. プールの名前を入力します。
- b. Data Protection Policy として **2-way-Replication** または **3-way-Replication** を選択します。
- c. データを圧縮する必要がある場合は、**Enable compression** を選択します。
圧縮を有効にするとアプリケーションのパフォーマンスに影響がある可能性があり、書き込まれるデータがすでに圧縮または暗号化されている場合は効果的ではない可能性があります。圧縮を有効にする前に書き込まれたデータは圧縮されません。
- d. **Create** をクリックしてストレージクラスを作成します。
- e. プールの作成後に **Finish** をクリックします。
- f. **Create** をクリックしてストレージクラスを作成します。

既存プールを使用します。

- a. 一覧からプールを選択します。
- b. **Create** をクリックしてプールが選択された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

外部鍵管理システム (KMS) を使用したストレージクラスの暗号化によってプロビジョニングされる永続ボリュームの暗号化はテクノロジープレビュー機能です。永続ボリュームの暗号化は RBD PV の場合にのみ利用できます。



重要

ストレージクラスの暗号化は、RBD PV でのみ利用可能な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 の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

詳細は、[テクノロジープレビュー機能のサポート範囲](#) を参照してください。

前提条件

- OpenShift Container Storage クラスターが **Ready** 状態にある。
- 外部の鍵管理システム (KMS) で、以下を実行します。
 - トークンのあるポリシーが存在し、Vault のキー値のバックエンドパスが有効にされていることを確認します。[Enabling key value and policy in Vault](#) 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。
- 以下のようにテナントの namespace にシークレットを作成します。
 - OpenShift Container Platform Web コンソールで、**Workloads** → **Secrets**に移動します。

- **Create** → **Key/value secret**をクリックします。
- **Secret Name** を **ceph-csi-kms-token** として入力します。
- **Key** を **token** として入力します。
- **Value** を入力します。これは Vault のトークンです。 **Browse** をクリックしてトークンが含まれるファイルを選択し、アップロードするか、またはテキストボックスにトークンを直接入力します。
- **Create** をクリックします。



注記

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

手順

1. **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 (削除) または Retain (保持) のいずれかを選択します。デフォルトで、Delete (削除) が選択されます。
5. 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。
6. ボリュームデータが保存される **Storage Pool**を選択します。
7. **Enable Encryption** チェックボックスを選択します。
 - a. **Key Management Service Provider**はデフォルトで Vault に設定されます。
 - b. Vault の **Service Name**、Vault サーバーのホストの **Address** ('https://<hostname または ip>')、および **Port number** を入力します。
 - c. **Advanced Settings** を拡張して、証明書の詳細を入力します。
 - i. OpenShift Container Storage 専用かつ特有のキー値のシークレットパスを **Backend Path** に入力します。
 - ii. (オプション) **TLS Server Name** および **Vault Enterprise Namespace**を入力します。
 - iii. それぞれの PEM でエンコードされた証明書ファイルをアップロードして、**CA Certificate**、**Client Certificate**、および **Client Private Key** を指定します。
 - iv. **Save** をクリックします。
 - d. **Connect** をクリックします。
8. 外部鍵管理サービスの接続の詳細を確認します。情報を変更するには、**Change connection details** をクリックし、フィールドを編集します。

9. **Create** をクリックします。
10. 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 **VAULT_BACKEND** パラメーターを追加します。



注記

VAULT_BACKEND 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** → **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、**VAULT_BACKEND** パラメーターを追加します。
VAULT_BACKEND パラメーターとして、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: >-

  {
    "KMS_PROVIDER": "vaulttokens",
    "KMS_SERVICE_NAME": "vault",
```

```
[...]  
"VAULT_BACKEND": "kv-v2"  
}
```

iii. **Save** をクリックします。



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 OPENSIFT CONTAINER PLATFORM サービスのストレージの設定

OpenShift Container Storage を使用して、イメージレジストリー、モニターリング、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 Container Storage デプロイメントで使用す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ー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 [Curator スケジュールの設定](#) および [永続ストレージの設定](#) の [Prometheus メトリクスデータの保持時間の変更](#) サブセクション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. OPENSIFT CONTAINER STORAGE 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 Container Storage をコンテナ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 Container Storage 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。
- 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** → **Storage Classes** 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. OPENSIFT CONTAINER STORAGE を使用するためのモニターリングの設定

OpenShift Container Storage は、Prometheus および Alert Manager で設定されるモニター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 Container Storage をモニター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、モニターリングは機能しません。モニターリング用に十分なストレージ容量があることを常に確認します。

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

前提条件

- OpenShift Web コンソールへの管理者アクセスがある。
- OpenShift Container Storage 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 → Storage Classes** 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。





















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

Project: openshift-monitoring ▼

Persistent Volume Claims

Create Persistent Volume Claim Filter by name...





0 Pending 5 Bound 0 Lost Select All Filters 5 Items

Name ↑	Namespace ↓	Status ↓	Persistent Volume ↓	Requested ↓
 my-alertmanager-claim-alertmanager-main-0	 openshift-monitoring	 Bound	 pvc-d00428a5-0ce6-11ea-8fe8-023bdfa29edc	40Gi
 my-alertmanager-claim-alertmanager-main-1	 openshift-monitoring	 Bound	 pvc-d00be111-0ce6-11ea-8fe8-023bdfa29edc	40Gi
 my-alertmanager-claim-alertmanager-main-2	 openshift-monitoring	 Bound	 pvc-d01ac717-0ce6-11ea-8fe8-023bdfa29edc	40Gi
 my-prometheus-claim-prometheus-k8s-0	 openshift-monitoring	 Bound	 pvc-ce290f1b-0ce6-11ea-8fe8-023bdfa29edc	40Gi
 my-prometheus-claim-prometheus-k8s-1	 openshift-monitoring	 Bound	 pvc-ce361010-0ce6-11ea-8fe8-023bdfa29edc	40Gi

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**)。

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

Volumes

Name ↓	Mount Path ↓	SubPath ↓	Type	Permissions ↓	Utilized By ↓
config-volume	/etc/alertmanager/config		 alertmanager-main	Read/Write	 alertmanager
ocs-alertmanager-claim	/alertmanager	alertmanager-db	 ocs-alertmanager-claim-alertmanager-main-0	Read/Write	 alertmanager

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** をクリックして、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**)。

prometheus-k8s-* Pod に割り当てられた Persistent Volume Claim(永続ボリューム要求、PVC)

Volumes					
Name	Mount Path	SubPath	Type	Permissions	Utilized By
config-out	/etc/prometheus/config_out		Container Volume	Read-only	prometheus
ocs-prometheus-claim	/prometheus	prometheus-db	PVC ocs-prometheus-claim-prometheus-k8s-0	Read/Write	prometheus

6.3. OPENSIFT CONTAINER STORAGE 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については、[Deploying cluster logging](#) 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 Container Storage 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 Container Storage で対応されるように編集し、OpenShift Container Storage 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ーレーションおよび保持期間を短く設定することを推奨します。詳細は、OpenShift Container Platform ドキュメントの [クラスターロギングキューレーター](#) を参照してください。

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

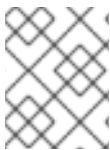
6.3.1. 永続ストレージの設定

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についての詳細は、[クラスターロギングのデプロイおよび設定について](#) の **Elasticsearch レプリケーションポリシー** について参照してください。



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、[Configuring cluster logging](#) を参照してください。

6.3.2. OpenShift Container Storage 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 Container Storage 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 Container Storage で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 Container Storage 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、または **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を更新する必要がある場合があります。

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 Container Storage 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
```

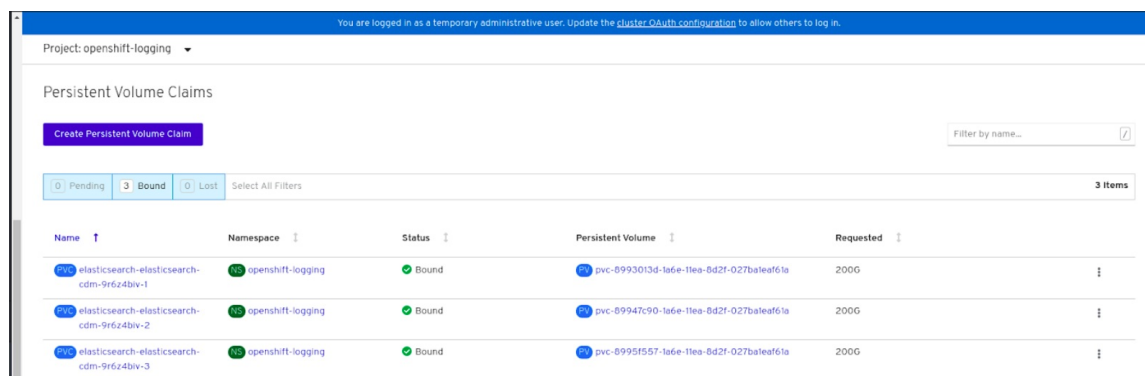
6. **Save** をクリックします。

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6.1 作成済みのバインドされたクラスターロギング



Name	Namespace	Status	Persistent Volume	Requested
elasticsearch-elasticsearch-cdm-9r6z4biv-1	openshift-logging	Bound	pvc-8993013d-1a6e-11ea-8d2f-027bataf61a	200G
elasticsearch-elasticsearch-cdm-9r6z4biv-2	openshift-logging	Bound	pvc-89947c90-1a6e-11ea-8d2f-027bataf61a	200G
elasticsearch-elasticsearch-cdm-9r6z4biv-3	openshift-logging	Bound	pvc-8995f557-1a6e-11ea-8d2f-027bataf61a	200G

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、キュー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、[Curation of Elasticsearch Data](#) 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 Container Storage からの削除についての手順を使用します。

第7章 OPENSIFT CONTAINER STORAGE を使用した OPENSIFT 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 Container Storage 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 Container Platform 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 Container Storage 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 Container Storage 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 OPENSIFT CONTAINER STORAGE に専用のワーカーノードを使用する方法

インフラストラクチャーノードを使用して Red Hat OpenShift Container Storage リソースをスケジューリングすると、Red Hat OpenShift Container Platform サブスクリプションコストを節約できます。**infra** ノードロールのラベルのある Red Hat OpenShift Container Platform (RHOC) ノードには OpenShift Container Storage サブスクリプションが必要ですが、RHOC サブスクリプションは必要ありません。

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 Container Storage 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 RHOC 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 Container Storage を実行するノードには OpenShift Container Storage 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 Container Storage リソースのみをスケジューリングできるようにするには、**NoSchedule** effect のある OpenShift Container Storage テイントを追加する必要があります。

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

RHOC 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 RHOC ノードを **infra** ノードとして識別します。テイントは、OpenShift Container Storage 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リングされないようにします。

OpenShift Container Storage 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に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 Container Storage は 4 つ以上のアベイラビリティ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ノードに必要な適切なティントおよびラベルを持つノードを作成します。これは OpenShift Container Storage 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: ""
```

8.3. インフラストラクチャノードの手動作成

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 Container Storage 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 Container Storage テイントを追加することも、**infra** ノードが OpenShift Container Storage リソースのみをスケジュールし、その他の OpenShift Container Storage 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 Container Storage 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。

第9章 ストレージノードのスケーリング

OpenShift Container Storage 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 Container Storage 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 Container Storage 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で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 OPENSIFT CONTAINER STORAGE 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

以下の手順を使用して、設定された Red Hat OpenShift Container Storage ワーカーノードにストレージ容量を追加し、パフォーマンスを強化します。

前提条件

- 実行中の OpenShift Container Storage Platform
- OpenShift Web コンソールの管理者権限

- デプロイメント時にプロビジョニングされたストレージクラス以外のストレージクラスを使用してスケールアップするには、最初に追加のストレージクラスを定義します。詳細は、[ストレージクラスの作成](#) 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** → **Installed Operators** をクリックします。
3. **OpenShift Container Storage Operator** をクリックします。
4. **Storage Cluster** タブをクリックします。
5. 表示されるリストには1つの項目のみが含まれます。右端の (⋮) をクリックして、オプションメニューを拡張します。
6. オプションメニューから **Add Capacity** を選択します。
7. **Storage Class** 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 Container Storage 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。

8. **Add** をクリックし、クラスターの状態が **Ready** になるまで待機します。

検証手順

- **Overview** → **Persistent Storage** タブに移動してから、**Raw Capacity breakdown** カードをチェックします。
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。

- 3 つの新規 OSD およびそれらの対応する新規 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。
 - 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。

- (オプション) クラスタでクラスタ全体の暗号化が有効な場合には、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 -o=custom-columns=NODE:.spec.nodeName pod/<OSD pod name>
```

以下に例を示します。

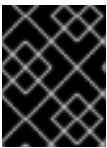
```
oc get -o=custom-columns=NODE:.spec.nodeName pod/rook-ceph-osd-0-544db49d7f-qrqgm
```

- b. 直前の手順で特定されたノードごとに、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>
$ chroot /host
```

- ii. `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
```



重要

ノードまたは OSD を削除して削減するかどうかに関わらず、クラスタの削減は現時点でサポートされていません。

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. 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へのノードの追加

前提条件

- OpenShift Container Platform (RHOC) クラスタにログインする必要があります。

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。
3. ノード数を追加し、**Save** をクリックします。

4. **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。
5. OpenShift Container Storage 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

- 新規ノードが追加されていることを確認するには、[新規ノードの追加の確認](#) について参照してください。

9.3.2. 新規ノードの追加の確認

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-***

9.3.3. ストレージ容量のスケールアップ

新規ノードを OpenShift Container Storage に追加した後に、[容量の追加によるストレージのスケールアップ](#) に説明されているように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、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 Container Storage 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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

- または、**mcg** パッケージを、[Download RedHat OpenShift Container Storage](#) ページにある OpenShift Container Storage RPM からインストールできます。

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」](#)

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

例10.1 例

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

ここで、**<bucket-name>** は 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 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:

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

2. Test S3 client:

```
kubectrl port-forward -n openshift-storage service/s3 10443:443 &
```

1

```
NOOBAA_ACCESS_KEY=$(kubectrl get secret noobaa-admin -n openshift-storage -o json | jq -r
'.data.AWS_ACCESS_KEY_ID|@base64d')
```

