



# Red Hat OpenShift Container Storage 4.6

## Red Hat OpenStack Platform 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 4.6 Red Hat OpenStack Platform 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\_Red\_Hat\_OpenStack\_Platform 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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

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

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> 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<sup>®</sup> 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

Red Hat OpenStack Platform (RHOSP) 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 Red Hat OpenStack Platform 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

前書き .....	5
<b>第1章 内部モードでの OPENSIFT CONTAINER STORAGE の RED HAT OPENSTACK PLATFORM へのデプロイ</b>	<b>6</b>
1.1. RED HAT OPENSIFT CONTAINER STORAGE OPERATOR のインストール	6
1.2. 内部モードでの OPENSIFT CONTAINER STORAGE CLUSTER SERVICE の作成	9
1.3. OPENSIFT CONTAINER STORAGE デプロイメントの検証	11
1.3.1. Pod の状態の確認	11
1.3.2. OpenShift Container Storage クラスターが正常であることの確認	13
1.3.3. Multicloud Object Gateway が正常であることの確認	13
1.3.4. OpenShift Container Storage 固有のストレージクラスが存在することの確認	14
1.4. 内部モードでの OPENSIFT CONTAINER STORAGE のアンインストール	14
1.4.1. 内部モードでの OpenShift Container Storage のアンインストール	14
1.4.1.1. OpenShift Container Storage からのモニターリングスタックの削除	19
1.4.1.2. OpenShift Container Storage からの OpenShift Container Platform レジストリーの削除	22
1.4.1.3. OpenShift Container Storage からのクラスターロギング Operator の削除	23
<b>第2章 外部モードでの OPENSIFT CONTAINER STORAGE の RED HAT OPENSTACK PLATFORM へのデプロイ</b>	<b>25</b>
2.1. RED HAT OPENSIFT CONTAINER STORAGE OPERATOR のインストール	25
2.2. 外部モードでの OPENSIFT CONTAINER STORAGE CLUSTER SERVICE の作成	28
2.3. 外部モードでの OPENSIFT CONTAINER STORAGE インストールの確認	33
2.3.1. Pod の状態の確認	33
2.3.2. OpenShift Container Storage クラスターが正常であることの確認	34
2.3.3. Multicloud Object Gateway が正常であることの確認	35
2.3.4. ストレージクラスが作成され、一覧表示されることの確認	35
2.3.5. Ceph クラスターが接続されていることの確認	36
2.3.6. ストレージクラスターの準備が整っていることを確認します。	36
2.4. 外部モードでの OPENSIFT CONTAINER STORAGE のアンインストール	36
2.4.1. 外部モードでの OpenShift Container Storage のアンインストール	36
2.4.2. OpenShift Container Storage からのモニターリングスタックの削除	40
2.4.3. OpenShift Container Storage からの OpenShift Container Platform レジストリーの削除	44
2.4.4. OpenShift Container Storage からのクラスターロギング Operator の削除	45
<b>第3章 ストレージクラスおよびストレージプール .....</b>	<b>47</b>
3.1. ストレージクラスおよびプールの作成	47
<b>第4章 OPENSIFT CONTAINER PLATFORM サービスのストレージの設定 .....</b>	<b>49</b>
4.1. OPENSIFT CONTAINER STORAGE を使用するためのイメージレジストリーの設定	49
4.2. OPENSIFT CONTAINER STORAGE を使用するためのモニターリングの設定	51
4.3. OPENSIFT CONTAINER STORAGE のクラスターロギング	54
4.3.1. 永続ストレージの設定	54
4.3.2. OpenShift Container Storage を使用するためのクラスターロギングの設定	55
<b>第5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 .....</b>	<b>58</b>
<b>第6章 RED HAT OPENSIFT CONTAINER STORAGE に専用のワーカーノードを使用する方法 .....</b>	<b>60</b>
6.1. インフラストラクチャーノードの仕組み	60
6.2. 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	60
6.3. インフラストラクチャーノードの手動作成	61
<b>第7章 ストレージノードのスケーリング .....</b>	<b>63</b>
7.1. ストレージノードのスケーリングの要件	63