2

```
NOOBAA_SECRET_KEY=$(kubectrl get secret noobaa-admin -n openshift-storage -o json | jq -r
'.data.AWS_SECRET_ACCESS_KEY|@base64d')
alias s3='AWS_ACCESS_KEY_ID=$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-](https://a3406079515be11eaa3b70683061451e-1194613580.us-east-2.elb.amazonaws.com:443)

[2.elb.amazonaws.com:443](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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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 : HKLbH1rSuVU0l/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 エンドポイント
- 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。

例10.2 例

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 Container Storage 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. MULTICLOUD OBJECT GATEWAY コンソールへのユーザーアクセスの許可

ユーザーに Multicloud Object Gateway コンソールへのアクセスを許可するには、ユーザーが以下の条件を満たしていることを確認してください。

- ユーザーは **cluster-admins** グループに属する。
- ユーザーは **system:cluster-admins** 仮想グループに属する。

前提条件

- 実行中の OpenShift Container Storage Platform

手順

1. Multicloud Object Gateway へのアクセスを有効にします。

クラスターで以下の手順を実行します。

- a. **cluster-admins** グループを作成します。

```
# oc adm groups new cluster-admins
```

- b. グループを **cluster-admin** ロールにバインドします。

```
# oc adm policy add-cluster-role-to-group cluster-admin cluster-admins
```

2. **cluster-admins** グループからユーザーを追加または削除して、Multicloud Object Gateway コンソールへのアクセスを制御します。

- ユーザーのセットを **cluster-admins** グループに追加するには、以下を実行します。

```
# oc adm groups add-users cluster-admins <user-name> <user-name> <user-name>...
```

ここで、**<user-name>** は追加するユーザーの名前です。



注記

ユーザーのセットを **cluster-admins** グループに追加する場合、新たに追加されたユーザーを cluster-admin ロールにバインドし、OpenShift Container Storage ダッシュボードへのアクセスを許可する必要はありません。

- ユーザーのセットを **cluster-admins** グループから削除するには、以下を実行します。

```
# oc adm groups remove-users cluster-admins <user-name> <user-name> <user-name>...
```

ここで、**<user-name>** は削除するユーザーの名前です。

検証手順

1. OpenShift Web コンソールで、Multicloud Object Gateway コンソールへのアクセスパーミッションを持つユーザーとしてログインします。
2. **Home** → **Overview** → **Object Service** タブ → に移動し、**Multicloud Object Gateway** リンクを選択します。
3. Multicloud Object Gateway コンソールで、アクセスパーミッションを持つ同じユーザーでログインします。
4. **Allow selected permissions** をクリックします。

10.4. 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

10.4.1. 新規バックングストアの作成

以下の手順を使用して、OpenShift Container Storage で新規のバックングストアを作成します。

前提条件

- OpenShift への管理者アクセス。

手順

1. OpenShift Web コンソールの左側のペインで **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。
2. **OpenShift Container Storage Operator** をクリックします。
3. OpenShift Container Storage Operator ページで右側にスクロールし、**Backing Store** タブをクリックします。
4. **Create Backing Store** をクリックします。

図10.1 Create Backing Store ページ

The screenshot shows the 'Create new Backing Store' page in the OpenShift Web Console. The left sidebar contains navigation links for Administrator, Home, Operators, Installed Operators, Workloads, Networking, Storage, Builds, Monitoring, Compute, User Management, and Administration. The main content area displays the 'Create new Backing Store' form. The form includes the following fields and options:

- Backing Store Name ***: A text input field containing 'my-backingstore'.
- Provider ***: A dropdown menu set to 'AWS S3'.
- Region ***: A dropdown menu set to 'us-east-1'.
- Endpoint**: An empty text input field.
- Secret ***: A dropdown menu set to 'Select Secret'.
- Target Bucket ***: An empty text input field.

At the bottom of the form, there are two buttons: 'Create Backing Store' and 'Cancel'.

5. 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。これはオプションです。
 - e. ドロップダウンリストから **Secret** を選択するか、または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。

バックキングストアごとに異なるシークレットが必要です。特定のバック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。

6. **Create Backing Store** をクリックします。

検証手順

1. **Operators → Installed Operators** をクリックします。
2. **OpenShift Container Storage Operator** をクリックします。
3. 新しいバックングストアを検索するか、または **Backing Store** タブをクリックし、すべてのバックングストアを表示します。

10.4.2. 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 で使用できるバック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ク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ク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クングストアの作成」](#) を参照してください。
- IBM COS でサポートされるバック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クングストアの作成」](#) を参照してください。
- Azure でサポートされるバック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クングストアの作成」](#) を参照してください。
- GCP でサポートされるバック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クングストアの作成」](#) 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ク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クングストアの作成」](#) 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 バックングストアの作成」](#) に進み、詳細の手順を確認します。

10.4.2.1. AWS でサポートされるバックングストアの作成

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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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

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。

```
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
```

- a. **<backingstore_name>** を、バックキングストア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、Multicloud Object Gateway に対して、バックキングストア、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 "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こともできます。

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>
```

- a. 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。
 - b. **<backingstore-secret-name>** を一意の名前に置き換えます。
2. 特定のバック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

```

- a. **<bucket-name>** を既存の AWS バケット名に置き換えます。この引数は、Multicloud Object Gateway に対して、バックングストア、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。
- b. **<backingstore-secret-name>** を直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。

10.4.2.2. IBM COS でサポートされるバックングストアの作成

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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg

```

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

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。

```

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

```

- a. **<backingstore_name>** を、バックングストア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。
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。
- 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、Multicloud Object Gateway に対して、バックングストア、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 "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こともできます。

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:
  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>
```

- a. 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。
- b. **<backingstore-secret-name>** を一意の名前に置き換えます。

2. 特定のバック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
```

- a. **<bucket-name>** を既存の IBM COS バケット名に置き換えます。この引数は、Multicloud Object Gateway に対して、バックングストア、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。
- b. **<endpoint>** を、既存の IBM バケット名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。この引数は、Multicloud Object Gateway に対して、バックングストア、およびその後のデータストレージおよび管理に使用するエンドポイントについて指示します。
- c. **<backingstore-secret-name>** を直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。

10.4.2.3. Azure でサポートされるバックングストアの作成

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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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

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。

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

- a. **<backingstore_name>** を、バックングストアの名前に置き換えます。
- b. **<AZURE ACCOUNT KEY>** および **<AZURE ACCOUNT NAME>** は、この目的のために作成した AZURE アカウントキーおよびアカウント名に置き換えます。
- c. **<blob container name>** を既存の Azure blob コンテナ名に置き換えます。この引数は、Multicloud Object Gateway に対して、バックングストア、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 "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こともできます。

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>
```

- a. Base64 を使用して独自の Azure アカウント名およびアカウントキーを指定し、エンコードし、その結果を **<AZURE ACCOUNT NAME ENCODED IN BASE64>** および **<AZURE ACCOUNT KEY ENCODED IN BASE64>** に使用する必要があります。
- b. **<backingstore-secret-name>** を一意の名前に置き換えます。

2. 特定のバック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
```

- a. **<blob-container-name>** を既存の Azure blob コンテナ名に置き換えます。この引数は、Multicloud Object Gateway に対して、バックングストア、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。
- b. **<backingstore-secret-name>** を直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。

10.4.2.4. GCP でサポートされるバックングストアの作成

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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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

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。

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

- a. **<backingstore_name>** を、バックングストアの名前に置き換えます。
- b. **<PATH TO GCP PRIVATE KEY JSON FILE>** を、この目的で作成された GCP プライベートキーへのパスに置き換えます。
- c. **<GCP bucket name>** を、既存の GCP オブジェクトストレージバケット名に置き換えます。この引数は、Multicloud Object Gateway に対して、バックングストア、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こともできます。

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>
```

- a. Base64 を使用して独自の GCP サービスアカウントプライベートキー ID を指定し、エンコードし、その結果を **<GCP PRIVATE KEY ENCODED IN BASE64>** の場所で使用する必要があります。
- b. **<backingstore-secret-name>** を一意の名前に置き換えます。

2. 特定のバック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
```

- a. **<target bucket>** を、既存の Google ストレージバケットに置き換えます。この引数は、Multicloud Object Gateway に対して、バックキングストア、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。
- b. **<backingstore-secret-name>** を直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。

10.4.2.5. ローカル永続ボリュームでサポートされるバック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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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

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。

```
noobaa backingstore create pv-pool <backingstore_name> --num-volumes=<NUMBER OF VOLUMES> --pv-size-gb=<VOLUME SIZE> --storage-class=<LOCAL STORAGE CLASS>
```

- a. **<backingstore_name>** を、バックキングストアの名前に置き換えます。
- b. **<NUMBER OF VOLUMES>** を、作成するボリューム数に置き換えます。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。
- c. **<VOLUME SIZE>** を、各ボリュームに必要なサイズ (GB 単位) に置き換えます。
- d. **<LOCAL STORAGE CLASS>** をローカルストレージクラスに置き換えます。これは、ocs-storagecluster-ceph-rbd を使用する際に推奨されます。出力は次のようになります。

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

YAML を使用してストレージリソースを追加することもできます。

1. 特定のバック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>
        storageClass: <LOCAL STORAGE CLASS>
  type: pv-pool
```

- a. **<backingstore_name>** を、バックキングストアの名前に置き換えます。

- b. **<NUMBER OF VOLUMES>** を、作成するボリューム数に置き換えます。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。
- c. **<VOLUME SIZE>** を、各ボリュームに必要なサイズ (GB 単位) に置き換えます。文字 G はそのままにする必要があることに注意してください。
- d. **<LOCAL STORAGE CLASS>** をローカルストレージクラスに置き換えます。これは、ocs-storagecluster-ceph-rbd を使用する際に推奨されます。

10.4.3. s3 と互換性のある Multicloud Object Gateway バックイングストアの作成

Multicloud Object Gateway 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クイングストアとして使用できます (例: Red Hat Ceph Storage の RADOS Gateway (RGW))。以下の手順では、Red Hat Ceph Storage の RADOS Gateway 用の S3 と互換性のある Multicloud Object Gateway バックイ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 Container Storage Operator は Multicloud Object Gateway の S3 と互換性のあるバックイ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. Multicloud Object Gateway (MCG) コマンドラインインターフェイスから、以下の NooBaa コマンドを実行します。

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

- 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、Multicloud Object Gateway に対して、バックイングストア、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 を使用してバックイ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クングストアについて適用します。

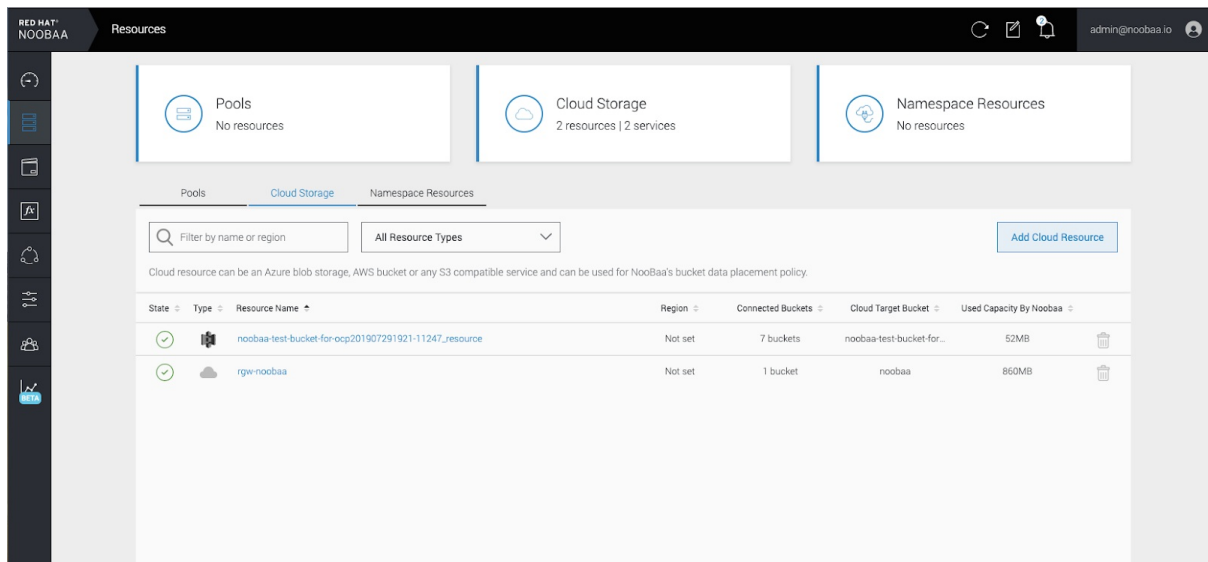
```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
    - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore-name>
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <RGW endpoint>
    secret:
      name: <backingstore-secret-name>
      namespace: 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、Multicloud Object Gateway に対して、バックングストア、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。

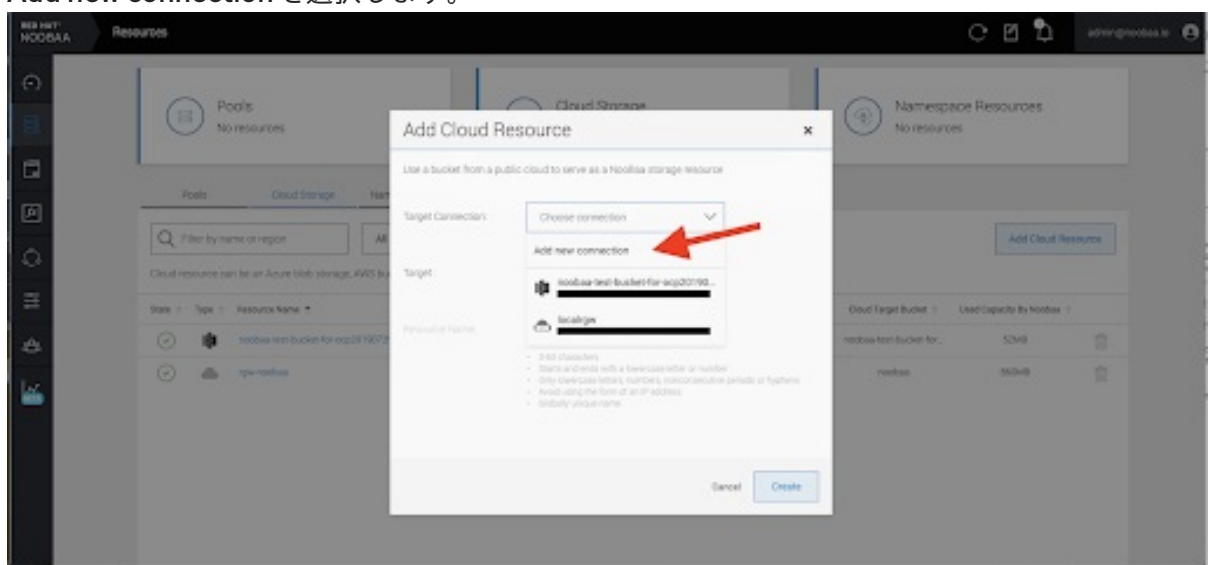
10.4.4. ユーザーインターフェイスを使用したハイブリッドおよびマルチクラウドのストレージリソースの追加

手順

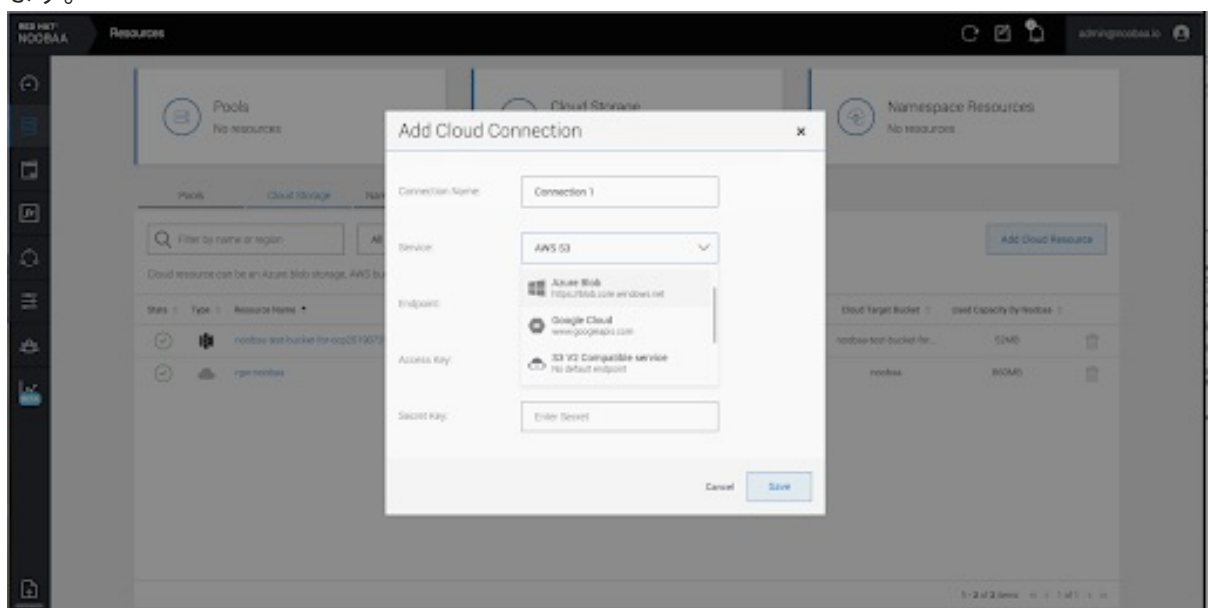
1. OpenShift Storage コンソールで、**Overview** → **Object Service** → **Multicloud Object Gateway** リンクをクリックします。
2. 以下に強調表示されているように左側にある **Resources** タブを選択します。設定する一覧から、**Add Cloud Resource** を選択します。



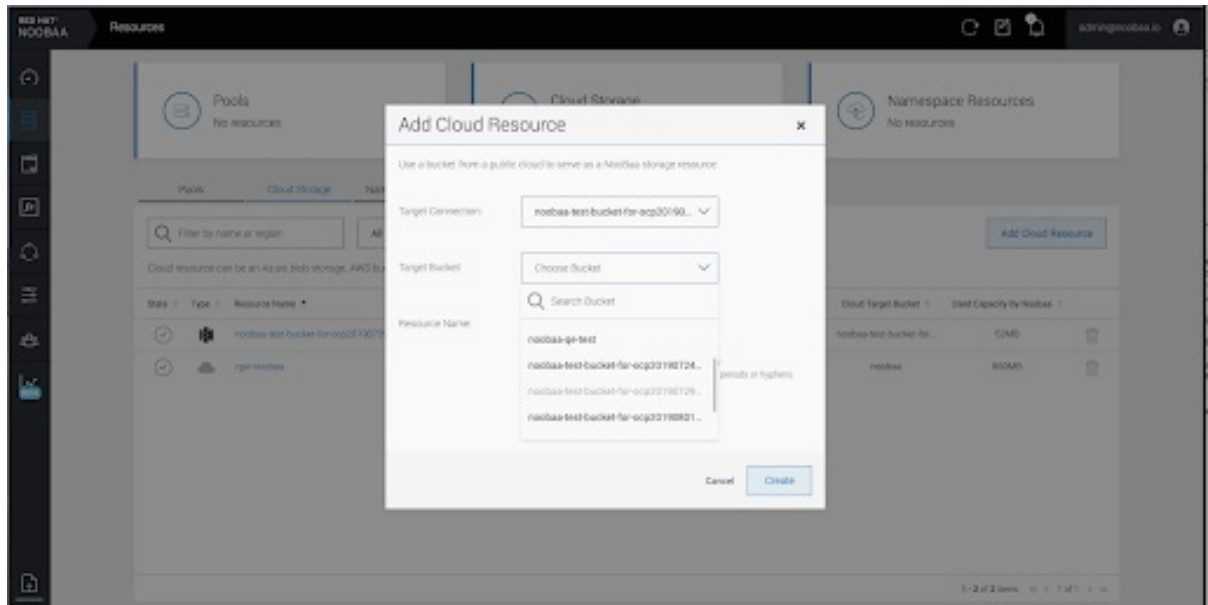
3. Add new connection を選択します。



4. 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。



5. 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。



6. これらの手順を繰り返して、必要な数のバックイ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。

10.4.5. 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 Container Storage でバケットクラスを作成します。

手順

1. OpenShift Web コンソールの左側のペインで **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。
2. **OpenShift Container Storage Operator** をクリックします。
3. OpenShift Container Storage Operator ページで右側にスクロールし、**Bucket Class** タブをクリックします。
4. **Create Bucket Class** をクリックします。
5. Create new Bucket Class ページで、以下を実行します。
 - a. **Bucket Class Name** を入力し、**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、[新しいバック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、2 つ以上のバック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. **Operators** → **Installed Operators** をクリックします。
2. **OpenShift Container Storage Operator** をクリックします。
3. 新しい Bucket Class を検索するか、または **Bucket Class** タブをクリックし、すべての Bucket Class を表示します。

10.4.6. 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 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. **Operators** → **Installed Operators** をクリックします。
3. **OpenShift Container Storage Operator** をクリックします。
4. OpenShift Container Storage Operator ページで右側にスクロールし、**Bucket Class** タブをクリックします。
5. 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。
6. **Edit Bucket Class** をクリックします。
7. **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。

10.4.7. バケットクラスのバックキングストアの編集

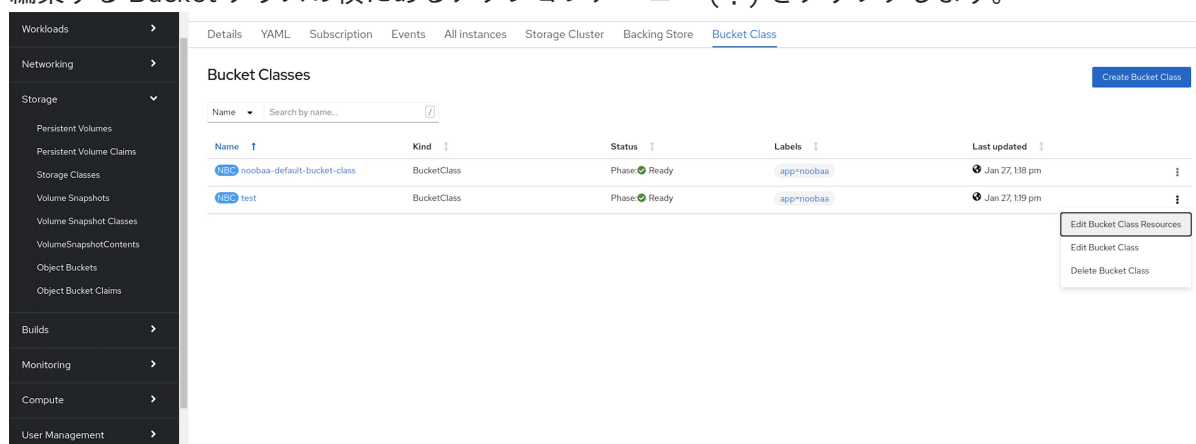
以下の手順を使用して、既存の Multicloud Object Gateway バケットクラスを編集し、バケットクラスで使用される基礎となるバックキングストアを変更します。

前提条件

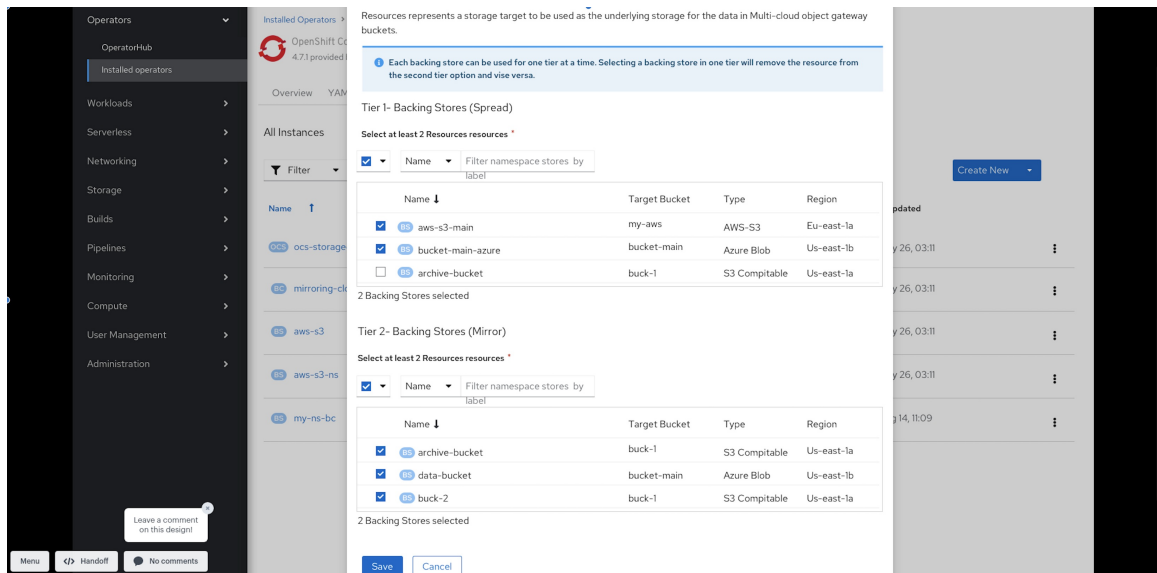
- OpenShift Web コンソールへの管理者アクセス。
- バケットクラス。
- バッキングストア。

手順

1. **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。
2. **OpenShift Container Storage Operator** をクリックします。
3. **Bucket Class** タブをクリックします。
4. 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。



5. **Edit Bucket Class Resources** をクリックします。
6. **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、または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。



7. **Save** をクリックします。

10.5. 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。

1. [プロバイダーを Multicloud Object Gateway に接続します。](#)
2. [プロバイダーのそれぞれに namespace リソースを作成](#) し、それらが namespace バケットに作成されるようにします。
3. [namespace リソースを namespace バケットに追加](#) し、バケットを適切な namespace リソースからの読み込み/への書き込みを行えるように設定します。

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。詳細は、[namespace バケットのオブジェクトの Amazon S3 API エンドポイント](#) について参照してください。



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。

10.5.1. プロバイダー接続の Multicloud Object Gateway への追加

Multicloud Object Gateway がプロバイダーにアクセスできるように各プロバイダーの接続を追加する必要があります。

前提条件

- OpenShift コンソールへの管理者アクセス。

手順

1. OpenShift コンソールで、**Home** → **Overview** をクリックし、**Object Service** タブをクリックします。
2. **Multicloud Object Gateway** をクリックし、プロンプトが表示されたらログインします。
3. **Accounts** をクリックし、接続を追加するアカウントを選択します。
4. **My Connections** をクリックします。
5. **Add Connection** をクリックします。
 - a. **Connection Name** を入力します。
 - b. クラウドプロバイダーは、デフォルトで **Service** ドロップダウンに表示されます。別のプロバイダーを使用するように選択を変更します。
 - c. クラウドプロバイダーのデフォルトエンドポイントはデフォルトで **Endpoint** フィールドに表示されます。必要に応じて代替エンドポイントを入力します。
 - d. このクラウドプロバイダーの **Access Key** を入力します。
 - e. このクラウドプロバイダーの **Secret Key** を入力します。
 - f. **Save** をクリックします。

10.5.2. Multicloud Object Gateway を使用した namespace リソースの追加

既存のストレージを namespace リソースとして Multicloud Storage Gateway に追加し、それらを Amazon Web Services S3 バケット、Microsoft Azure blob、IBM Cloud Object Storage バケットなどの既存のストレージターゲットの統合ビュー用に namespace バケットに含めることができます。

前提条件

- OpenShift コンソールへの管理者アクセス。
- ターゲット接続 (プロバイダー) がすでに Multicloud Object Gateway に追加されている。詳細は、[「プロバイダー接続の Multicloud Object Gateway への追加」](#) を参照してください。

手順

1. OpenShift コンソールで、**Home** → **Overview** をクリックし、**Object Service** タブをクリックします。
2. **Multicloud Storage Gateway** をクリックし、プロンプトが表示されたらログインします。
3. **Resources** をクリックし、**Namespace Resources** タブをクリックします。
4. **Create Namespace Resource** をクリックします。
 - a. **Target Connection** で、この namespace のストレージプロバイダーに使用される接続を選択します。
新規の接続を追加する必要がある場合は、**Add New Connection** をクリックし、プロバイダーの詳細を入力します。詳細は、[「プロバイダー接続の Multicloud Object Gateway への追加」](#) を参照してください。
 - b. **Target Bucket** で、ターゲットとして使用するバケットの名前を選択します。

- c. namespace リソースの **Resource Name** を入力します。
- d. **Create** をクリックします。

検証

- 新規リソースが **State** 列に緑色のチェックマークと共に、また **Connected Namespace Buckets** 列の 0 バケットと共に一覧表示されていることを確認します。