7.2. RED HAT OPENSTACK PLATFORM インフラストラクチャー上の OPENSIFT CONTAINER STORAGE ノードへの容量追加によるストレージのスケールアップ	63
7.3. 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	66
7.3.1. 新規ノードの追加の確認	67
7.3.2. ストレージ容量のスケールアップ	67
<b>第8章 MULTICLOUD OBJECT GATEWAY</b>	<b>68</b>
8.1. MULTICLOUD OBJECT GATEWAY について	68
8.2. アプリケーションの使用による MULTICLOUD OBJECT GATEWAY へのアクセス	68
8.2.1. ターミナルから Multicloud Object Gateway へのアクセス	69
8.2.2. 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	71
8.3. ハイブリッドまたはマルチクラウド用のストレージリソースの追加	73
8.3.1. 新規バックスタアの作成	73
8.3.2. MCG コマンドラインインターフェイスを使用したハイブリッドまたはマルチクラウドのストレージリソースの追加	75
8.3.2.1. AWS でサポートされるバックスタアの作成	75
8.3.2.2. IBM COS でサポートされるバックスタアの作成	77
8.3.2.3. Azure でサポートされるバックスタアの作成	79
8.3.2.4. GCP でサポートされるバックスタアの作成	80
8.3.2.5. ローカル永続ボリュームでサポートされるバックスタアの作成	81
8.3.3. s3 と互換性のある Multicloud Object Gateway バックスタアの作成	83
8.3.4. ユーザーインターフェイスを使用したハイブリッドおよびマルチクラウドのストレージリソースの追加	84
8.3.5. 新規バケットクラスの作成	86
8.4. NAMESPACE バケットの設定	89
8.4.1. プロバイダー接続の Multicloud Object Gateway への追加	89
8.4.2. Multicloud Object Gateway を使用した namespace リソースの追加	90
8.4.3. Multicloud Object Gateway を使用した namespace バケットへのリソースの追加	90
8.4.4. namespace バケットのオブジェクトの Amazon S3 API エンドポイント	91
8.5. ハイブリッドおよびマルチクラウドバケットのデータのミラーリング	92
8.5.1. MCG コマンドラインインターフェイスを使用したデータのミラーリング用のバケットクラスの作成	92
8.5.2. YAML を使用したデータのミラーリング用のバケットクラスの作成	93
8.5.3. 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	93
8.6. MULTICLOUD OBJECT GATEWAY のバケットポリシー	95
8.6.1. バケットポリシーについて	95
8.6.2. バケットポリシーの使用	96
8.6.3. Multicloud Object Gateway での AWS S3 ユーザーの作成	97
8.7. OBJECT BUCKET CLAIM(オブジェクトバケット要求)	99
8.7.1. 動的 Object Bucket Claim(オブジェクトバケット要求)	99
8.7.2. コマンドラインインターフェイスを使用した Object Bucket Claim(オブジェクトバケット要求) の作成	101
8.7.3. OpenShift Web コンソールを使用した Object Bucket Claim(オブジェクトバケット要求) の作成	104
8.8. 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ルアップ	107
8.8.1. Multicloud Object Gateway での S3 エンドポイント	107
8.8.2. ストレージノードを使用したスケールアップ	107
<b>第9章 永続ボリューム要求の管理</b>	<b>111</b>
9.1. OPENSIFT CONTAINER PLATFORM を使用するためのアプリケーション POD の設定	111
9.2. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) 要求ステータスの表示	112
9.3. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) 要求イベントの確認	113
9.4. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) の拡張	113
9.5. 動的プロビジョニング	115
9.5.1. 動的プロビジョニングについて	115
9.5.2. OpenShift Container Storage の動的プロビジョニング	115