10.5.3. Multicloud Object Gateway を使用した namespace バケットへのリソースの追加

namespace リソースを、各種プロバイダー間でのストレージの統合ビュー用に namespace バケットに追加します。また、1つのプロバイダーのみが新しいデータを受け入れ、すべてのプロバイダーで既存データの読み取りが許可されるように読み取りおよび書き込み動作を設定できます。

前提条件

- バケットで処理する必要のあるすべての namespace リソースが Multicloud Object Gateway: Multicloud Object Gateway に追加されていることを確認します:[Multicloud Object Gateway を使用した namespace リソースの追加](#)

手順

1. OpenShift コンソールで、**Home** → **Overview** をクリックし、**Object Service** タブをクリックします。
2. **Multicloud Object Gateway** をクリックし、プロンプトが表示されたらログインします。
3. **Buckets** をクリックし、**Namespace Buckets** タブをクリックします。
4. **Create Namespace Bucket** をクリックします。
 - a. **Choose Name** タブで、namespace バケットの **Name** 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各 namespace リソースのチェックボックスを選択します。
 - ii. **Write Policy** で、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. 実稼働環境の **Set Caching Policy** タブを変更しないでください。このタブは開発プレビューとして提供され、サポート制限の対象となります。
 - d. **Create** をクリックします。

検証

- 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。

10.5.4. 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 Container Storage 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](#)

10.5.5. 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](#)

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

前提条件

- 実行中の OpenShift Container Storage Platform
- Multicloud Object Gateway へのアクセスについては、第 2 章の [アプリケーションでの Multicloud Object Gateway へのアクセス](#) について参照してください。

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>
```

- a. 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。
 - ii. **<namespacestore-secret-name>** を一意の名前に置き換えます。
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、Multicloud Object Gateway 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。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: mybucketnamespace
  namespace: k8snamespace
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: k8snamespace
    targetBucket: awsdatalake
  type: aws-s3
```

- a. **<namespacestore-secret-name>** を、手順 1 で作成したシークレットに置き換えます。
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> を一意のバケットクラス名に置き換えます。

<resource> を 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 namespace-store に置き換えます。

- 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> を一意のバケットクラス名に置き換えます。

write-resource を、namespace バケットの書き込みターゲットを定義する単一の namespace-store に置き換えます。

<read-resources> を、namespace バケットの読み取りターゲットを定義する namespace-stores の一覧に置き換えます。

4. 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 (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: my-bucket-claim
  namespace: my-app
spec:
  generateBucketName: my-bucket
  storageClassName: noobaa.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>

```

- a. **<my-bucket-class>** を直前の手順で作成したバケットクラスに置き換えます。

OBC が Operator によってプロビジョニングされると、バケットが Multicloud Object Gateway で作成され、Operator は OBC の同じ namespace 上に OBC に同じ名前でシークレットおよび ConfigMap を作成します。

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

前提条件

- 実行中の OpenShift Container Storage Platform
- Multicloud Object Gateway へのアクセスについては、第 2 章の [アプリケーションでの Multicloud Object Gateway へのアクセス](#) について参照してください。

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>

```

- a. 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。
 - b. **<namespacestore-secret-name>** を一意の名前に置き換えます。
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、Multicloud Object Gateway 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。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: k8snamespace
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
    secret:
      name: <namespacestore-secret-name>
      namespace: openshift-storage
    signatureVersion: v2
    targetBucket: BUCKET
    type: ibm-cos

```

- a. **<IBM COS ENDPOINT>** を適切な IBM COS エンドポイントに置き換えます。
 - b. **<namespacestore-secret-name>** を手順 1 で作成したシークレットに置き換えます。
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> を一意のバケットクラス名に置き換えます。

<resource> を 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 namespace-store に置き換えます。

- 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> を一意のバケットクラス名に置き換えます。

write-resource を、namespace バケットの書き込みターゲットを定義する単一の namespace-store に置き換えます。

<read-resources> を、namespace バケットの読み取りターゲットを定義する namespace-stores の一覧に置き換えます。

- 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 (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: my-bucket-claim
  namespace: my-app
spec:
  generateBucketName: my-bucket
  storageClassName: noobaa.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

- <my-bucket-class>** を直前の手順で作成したバケットクラスに置き換えます。

OBC が Operator によってプロビジョニングされると、バケットが Multicloud Object Gateway で作成され、Operator は OBC の同じ namespace 上に OBC に同じ名前でシークレットおよび ConfigMap を作成します。

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

前提条件

- 実行中の OpenShift Container Storage Platform
- Multicloud Object Gateway へのアクセスについては、第 2 章の [アプリケーションでの Multicloud Object Gateway へのアクセス](#) について参照してください。
- Multicloud Object Gateway 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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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 Container Storage RPM からインストールできます。

手順

- NamespaceStore リソースを作成します。NamespaceStore は、Multicloud Object Gateway 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> -n openshift-storage
```

- 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、Multicloud Object Gateway に対して、バックイングストア、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。
- 以下のコマンドを実行して、タイプ **single** の namespace ポリシーで namespace バケットクラスを作成します。

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

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

<resource> を 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 namespace-store に置き換えます。

- 以下のコマンドを実行して、タイプ **multi** の namespace ポリシーで namespace バケットクラスを作成します。

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

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

write-resource を、namespace バケットの書き込みターゲットを定義する単一の namespace-store に置き換えます。

<read-resources> を、namespace バケットの読み取りターゲットを定義する、コンマで区切られた namespace-stores の一覧に置き換えます。

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 (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>
```

- a. **<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。

OBC が Operator によってプロビジョニングされると、バケットが Multicloud Object Gateway で作成され、Operator は OBC の同じ namespace 上に OBC に同じ名前でもシークレットおよび ConfigMap を作成します。

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

並列タスク

前提条件

- 実行中の OpenShift Container Storage Platform
- Multicloud Object Gateway へのアクセスについては、第 2 章の [アプリケーションでの Multicloud Object Gateway へのアクセス](#) について参照してください。
- Multicloud Object Gateway 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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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 Container Storage RPM からインストールできます。

手順

1. NamespaceStore リソースを作成します。NamespaceStore は、Multicloud Object Gateway 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> -n openshift-storage
```

- 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、Multicloud Object Gateway に対して、バックイングストア、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。
 - 以下のコマンドを実行して、タイプ **single** の namespace ポリシーで namespace バケットクラスを作成します。

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

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

<resource> を 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 namespace-store に置き換えます。

- 以下のコマンドを実行して、タイプ **multi** の namespace ポリシーで namespace バケットクラスを作成します。

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

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

write-resource を、namespace バケットの書き込みターゲットを定義する単一の namespace-store に置き換えます。

<read-resources> を、namespace バケットの読み取りターゲットを定義する、コンマで区切られた namespace-stores の一覧に置き換えます。

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 (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>
```

- a. **<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。

OBC が Operator によってプロビジョニングされると、バケットが Multicloud Object Gateway で作成され、Operator は OBC の同じ namespace 上に OBC に同じ名前でもシークレットおよび ConfigMap を作成します。

10.6. 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) 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。

前提条件

- まず、MCG で使用できるバックングストレージを追加する必要があります。[「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。

次に、データ管理ポリシー (ミラーリング) を反映するバケットクラスを作成します。

手順

ミラーリングデータは、以下の 3 つの方法で設定できます。

- 「[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](#)」
- 「[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](#)」
- 「[ユーザーインターフェイスを使用したデータミラーリングを行うためのバケットの設定](#)」

10.6.1. 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

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、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
```

10.6.2. 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。

```
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
```

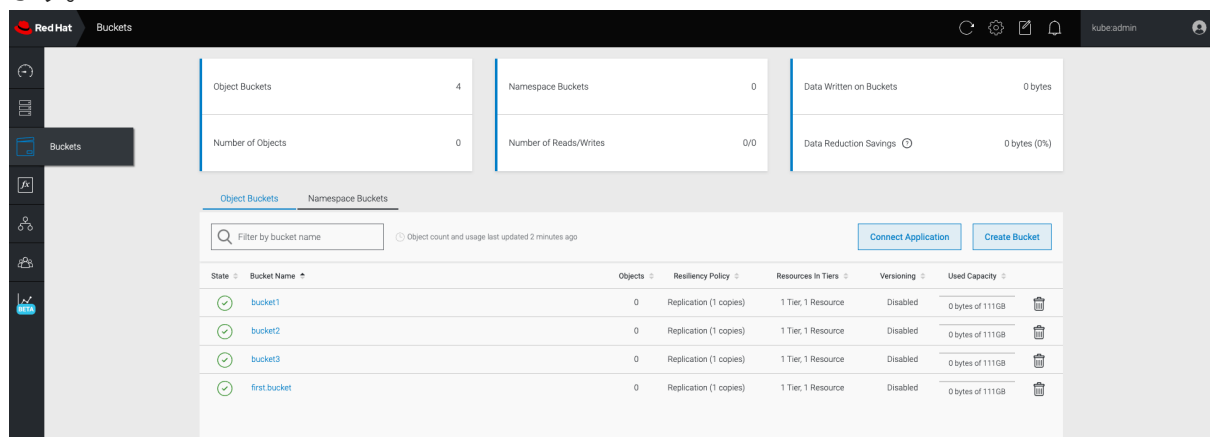
2. 以下の行を標準の Object Bucket Claim (オブジェクトバケット要求、OBC) に追加します。

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

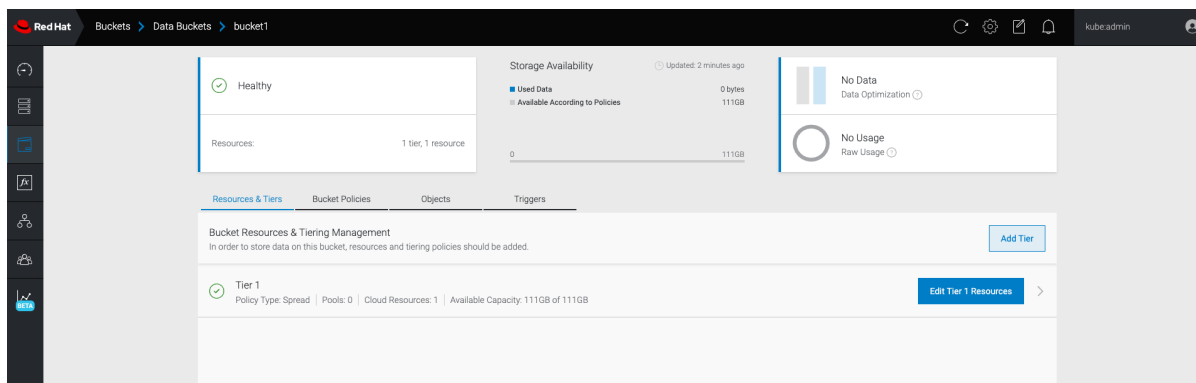
OBC についての詳細は、[「Object Bucket Claim\(オブジェクトバケット要求\)」](#) を参照してください。

10.6.3. ユーザーインターフェイスを使用したデータミラーリングを行うためのバケットの設定

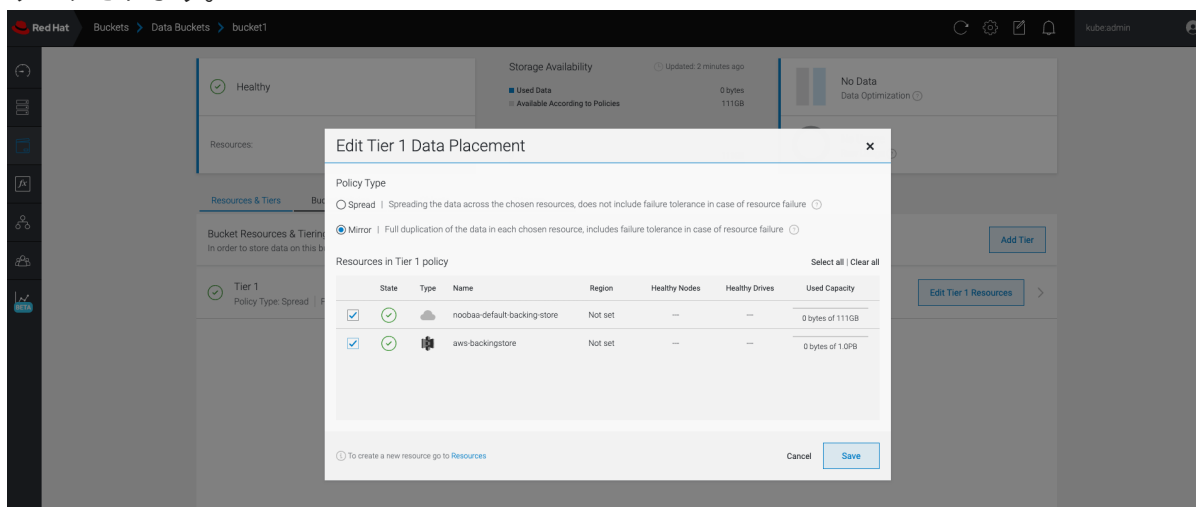
1. OpenShift Storage コンソールで、**Overview → Object Service →** に移動し、**Multicloud Object Gateway** リンクをクリックします。
2. NooBaa ページの左側にある **buckets** アイコンをクリックします。バケットの一覧が表示されます。



3. 更新するバケットをクリックします。
4. **Edit Tier 1 Resources** をクリックします。



5. **Mirror** を選択し、このバケットに使用する関連リソースを確認します。次の例では、RGW にある **noobaa-default-backing-store** と AWS にある **AWS-backingstore** の間のデータがミラーリングされます。



6. **Save** をクリックします。



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。

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

OpenShift Container Storage 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。

10.7.1. バケットポリシーについて

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。

10.7.2. バケットポリシーの使用

前提条件

- 実行中の OpenShift Container Storage Platform
- Multicloud Object Gateway 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

Multicloud Object Gateway 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。以下の例を参照してください。

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

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。

S3 ユーザーの作成方法については、[「Multicloud Object Gateway での AWS S3 ユーザーの作成」](#) を参照してください。

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
```

ENDPOINT を S3 エンドポイントに置き換えます。

MyBucket を、ポリシーを設定するバケットに置き換えます。

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

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。

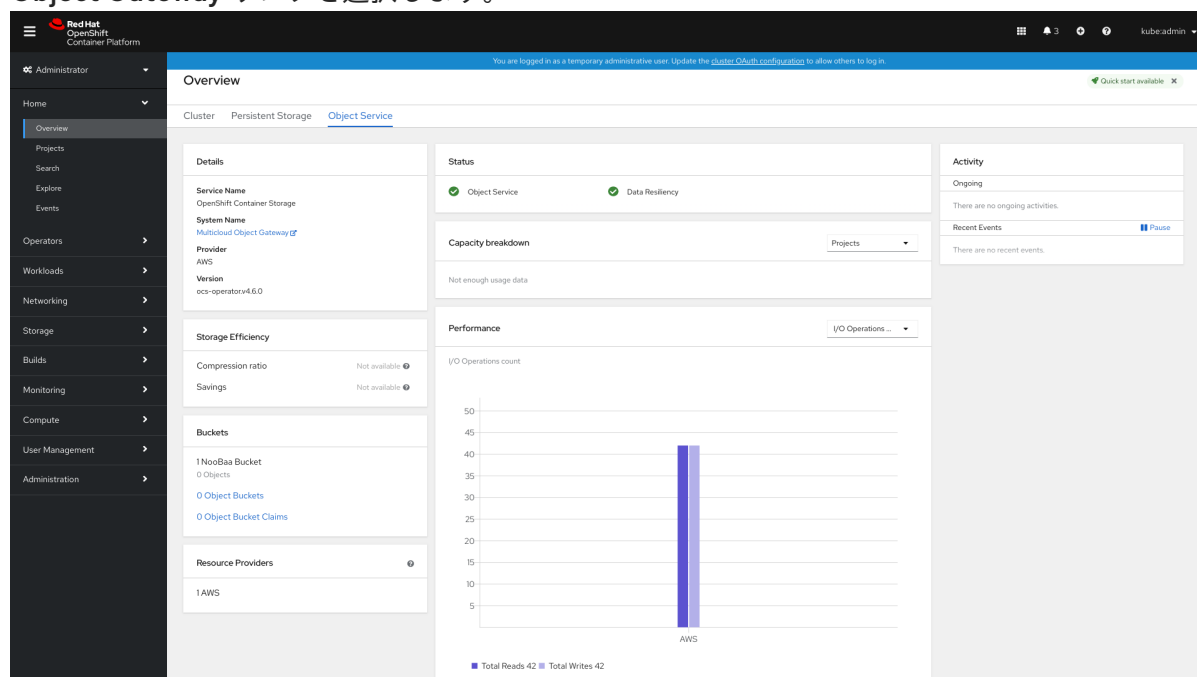
10.7.3. Multicloud Object Gateway での AWS S3 ユーザーの作成

前提条件

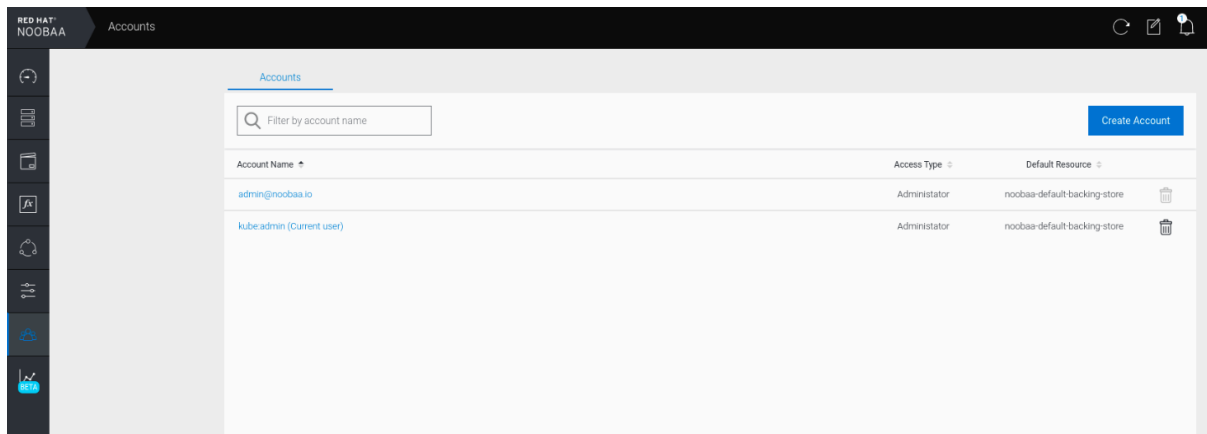
- 実行中の OpenShift Container Storage Platform
- Multicloud Object Gateway へのアクセス。「[アプリケーションの使用による Multicloud Object Gateway へのアクセス](#)」を参照してください。

手順

1. OpenShift Storage コンソールで、**Overview** → **Object Service** → に移動し、**Multicloud Object Gateway** リンクを選択します。



2. **Accounts** タブで、**Create Account** をクリックします。



3. **S3 Access Only** を選択し、**Account Name** を指定します (例: `john.doe@example.com`)。Next をクリックします。

Create Account

1 Account Details
2 S3 Access

Access Type:

☐ Administrator

Enabling administrative access will generate a password that allows login to NooBaa management console as a system admin

☒ S3 Access Only

Granting S3 access will allow this account to connect S3 client applications by generating security credentials (key set).

Account Name:

john.doe@example.com

3 - 32 characters

Cancel
Next

4. **S3 default placement** を選択します (例: `noobaa-default-backing-store`)。 **Buckets Permissions** を選択します。特定のバケットまたはすべてのバケットを選択できます。 **Create** をクリックします。

Create Account

✓ Account Details

2 S3 Access

S3 default placement: ⓘ

noobaa-default-backing-store ▼

Buckets Permissions:

All buckets selected ▼

☒ Include any future buckets

Allow new bucket creation: ⓘ

☒ Enabled

Previous

Create

10.8. 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。

10.8.1. 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 (オブジェクトバケット要求) 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

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。

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 Object Bucket Claim(オブジェクトバケット要求) 自体になります。

- a. **<obc-name>** を、一意の Object Bucket Claim(オブジェクトバケット要求) の名前に置き換えます。
 - b. **<obc-bucket-name>** を、Object Bucket Claim(オブジェクトバケット要求) 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、Object Bucket Claim(オブジェクトバケット要求) の使用を自動化できます。以下の例はバケット要求の結果のマッピングです。これは、データを含む設定マップおよび認証情報のあるシークレットです。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。

```
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

```

- a. <obc-name> のすべてのインスタンスを、Object Bucket Claim(オブジェクトバケット要求)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>
```

- a. **<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>
```

- a. **obc-name** を、Object Bucket Claim(オブジェクトバケット要求)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、または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。

10.8.2. 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(オブジェクトバケット要求) 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

- 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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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 Object Bucket Claim(オブジェクトバケット要求) の名前に置き換えます (例: **myappobc**)。

さらに、**--app-namespace** オプションを使用して、Object Bucket Claim(オブジェクトバケット要求) 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**)。

出力例:

```
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 Object Bucket Claim(オブジェクトバケット要求) を表示します。

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

出力例:

```
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 Object Bucket Claim(オブジェクトバケット要求) の YAML ファイルを表示します。

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

出力例:

```
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 Object Bucket Claim(オブジェクトバケット要求)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 Object Bucket Claim(オブジェクトバケット要求)の名前と同じです。シークレットを表示するには、以下を実行します。

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

出力例:

```

Example output:
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: c0M0R2xVanF3ODR3bHBkVW94cmY=
  AWS_SECRET_ACCESS_KEY:
Wi9kcFluSWxHRzIWaFlzNk1hc0xma2JXcjM1MVhqa051SlBleXpmOQ==
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. 設定マップを表示するには、以下を実行します。

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

出力例:

```
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。

10.8.3. 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。

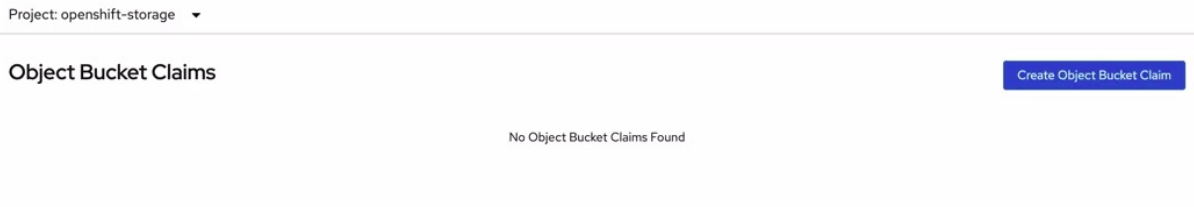
前提条件

- 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。

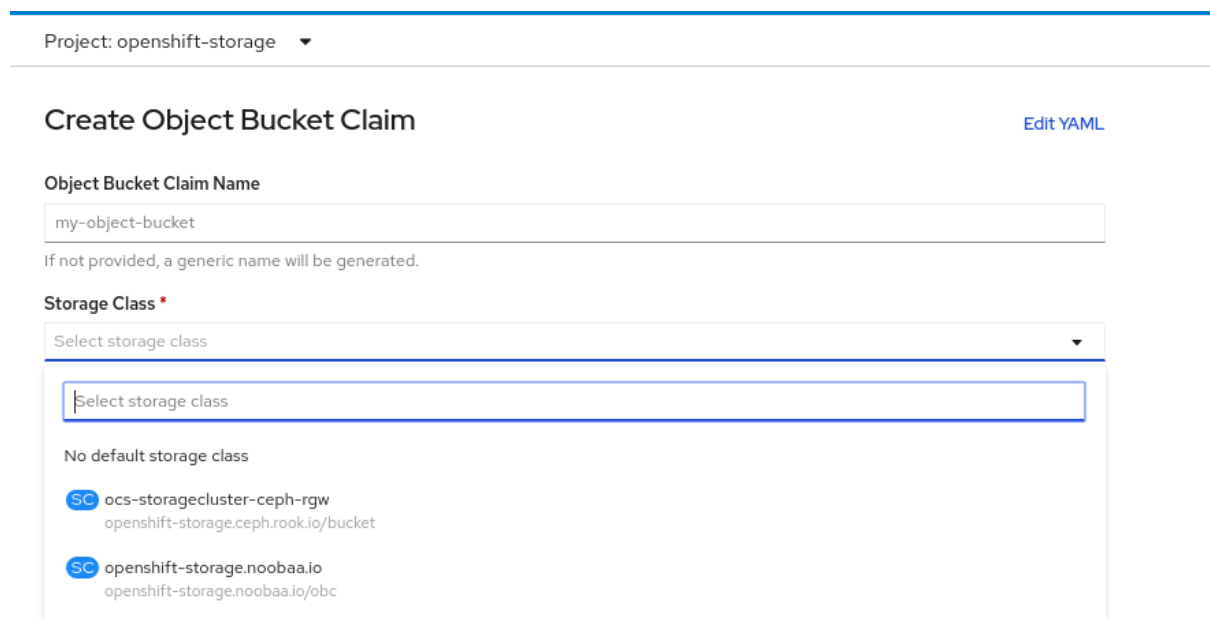
手順

- OpenShift Web コンソールにログインします。
- 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。
- Create Object Bucket Claim** をクリックします。



- 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 を使用します。

外部モード

Project: openshift-storage ▼

Create Object Bucket Claim

[Edit YAML](#)

Object Bucket Claim Name

If not provided, a generic name will be generated.

Storage Class *

No default storage class

 **ocs-external-storagecluster-ceph-rgw**
openshift-storage.ceph.rook.io/bucket

 **openshift-storage.noobaa.io**
openshift-storage.noobaa.io/obc

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。

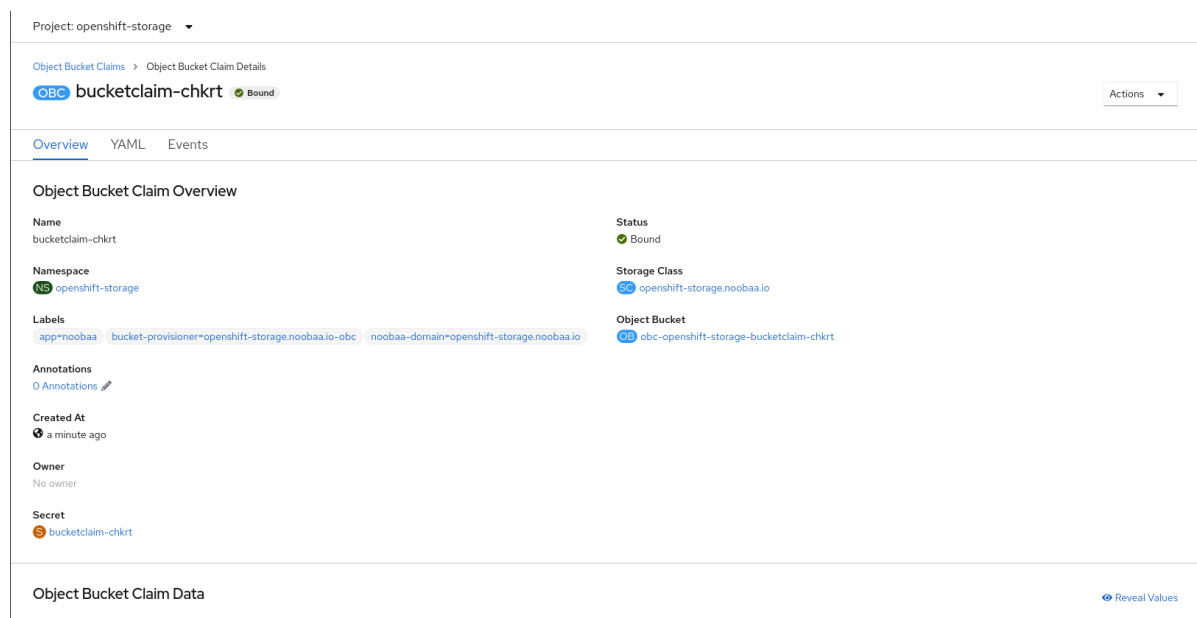
- **ocs-external-storagecluster-ceph-rgw** は Ceph Object Gateway (RGW) を使用します。
- **openshift-storage.noobaa.io** は Multicloud Object Gateway を使用します。



注記

RGW OBC ストレージクラスは、OpenShift Container Storage 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 Container Storage 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。