---

9.5.3. 利用可能な動的プロビジョニングプラグイン	116
<b>第10章 ボリュームスナップショット</b>	<b>118</b>
10.1. ボリュームスナップショットの作成	118
10.2. ボリュームスナップショットの復元	119
10.3. ボリュームスナップショットの削除	120
<b>第11章 ボリュームのクローン作成</b>	<b>122</b>
11.1. クローンの作成	122
<b>第12章 ストレージノードの置き換え</b>	<b>123</b>
12.1. RED HAT OPENSTACK PLATFORM 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
12.2. RED HAT OPENSTACK PLATFORM 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障害のあるノードの置き換え	124
<b>第13章 ストレージデバイスの置き換え</b>	<b>127</b>
13.1. RED HAT OPENSTACK PLATFORM 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障害のあるストレージデバイスの置き換え	127
<b>第14章 OPENSIFT CONTAINER STORAGE の更新</b>	<b>132</b>
14.1. OPENSIFT CONTAINER STORAGE 更新プロセスの概要	132
14.2. 非接続環境での更新の準備	132
14.2.1. ミラーレジストリーの認証情報の追加	133
14.2.2. Red Hat Operator カタログのビルドおよびミラーリング	134
14.2.3. Operator imageContentSourcePolicy を作成します。	135
14.2.4. redhat-operator CatalogSource の更新	135
14.2.5. 更新の継続	136
14.3. 内部モードでの OPENSIFT CONTAINER STORAGE の更新	137
14.3.1. 内部モードでの OpenShift Container Storage Operator の自動更新の有効化	137
14.3.2. 内部モードでの OpenShift Container Storage Operator の手動による更新	139



## 前書き

Red Hat OpenShift Container Storage 4.6 では、Red Hat OpenStack Platform を使用した既存の Red Hat OpenShift Container Platform (RHOCP) でのデプロイメントをサポートします。



### 注記

Red Hat OpenStack Platform では、内部および外部の Openshift Container Storage クラスタ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 をデプロイするには、お使いの環境に適切なデプロイメントプロセスを実行します。

- 内部モード

[内部モードでの OpenShift Container Storage の Red Hat OpenStack Platform へのデプロイ](#)

- 外部モード

[外部モードでの OpenShift Container Storage の Red Hat OpenStack Platform へのデプロイ](#)

# 第1章 内部モードでの OPENSIFT CONTAINER STORAGE の RED HAT OPENSTACK PLATFORM へのデプロイ

Red Hat OpenStack Platform 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内部モードで 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。

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

## 1.1. 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。ハードウェアおよびソフトウェアの要件に関する詳細は、[デプロイメントのプランニング](#) を参照してください。

### 前提条件

- OpenShift Container Platform (RHOCP) クラスターにログインしている必要があります。
- RHOCP クラスターにワーカーノードが少なくとも 3 つ必要です。

### 注記

- OpenShift Container Storage のクラスター全体でのデフォルトノードセレクターを上書きする必要がある場合は、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 の空のノードセレクターを指定できます。

```
$ 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 の専用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. OpenShift Web コンソールの左側のペインで、**Operators → OperatorHub** をクリックします。
2. **Filter by keyword** テキストボックスまたはフィルター一覧を使用して、Operator の一覧から OpenShift Container Storage を検索します。
3. **OpenShift Container Storage** をクリックします。
4. **OpenShift Container Storage Operator** ページで、**Install** をクリックします。

5. **Install Operator** ページで、以下のオプションがデフォルトで選択されていることを確認します。
  - a. Channel を **stable-4.6**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. **Enable operator recommended cluster monitoring on this namespace**が選択されていることを確認します。この設定はクラスタのモニタリングに必要です。
  - e. **承認ストラテジー** を **Automatic** または **Manual** として選択している。承認ストラテジーはデフォルトで **Automatic** に設定されます。
    - **Approval Strategy** に **Automatic** を選択します。



#### 注記

Approval Strategy を **Automatic** として選択すると、新規インストール時、または OpenShift Container Storage の最新バージョンへの更新時に承認は必要ありません。

- i. **インストール** をクリックします。
  - ii. インストール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。
  - iii. **Operators** → **Installed Operators** をクリックします。
  - iv. **Project** が **openshift-storage** であることを確認します。デフォルトで、**プロジェクト** は **openshift-storage** です。
  - v. **OpenShift Container Storage** の **Status** が **Succeeded** に変更するまで待機します。
- **Approval Strategy** に **Manual** を選択します。



#### 注記

Approval Strategy を **Manual** として選択すると、新規インストール時、または OpenShift Container Storage の最新バージョンへの更新時に承認が必要になります。

- i. **インストール** をクリックします。
- ii. **Manual approval required** ページで、**Approve** または **View Installed Operators in namespace openshift-storage** のいずれかをクリックし、Operator をインストールできます。



### 重要

オプションのいずれかをクリックする前に、**Manual approval required** ページで、インストール計画がウィンドウに読み込まれるまで数分待機します。



### 重要

**Approve** をクリックする場合は、続行する前にインストール計画を確認する必要があります。

- **Approve** をクリックします。
  - OpenShift Container Storage Operator がインストールされている間に数分待機します。
  - **Installed operator - ready for use** ページで、**View Operator** をクリックします。
  - **Project** が **openshift-storage** であることを確認します。デフォルトで、プロジェクトは **openshift-storage** です。
  - **Operators** → **Installed Operators** をクリックします。
  - OpenShift Container Storage の **Status** が **Succeeded** に変更するまで待機します。
- **View Installed Operators in namespace openshift-storage** をクリックします。
  - **Installed Operators** ページで、**ocs-operator** をクリックします。
  - **Subscription Details** ページで、**Install Plan** リンクをクリックします。
  - **InstallPlan Details** ページで、**Preview Install Plan** をクリックします。
  - インストール計画を確認し、**Approve** をクリックします。
  - **Components** の **Status** が **Unknown** から **Created** または **Present** のいずれかに変更するまで待機します。