5. **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。



関連情報

- 「Object Bucket Claim(オブジェクトバケット要求)」

10.8.4. 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

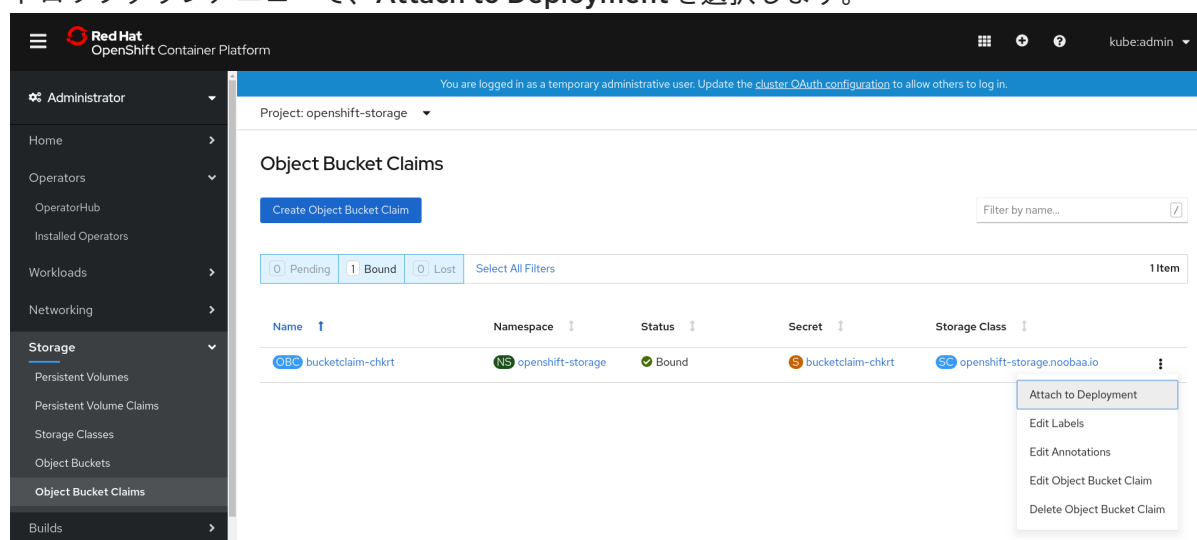
Object Bucket Claim(オブジェクトバケット要求、OBC) は作成後に、特定のデプロイメントに割り当てることができます。

前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

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



- 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。

Attach OBC to a Deployment

Deployment Name *

Cancel Attach

関連情報

- 「Object Bucket Claim(オブジェクトバケット要求)」

10.8.5. 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。

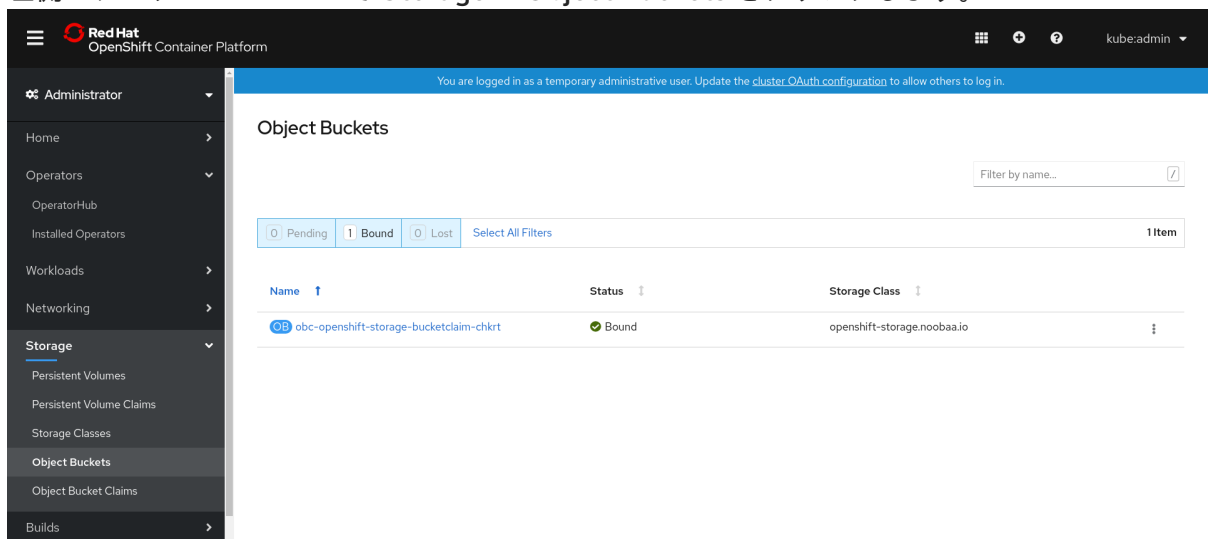
前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

オブジェクトバケットの詳細を表示するには、以下を実行します。

- OpenShift Web コンソールにログインします。
- 左側のナビゲーションバーで **Storage → Object Buckets** をクリックします。



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します。

- 詳細を表示するオブジェクトバケットを選択します。オブジェクトバケットの詳細ページに移動します。

Object Buckets > Object Bucket Details

OB obc-openshift-storage-bucketclaim-chkrt ✔ Bound Actions

[Overview](#) [YAML](#) [Events](#)

Object Bucket Overview

Name obc-openshift-storage-bucketclaim-chkrt	Status ✔ Bound
Labels app=noobaa bucket-provisioner=openshift-storage.noobaa.io-obc noobaa-domain=openshift-storage.noobaa.io	Storage Class SC openshift-storage.noobaa.io
Annotations 0 Annotations	Object Bucket Claim OBC bucketclaim-chkrt
Created At Apr 1, 2:03 pm	
Owner No owner	

関連情報

- 「Object Bucket Claim(オブジェクトバケット要求)」

10.8.6. 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

- 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。
- 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。

Red Hat OpenShift Container Platform

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#) to allow others to log in.

Project: openshift-storage

Object Bucket Claims

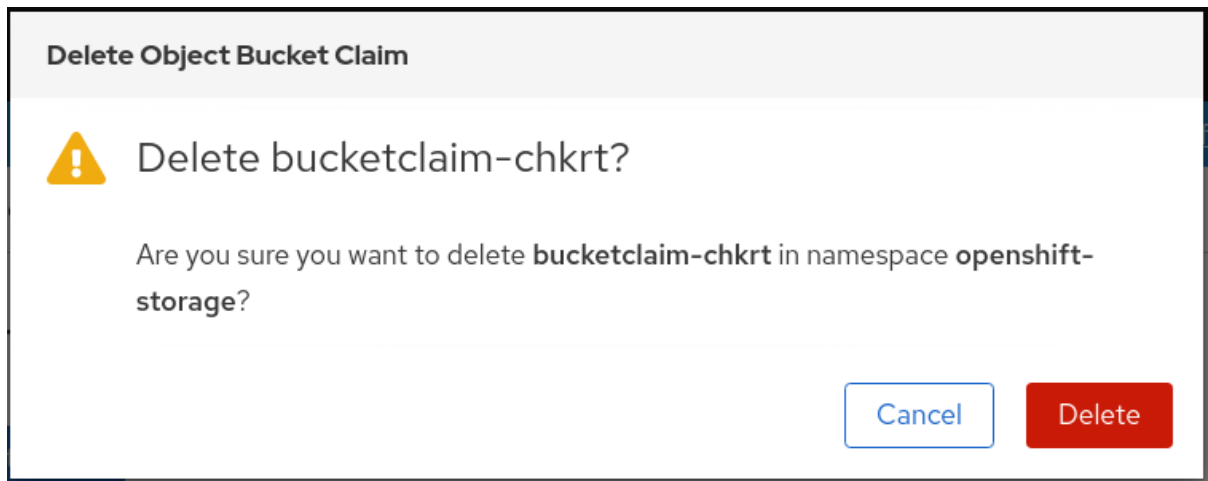
[Create Object Bucket Claim](#) Filter by name...

0 Pending 1 Bound 0 Lost [Select All Filters](#) 1 Item

Name	Namespace	Status	Secret	Storage Class
OBC bucketclaim-chkrt	NS openshift-storage	✔ Bound	S bucketclaim-chkrt	SC openshift-storage.noobaa.io

- Attach to Deployment
- Edit Labels
- Edit Annotations
- Edit Object Bucket Claim
- Delete Object Bucket Claim

- メニューから **Delete Object Bucket Claim** を選択します。



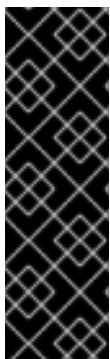
4. **Delete** をクリックします。

関連情報

- [「Object Bucket Claim\(オブジェクトバケット要求\)」](#)

10.9. 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](#)

10.9.1. 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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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 Container Storage RPM からインストールできます。

手順

1. NamespaceStore リソースを作成します。NamespaceStore は、Multicloud Object Gateway 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、Multicloud Object Gateway に対して、バックングストア、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。
- 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>** を関連するバックキングストアに置き換えます。コンマで区切られた1つ以上のバック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。
- 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。

10.9.2. 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-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

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 Container Storage RPM からインストールできます。

手順

- NamespaceStore リソースを作成します。NamespaceStore は、Multicloud Object Gateway 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、Multicloud Object Gateway に対して、バックングストア、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>** を関連するバックキングストアに置き換えます。コンマで区切られた1つ以上のバック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。

10.10. 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ます、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 デモンコンテナのグループです。

- ストレージサービス
- S3 エンドポイントサービス

10.10.1. Multicloud Object Gateway での S3 エンドポイント

S3 エンドポイントは、すべての Multicloud Object Gateway がデフォルトで提供するサービスであり、これは Multicloud Object Gateway で負荷の高いデータ消費タスクの大部分を処理します。エンドポイントサービスは、インラインのデータチャンク、重複排除、圧縮、および暗号化を処理し、Multicloud Object Gateway からのデータ配置の指示を受け入れます。

10.10.2. ストレージノードを使用したスケーリング

前提条件

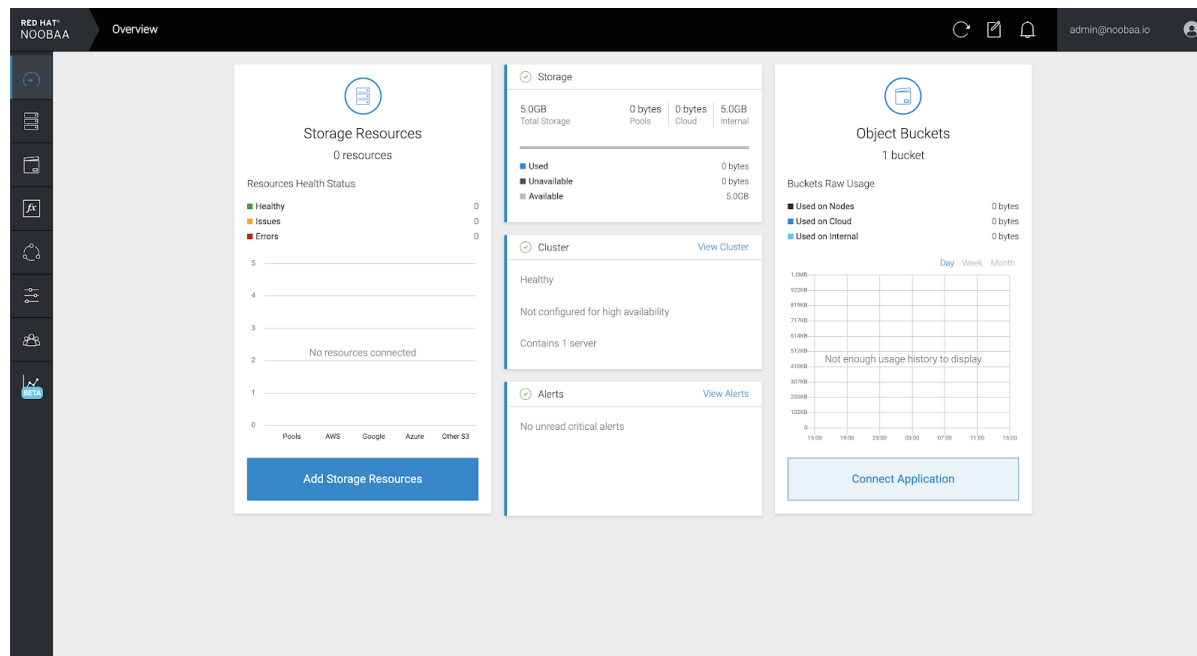
- Multicloud Object Gateway へのアクセスのある OpenShift Container Platform で実行中の OpenShift Container Storage Platform

Multicloud Object Gateway のストレージノードは1つ以上の永続ボリュームに割り当てられた NooBaa

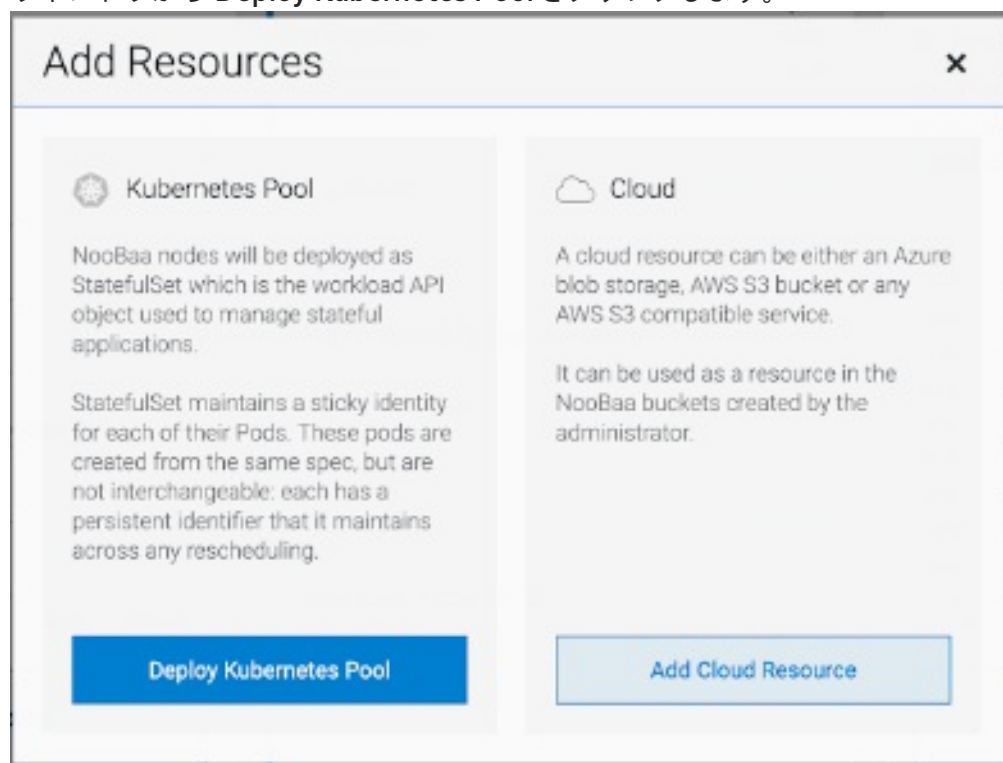
デーモンコンテナであり、ローカルオブジェクトサービスデータストレージに使用されます。NooBaa デーモンは Kubernetes ノードにデプロイできます。これは、StatefulSet Pod で設定される Kubernetes プールを作成して実行できます。