  - **Operators** → **Installed Operators** をクリックします。
  - **Project** が **openshift-storage** であることを確認します。デフォルトで、プロジェクトは **openshift-storage** です。
  - OpenShift Container Storage の **Status** が **Succeeded** に変更するまで待機します。

### 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。
- **View Installed Operators in namespace openshift-storage** リンクをクリックし、OpenShift Container Storage Operator が **Installed Operators** ダッシュボードで **Status** を **Succeeded** として表示していることを確認します。

## 1.2. 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。

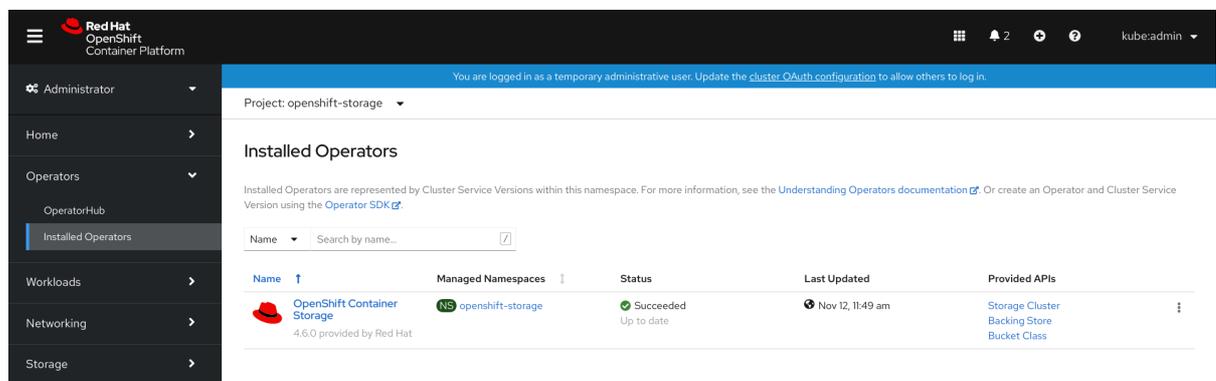
### 前提条件

- 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、[Installing OpenShift Container Storage Operator using the Operator Hub](#) を参照してください。

### 手順

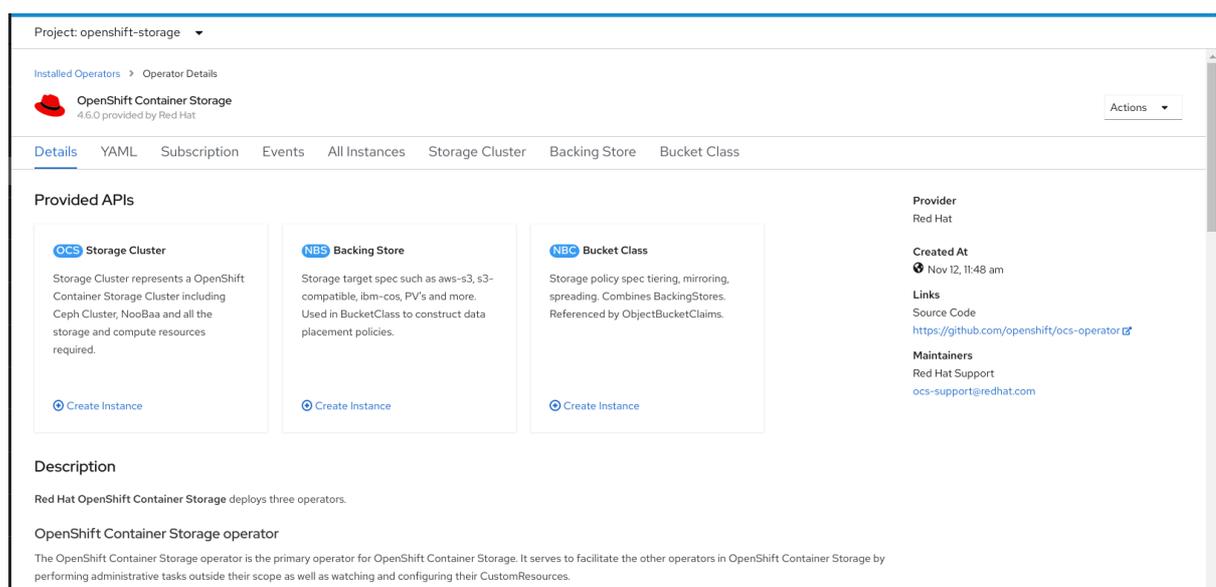
1. **Operators** → **Installed Operators** をクリックし、インストールされた Operator をすべて表示します。  
選択された **Project** が **openshift-storage** であることを確認します。

図1.1 OpenShift Container Storage Operator ページ



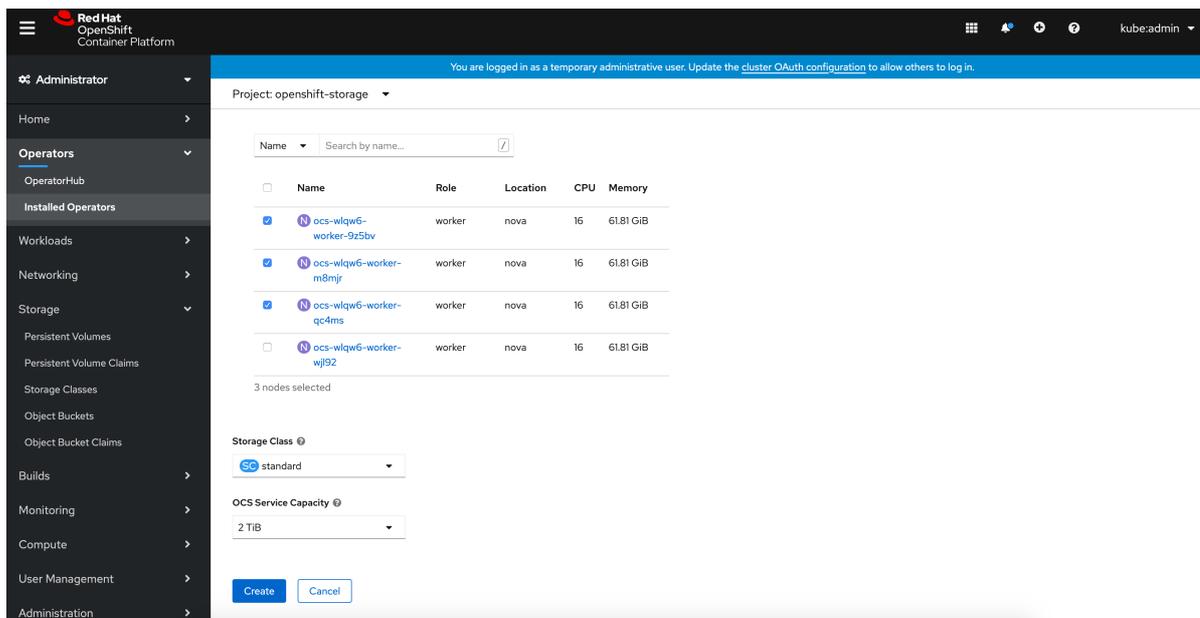
2. **OpenShift Container Storage** をクリックします。

図1.2 OpenShift Container Storage の Details タブ



3. **Storage Cluster** の **Create Instance** リンクをクリックします。

## 図1.3 Create Storage Cluster ページ



4. Create Storage Cluster ページで、以下のオプションが選択されていることを確認します。

- Select Mode セクションで、**Internal** モードがデフォルトで選択されます。
- Storage Class は、デフォルトで **standard** に設定されます。
- ドロップダウンリストから **OpenShift Container Storage Service Capacity** を選択します。



### 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)。

- (任意) Encryption セクションで、トグルを **Enabled** に設定して、クラスターでデータ暗号化を有効にします。
- Nodes セクションでは、OpenShift Container Storage サービスを使用するために、利用可能な一覧から 3 つ以上のワーカー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、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。



### 注記

クラスターで特定のワーカーノードを見つけるには、Name または Label に基づいてノードをフィルターできます。

- Name では、ノード名で検索できます。
- Label では、事前に定義されたラベルを選択して検索できます。

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。

5. **Create** をクリックします。  
**Create** ボタンは、3つのノードを選択した後にのみ有効になります。選択したノードごとに、3つのストレージデバイスを持つ新しいストレージクラスターが作成されます。デフォルト設定では、レプリケーション係数3を使用します。

## 検証手順

1. 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 の **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。
  - OpenShift Container Storage 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。

## 1.3. 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。

### 1.3.1. 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1.1「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1.1 OpenShift Container Storage クラスターに対応する Pod

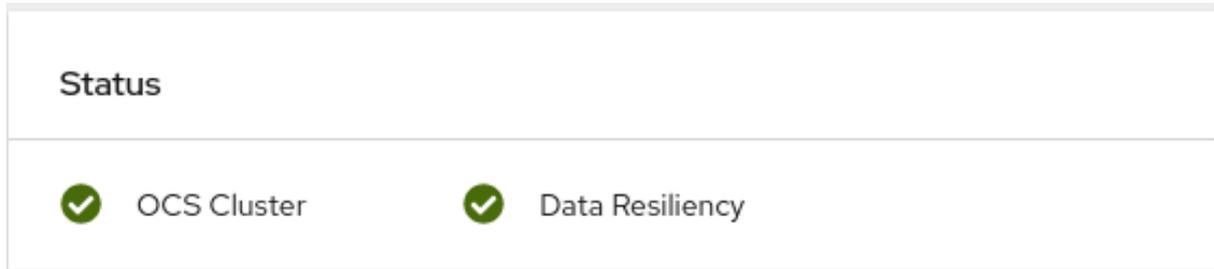
コンポーネント	対応する Pod
OpenShift Container Storage Operator	<ul style="list-style-type: none"> <li>● <b>ocs-operator-*</b> (任意のワーカーノードに 1 Pod)</li> <li>● <b>ocs-metrics-exporter-*</b></li> </ul>
Rook-ceph Operator	<b>rook-ceph-operator-*</b>  (任意のワーカーノードに 1 Pod)

コンポーネント	対応する Pod
Multicloud Object Gateway	<ul style="list-style-type: none"> <li>● <b>noobaa-operator-*</b> (任意のワーカーノードに 1 Pod)</li> <li>● <b>noobaa-core-*</b> (任意のストレージノードに 1 Pod)</li> <li>● <b>noobaa-db-*</b> (任意のストレージノードに 1 Pod)</li> <li>● <b>noobaa-endpoint-*</b> (任意のストレージノードに 1 Pod)</li> </ul>
MON	<p><b>rook-ceph-mon-*</b></p> <p>(ストレージノードに分散する 3 Pod)</p>
MGR	<p><b>rook-ceph-mgr-*</b></p> <p>(任意のストレージノードに 1 Pod)</p>
MDS	<p><b>rook-ceph-mds-ocs-storagecluster-cephfilesystem-*</b></p> <p>(ストレージノードに分散する 2 Pod)</p>
CSI	<ul style="list-style-type: none"> <li>● <b>cephfs</b> <ul style="list-style-type: none"> <li>○ <b>csi-cephfsplugin-*</b> (各ワーカーノードに 1 Pod)</li> <li>○ <b>csi-cephfsplugin-provisioner-*</b> (ワーカーノードに分散する 2 Pod)</li> </ul> </li> <li>● <b>rbd</b> <ul style="list-style-type: none"> <li>○ <b>csi-rbdplugin-*</b> (各ワーカーノードに 1 Pod)</li> <li>○ <b>csi-rbdplugin-provisioner-*</b> (ストレージノードに分散する 2 Pod)</li> </ul> </li> </ul>
rook-ceph-crashcollector	<p><b>rook-ceph-crashcollector-*</b></p> <p>(各ストレージノードに 1 Pod)</p>
OSD	<ul style="list-style-type: none"> <li>● <b>rook-ceph-osd-*</b> (各デバイス用に 1 Pod)</li> <li>● <b>rook-ceph-osd-prepare-ocs-deviceset-*</b> (各デバイス用に 1 Pod)</li> </ul>

### 1.3.2. OpenShift 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1.4 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

OpenStack

#### モード

内部

#### バージョン

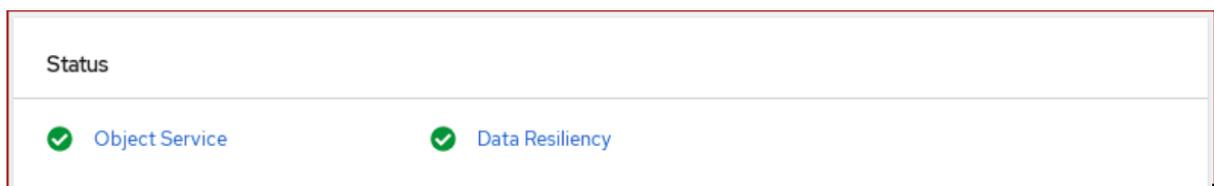
ocs-operator-4.6.0

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。

### 1.3.3. 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1.5 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
<b>システム名</b>
Multicloud Object Gateway
<b>プロバイダー</b>
OpenStack
<b>バージョン</b>
ocs-operator-4.6.0

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。

### 1.3.4. OpenShift 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**

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

### 1.4.1. 内部モードでの OpenShift 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1.2 **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
------------	---	-------	----

Annotation	値	デフォルト	動作
cleanup-policy	delete	はい	Rook は物理ドライブおよび <b>DataDirHostPath</b> をクリーンアップします。
cleanup-policy	Retain	いいえ	Rook は物理ドライブおよび <b>DataDirHostPath</b> をクリーンアップし <b>ません</b> 。
mode	graceful	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。
mode	forced	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ンインストールを 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-  
cluster-cleanup-job-  
cluster-cleanup-job-
```