手順

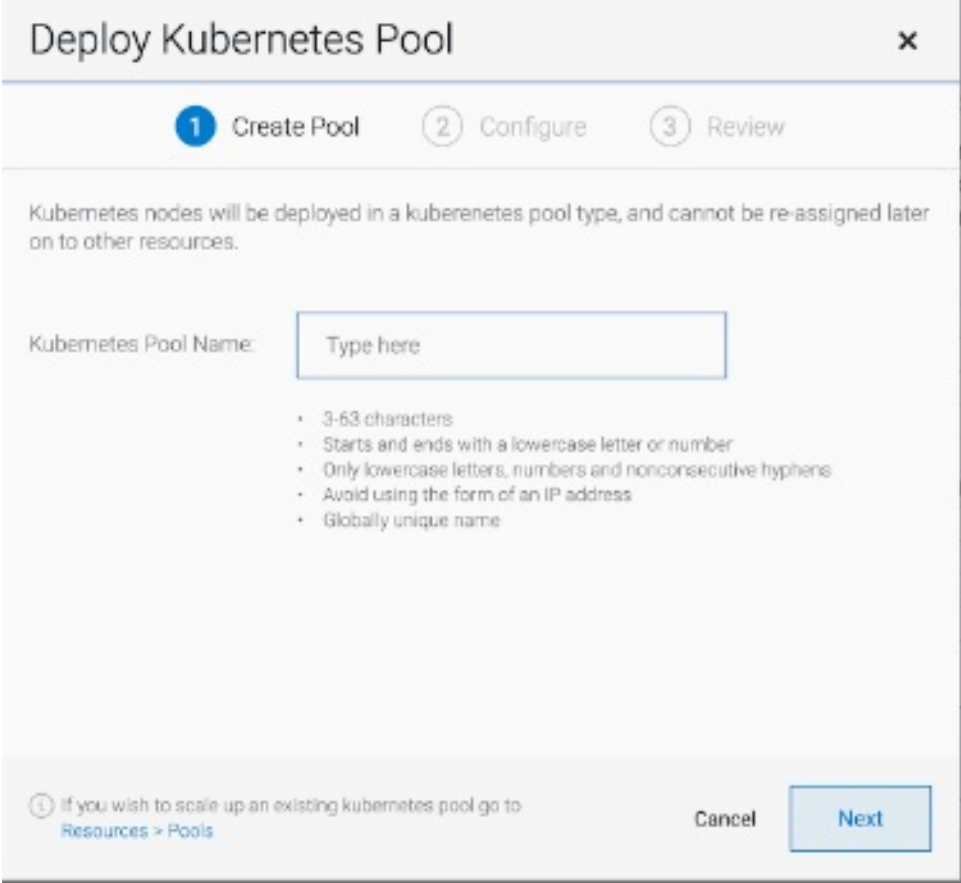
1. Mult-Cloud Object Gateway ユーザーインターフェイスの **Overview** ページで、**Add Storage Resources** をクリックします。



2. ウィンドウから **Deploy Kubernetes Pool** をクリックします。



3. **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。



Deploy Kubernetes Pool [Close]

1 Create Pool 2 Configure 3 Review

Kubernetes nodes will be deployed in a kubernetes pool type, and cannot be re-assigned later on to other resources.

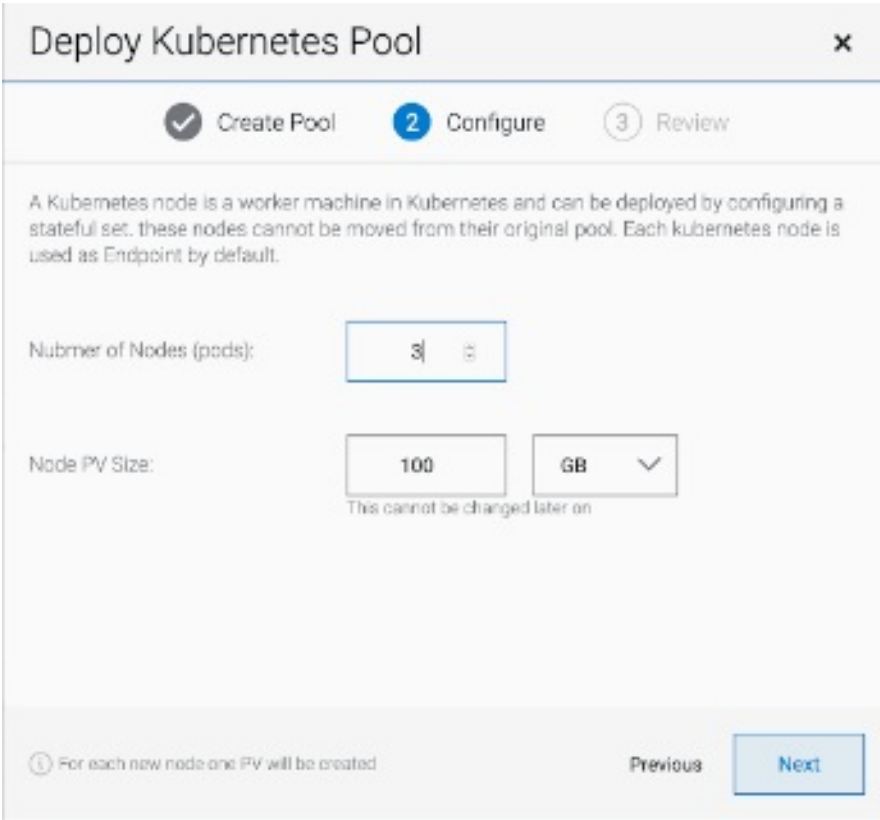
Kubernetes Pool Name:

- 3-63 characters
- Starts and ends with a lowercase letter or number
- Only lowercase letters, numbers and nonconsecutive hyphens
- Avoid using the form of an IP address
- Globally unique name

① If you wish to scale up an existing kubernetes pool go to [Resources > Pools](#)

Cancel Next

4. **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。



Deploy Kubernetes Pool [Close]

✓ Create Pool 2 Configure 3 Review

A Kubernetes node is a worker machine in Kubernetes and can be deployed by configuring a stateful set. these nodes cannot be moved from their original pool. Each kubernetes node is used as Endpoint by default.

Number of Nodes (pods):

Node PV Size: [v]

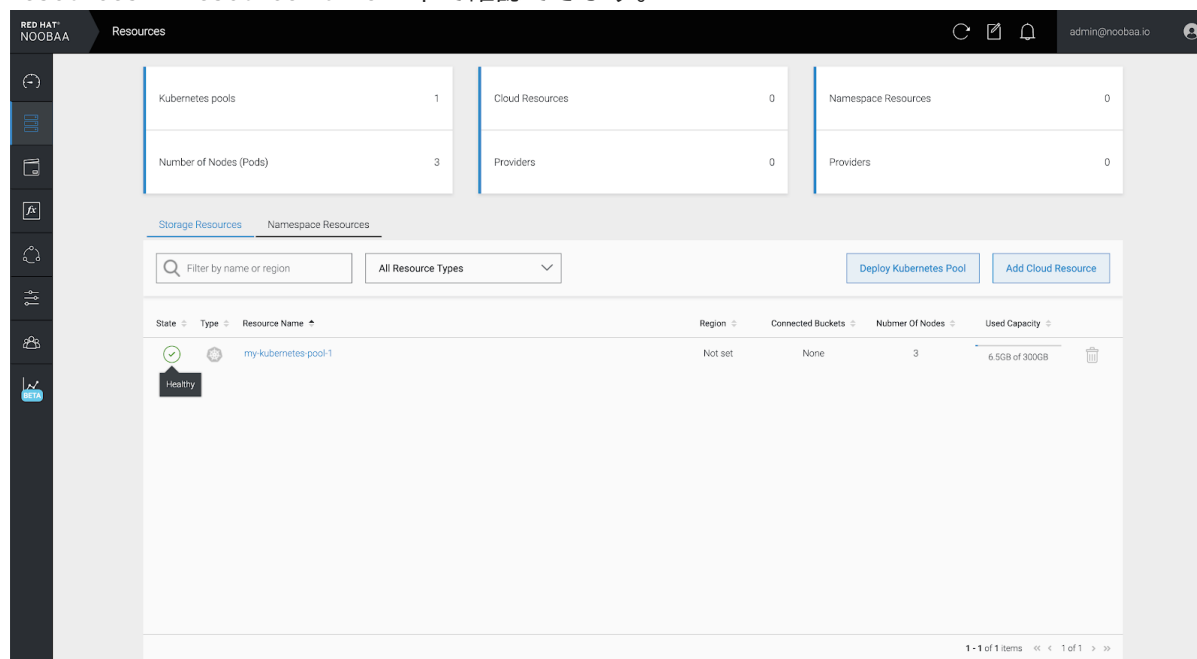
This cannot be changed later on

① For each new node one PV will be created

Previous Next

5. **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。

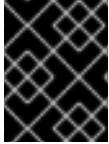
6. すべてのノードは最初の手順で選択したプールに割り当てられ、**Resources → Storage resources → Resource name** の下で確認できます。



10.11. 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。{product-name-short} クラスターは、1つのアクティブな MCG エンドポイントでデプロイされます。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 Container Storage がサポートする PVC ではサポートされません。

11.1. OPENSIFT CONTAINER PLATFORM 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 Container Storage 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 Container Storage 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。
- OpenShift Container Storage 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** → **Storage Class** 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 Container Storage 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。
 - iv. アプリケーション要件に応じて **Size** を指定します。
 - v. **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:** セクションで、**volume:** 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 menu (⋮)** → **Edit Deployment Config** をクリックします。
 - iv. **spec:** セクションで、**volume:** 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 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: **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 Container Storage 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 コンソールにログインします。
2. **Home** → **Overview** → **Persistent Storage** 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。これらはすべて管理者によって静的にプロビジョニングされますが、一部のストレージタイプは組み込みプロバイダーとプラグイン API を使用して動的に作成できます。

11.4.2. OpenShift Container Storage の動的プロビジョニング

Red Hat OpenShift Container Storage は、コンテナ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。これは OpenShift Container Platform の Operator として実行され、コンテナの統合され、単純化された永続ストレージの管理を可能にします。

OpenShift Container Storage 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 Container Storage 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。

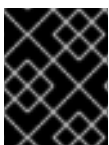
使用するドライバー (RBD または CephFS) の判断は、**storageclass.yaml** ファイルのエントリーに基づいて行われます。

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	

ストレージタイプ	プロビジョナープラグインの名前	注記
AWS Elastic 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	



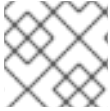
重要

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があります。

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 Container Storage 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、または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。

- OpenShift Web コンソールで、**Storage** → **Persistent Volume Claims** をクリックします。
- 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。
- ボリュームスナップショットの **Name** を入力します。
- ドロップダウンリストから **Snapshot Class** を選択します。
- 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。

- 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. 任意の **Access Mode** を選択します。



重要

ReadOnlyMany (ROX) アクセスモードは Developer プレビュー機能であり、Developer プレビューのサポート制限の対象となります。Developer 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場合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. **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。

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。

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. 任意の **Access Mode** を選択します。



重要

ReadOnlyMany (ROX) アクセスモードは Developer プレビュー機能であり、Developer プレビューのサポート制限の対象となります。Developer 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場合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。

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。

7. **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わ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. 削除する必要が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。

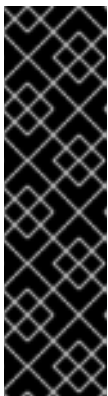


注記

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がある 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 Developer プレビュー機能であり、Developer プレビューのサポート制限の対象となります。Developer 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場合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。

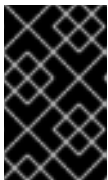
手順

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。

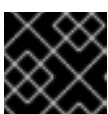
```
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets
```



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 Container Storage 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 Container Storage 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 Container Storage 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) クラスタでクラスタ全体の暗号化が有効な場合には、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. 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。
 - 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>
$ chroot /host
```

- ii. `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
```

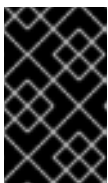
6. 検証手順が失敗した場合は、[Red Hat サポートにお問い合わせください](#)。

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 Container Storage 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 Container Storage 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 Container Storage 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 Container Storage 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には、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. 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。
 - 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>  
$ chroot /host
```

- ii. 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 OPENSIFT CONTAINER STORAGE の更新

16.1. OPENSIFT CONTAINER STORAGE 更新プロセスの概要

Red Hat OpenShift Container Storage およびそのコンポーネントを (4.6 と 4.7 間などのマイナーリリース間、または 4.7.0 と 4.7.1 間などのバッチ更新間でアップグレードできます。

OpenShift Container Storage の異なる部分を特定の順序でアップグレードする必要があります。

1. OpenShift Container Platform の [Updating clusters](#) ドキュメントに従って **OpenShift Container Platform** を更新します。
2. **OpenShift Container Storage** を更新します。
 - a. **更新に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のための [Operator ガイド](#) を参照し、OpenShift Container Storage 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. お使いのセットアップに適したプロセスを使用して、**OpenShift Container Storage Operator** を更新します。
 - [内部モードでの OpenShift Container Storage の更新](#)

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 では、Red Hat OpenShift Container Storage で同じバージョンの Red Hat OpenShift Container Platform を使用することを推奨しています。
OpenShift Container Platform および OpenShift Container Storage のサポートされる組み合わせについての詳細は、[相互運用性マトリックス](#) 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 バージョンが Red Hat OpenShift Container Platform 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。

16.2. 非接続環境での更新の準備

Red Hat OpenShift Container Storage 環境がインターネットに直接接続されていない場合には、デフォルトの Operator Hub およびイメージレジストリーの代替オプションとして Operator Lifecycle Manager(OLM) を提供するために追加の設定が必要になります。

概要については OpenShift Container Platform ドキュメント [Operator カタログイメージの更新](#) を参照してください。

クラスターで非接続更新を設定するには、以下を実行します。

1. [代替レジストリーの認証を設定します](#)。
2. [Red Hat Operator カタログをビルドし、ミラーリングします](#)。
3. [Operator imageContentSourcePolicy](#) を作成します。
4. [redhat-operator catalogsource](#) を更新します。

これらの手順が完了したら、通常通りに [更新を継続](#) します。

16.2.1. ミラーレジストリーの認証情報の追加

前提条件

- 既存の非接続クラスターが OpenShift Container Platform 4.3 以降を使用していることを確認します。
- **oc client** がバージョン 4.4 以降であることを確認します。
- ミラーレジストリーでミラーホストを準備します。詳細は、[ミラーホストの準備](#) について参照してください。

手順

1. **cluster-admin** ロールを使用して OpenShift Container Platform クラスターにログインします。
2. **auth.json** ファイルを見つけます。
このファイルは、podman または docker を使用してレジストリーにログインする際に生成されます。これは、以下のいずれかの場所にあります。
 - **~/.docker/auth.json**
 - **/run/user/<UID>/containers/auth.json**
 - **/var/run/containers/<UID>/auth.json**
3. 一意の Red Hat レジストリー [プルシークレット](#) を取得して **auth.json** ファイルに貼り付けます。以下ようになります。

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "*****",
      "email": "user@example.com"
    },
    "quay.io": {
      "auth": "*****",
      "email": "user@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "*****",
      "email": "user@example.com"
    },
    "registry.redhat.io": {
      "auth": "*****",
      "email": "user@example.com"
    }
  }
}
```

4. 設定に対応する詳細と共に環境変数をエクスポートします。

```
$ export AUTH_FILE="<location_of_auth.json>"
$ export MIRROR_REGISTRY_DNS="<your_registry_url>:<port>"
```

5. **podman** を使用してミラーレジストリーにログインし、認証情報を **\${AUTH_FILE}** に保存します。

```
$ podman login ${MIRROR_REGISTRY_DNS} --tls-verify=false --authfile ${AUTH_FILE}
```

これにより、ミラーレジストリーが **auth.json** ファイルに追加されます。

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "*****",
      "email": "user@example.com"
    },
    "quay.io": {
      "auth": "*****",
      "email": "user@example.com"
    },
    "registry.connect.redhat.com": {
      "auth": "*****",
      "email": "user@example.com"
    },
    "registry.redhat.io": {
      "auth": "*****",
      "email": "user@example.com"
    },
    "<mirror_registry>": {
      "auth": "*****",
    }
  }
}
```

16.2.2. Red Hat Operator カタログのビルドおよびミラーリング

Red Hat レジストリーにアクセスできるホストでこのプロセスを実行し、それらのレジストリーのミラーを作成します。

前提条件

- これらのコマンドをクラスター管理者として実行します。
- **redhat-operator** カタログのミラーリングには完了するまでに時間がかかる場合があります。また、ミラーホストに大きなディスク領域が利用可能である必要があることに注意してください。

手順

1. **redhat-operators** のカタログをビルドします。
ターゲット OpenShift Container Platform クラスターのメジャーバージョンおよびマイナーバージョンに一致するタグを使用して、**--from** を **ose-operator-registry** ベースイメージに設定します。

```
$ oc adm catalog build --appregistry-org redhat-operators \
  --from=registry.redhat.io/openshift4/ose-operator-registry:v4.7 \
  --to=${MIRROR_REGISTRY_DNS}/olm/redhat-operators:v2 \
```

```
--registry-config=${AUTH_FILE} \
--filter-by-os="linux/amd64" --insecure
```

2. **redhat-operators** のカタログをミラーリングします。

これは長時間の操作となり、1-5 時間の時間がかかる場合があります。ミラーホストに 100 GB の空きディスク領域があることを確認します。

```
$ oc adm catalog mirror ${MIRROR_REGISTRY_DNS}/olm/redhat-operators:v2 \
${MIRROR_REGISTRY_DNS} --registry-config=${AUTH_FILE} --insecure
```

16.2.3. Operator imageContentSourcePolicy を作成します。

oc adm catalog mirror コマンドが完了すると、**imageContentSourcePolicy.yaml** ファイルが作成されます。通常、このファイルの出力ディレクトリーは **./[catalog image name]-manifests** です。以下の手順を使用して、不足しているエントリーを **.yaml** ファイルに追加し、それらをクラスターに適用します。

手順

1. このファイルの内容で、以下のようにミラーマッピングを確認します。

```
spec:
  repositoryDigestMirrors:
    - mirrors:
      - <your_registry>/ocs4
      source: registry.redhat.io/ocs4
    - mirrors:
      - <your_registry>/rhceph
      source: registry.redhat.io/rhceph
    - mirrors:
      - <your_registry>/openshift4
      source: registry.redhat.io/openshift4
    - mirrors:
      - <your_registry>/rhsc1
      source: registry.redhat.io/rhsc1
```

2. 不足しているエントリーを **imageContentSourcePolicy.yaml** ファイルの最後に追加します。
3. **imageContentSourcePolicy.yaml** ファイルをクラスターに適用します。

```
$ oc apply -f ./[output dir]/imageContentSourcePolicy.yaml
```

Image Content Source Policy を更新したら、クラスター内のすべてのノード (マスター、インフラストラクチャー、およびワーカー) を更新し、再起動する必要があります。このプロセスは Machine Config Pool Operator で自動的に処理され、実際の経過時間は OpenShift クラスターのノード数によって異なる可能性があります、最長で 30 分の時間がかかります。**oc get mcp** コマンドまたは **oc get node** コマンドを使用して更新プロセスをモニターできます。

16.2.4. redhat-operator CatalogSource の更新

手順

代替カタログソースを設定した後も、適切な更新プロセスを続行できます。

- [内部モードでの OpenShift Container Storage の更新](#)

16.3. 内部モードでの OPENSIFT CONTAINER STORAGE の更新

以下の手順に従って、内部モードでデプロイされた OpenShift Container Storage クラスターを更新します。

16.3.1. 内部モードでの OpenShift Container Storage Operator の自動更新の有効化

以下の手順を使用して、OpenShift Container Platform で OpenShift Container Storage Operator の自動の更新承認を有効にします。

前提条件

- Status カードの **Persistent Storage** で、**OCS Cluster** および **Data Resiliency** に緑色のチェックマークが付いていることを確認します。
- Status カードの **Object Service** で、**Object Service** および **Data Resiliency** の両方が **Ready** 状態 (緑のチェックマーク) にあることを確認します。
- OpenShift Container Platform クラスターをバージョン 4.7.Y の最新の安定したリリースに更新する場合は、[Updating Clusters](#) を参照してください。
- Red Hat OpenShift Container Storage チャンネルを **stable-4.6** から **stable-4.7** に切り替えます。チャンネルの詳細は、[OpenShift Container Storage upgrade channels and releases](#) を参照してください。



注記

マイナーバージョンを更新する場合 (例: 4.6 から 4.7 に更新) にのみチャンネルを切り換える必要があり、4.7 のバッチの更新間に更新する場合 (例: 4.7.0 から 4.7.1 に更新) はチャンネルを切り換える必要はありません。

- Operator Pod を含むすべての OpenShift Container Storage Pod が **openshift-storage namespace** で **Running** 状態にあることを確認します。
Pod の状態を確認するには、OpenShift Web コンソールの左側のペインから **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。
- 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 Container Storage 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 コンソールにログインします。
2. **Operators** → **Installed Operators** をクリックします。
3. **openshift-storage** プロジェクトを選択します。
4. OpenShift Container Storage Operator 名をクリックします。
5. **Subscription** タブをクリックしてから、**Approval** の下にあるリンクをクリックします。

6. **Automatic (default)** を選択し、**Save** をクリックします。
7. **Upgrade Status** に応じて以下のいずれかを実行します。
 - **Upgrade Status** には、**requires approval** と表示されます。



注記

Upgrade status には、新規 OpenShift Container Storage バージョンがチャネルですでに検知され、承認ストラテジーが更新時に **Manual** から **Automatic** に変更されている場合に **requires approval** が表示されます。

- a. **Install Plan** リンクをクリックします。
 - b. **InstallPlan Details** ページで、**Preview Install Plan** をクリックします。
 - c. インストール計画を確認し、**Approve** をクリックします。
 - d. **Status** が **Unknown** から **Created** に変更されるまで待機します。
 - e. **Operators** → **Installed Operators** をクリックします。
 - f. **openshift-storage** プロジェクトを選択します。
 - g. **Status** が **Up to date** に変更するまで待機します。
- **Upgrade Status** には、**requires approval** は表示されません。
 - a. 更新が開始するまで待機します。これには、最長 20 分の時間がかかる可能性があります。
 - b. **Operators** → **Installed Operators** をクリックします。
 - c. **openshift-storage** プロジェクトを選択します。
 - d. **Status** が **Up to date** に変更するまで待機します。



注記

NooBaa DB を MongoDB から PostgreSQL に移行するため、アップグレード中は Multicloud Object Gateway の停止が短期間予想されます。

検証手順

1. **Status** カードで **Overview** → **Persistent Storage** タブをクリックし、**OCS Cluster** および **Data Resiliency** で正常であることを示す緑色のチェックマークが表示されていることを確認します。
2. **Overview** → **Object Service** タブをクリックし、**Status** カードで、**Object Service** と **Data Resiliency** の両方が正常なことを示す **Ready** 状態 (Green tick) であることを確認します。
3. **Operators** → **Installed Operators** → **OpenShift Container Storage Operator** をクリックします。**Storage Cluster** で、クラスターサービスのステータスが **Ready** であることを確認します。



注記

OpenShift Container Storage バージョン 4.6 から 4.7 に更新された後も、**Version** フィールドには依然として 4.6 が表示されます。これは、**ocs-operator** がこのフィールドで表示される文字列を更新しないためです。

4. Operator Pod を含むすべての OpenShift Container Storage Pod が **openshift-storage namespace** で **Running** 状態にあることを確認します。
Pod の状態を表示するには、**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。
5. 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。



注記

柔軟なスケーリング機能は、Red Hat OpenShift Container Storage 4.7 の新規デプロイメントでのみ利用できます。4.7 バージョンにアップグレードされたストレージクラスターは、柔軟なスケーリングをサポートしていません。

次のステップ

- [既存のバックングストアへのアノテーションの追加](#)

関連情報

OpenShift Container Storage の更新中に問題が発生した場合は、[Troubleshooting guide](#) の **Commonly required logs for troubleshooting** セクションを参照してください。

16.3.2. 内部モードでの OpenShift Container Storage Operator の手動による更新

以下の手順を使用して、インストール計画に手動の承認を指定し、OpenShift Container Storage Operator を更新します。

前提条件

- **Status** カードの **Persistent Storage** で、**OCS Cluster** および **Data Resiliency** に緑色のチェックマークが付いていることを確認します。
- **Status** カードの **Object Service** で、**Object Service** および **Data Resiliency** の両方が **Ready** 状態 (緑のチェックマーク) にあることを確認します。
- OpenShift Container Platform クラスターをバージョン 4.7.Y の最新の安定したリリースに更新する場合は、[Updating Clusters](#) を参照してください。
- Red Hat OpenShift Container Storage チャンネルを **stable-4.6** から **stable-4.7** に切り替えます。チャンネルの詳細は、[OpenShift Container Storage upgrade channels and releases](#) を参照してください。



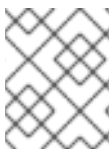
注記

マイナーバージョンを更新する場合 (例: 4.6 から 4.7 に更新) にのみチャンネルを切り換える必要があり、4.7 のバッチの更新間に更新する場合 (例: 4.7.0 から 4.7.1 に更新) はチャンネルを切り換える必要はありません。

- Operator Pod を含むすべての OpenShift Container Storage Pod が **openshift-storage namespace** で **Running** 状態にあることを確認します。
Pod の状態を確認するには、OpenShift Web コンソールの左側のペインから **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。
- 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 Container Storage 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 コンソールにログインします。
2. **Operators** → **Installed Operators** をクリックします。
3. **openshift-storage** プロジェクトを選択します。
4. **OpenShift Container Storage Operator** 名をクリックします。
5. **Subscription** タブをクリックしてから、**Approval** の下にあるリンクをクリックします。
6. **Manual** を選択し、**Save** をクリックします。
7. **Upgrade Status** が **Upgrading** に変更するまで待機します。
8. **Upgrade Status** に **requires approval** が表示される場合は、**requires approval** をクリックします。
9. **InstallPlan Details** ページで、**Preview Install Plan** をクリックします。
10. インストール計画を確認し、**Approve** をクリックします。
11. **Status** が **Unknown** から **Created** に変更されるまで待機します。
12. **Operators** → **Installed Operators** をクリックします。
13. **openshift-storage** プロジェクトを選択します。
14. **Status** が **Up to date** に変更するまで待機します。



注記

NooBaa DB を MongoDB から PostgreSQL に移行するため、アップグレード中は Multicloud Object Gateway の停止が短期間予想されます。

検証手順

1. **Status** カードで **Overview** → **Persistent Storage** タブをクリックし、**OCS Cluster** および **Data Resiliency** で正常であることを示す緑色のチェックマークが表示されていることを確認します。
2. **Overview** → **Object Service** タブをクリックし、**Status** カードで、**Object Service** と **Data Resiliency** の両方が正常なことを示す **Ready** 状態 (Green tick) であることを確認します。
3. **Operators** → **Installed Operators** → **OpenShift Container Storage Operator** をクリックします。**Storage Cluster** で、クラスターサービスのステータスが **Ready** であることを確認します。



注記

OpenShift Container Storage バージョン 4.6 から 4.7 に更新された後も、**Version** フィールドには依然として 4.6 が表示されます。これは、**ocs-operator** がこのフィールドで表示される文字列を更新しないためです。

4. Operator Pod を含むすべての OpenShift Container Storage Pod が **openshift-storage namespace** で **Running** 状態にあることを確認します。
Pod の状態を確認するには、OpenShift Web コンソールの左側のペインから **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。
5. 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。

次のステップ

- [既存のバックングストアへのアノテーションの追加](#)

関連情報

OpenShift Container Storage 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](#)で一般に必要な**ログ** セクションを参照してください。