5. `/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
```

6. 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。

- a. `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
```

- b. 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)
```

- c. 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
```

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 `cryptsetup` プロセスの PID を検索します。

```
$ ps
```

出力例:

```
PID  TTY  TIME  CMD
778825  ?    00:00:00 cryptsetup
```

**PID** 番号を書き留めて強制終了します。この例では、**PID** は **778825** です。

- `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。

- 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-
```

- 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-
```

- 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>
```

- Multicloud Object Gateway storageclass を削除します。

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

- 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 cephrbdmirrors.ceph.rook.io --wait=true --timeout=5m
```

- 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。

#### 1.4.1.1. OpenShift 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、モニターリングスタックを OpenShift Container Storage からクリーンアップします。

モニター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 モニターリングスタックを使用できるように設定されません。  
詳細は、[モニターリングスタックの設定](#) を参照してください。

## 手順

1. **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
```

#### 1.4.1.2. OpenShift Container Storage からの OpenShift 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は、[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
```

#### 1.4.1.3. OpenShift 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2章 外部モードでの OPENSIFT CONTAINER STORAGE の RED HAT OPENSTACK PLATFORM へのデプロイ

Red Hat OpenShift Container Storage は、外部でホストされる Red Hat Ceph Storage (RHCS) クラスタを Red Hat OpenStack Platform のストレージプロバイダーとして使用できます。詳細は、[Planning your deployment](#) を参照してください。

RHCS 4 クラスタのインストール方法は、[インストールガイド](#) を参照してください。

以下の手順に従って、OpenShift Container Storage を外部モードでデプロイします。

1. [OpenShift Container Storage Operator](#) をインストールします。
2. [OpenShift Container Storage Cluster Service](#) を作成します。

### 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。ハードウェアおよびソフトウェアの要件に関する詳細は、[デプロイメントのプランニング](#) を参照してください。

#### 前提条件

- OpenShift Container Platform (RHOC) クラスタにログインしている必要があります。
- RHOC クラスタにワーカーノードが少なくとも3つ必要です。

#### 注記

- OpenShift Container Storage のクラスタ全体でのデフォルトノードセレクターを上書きする必要がある場合は、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 の空のノードセレクターを指定できます。

```
$ 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 の専用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. OpenShift Web コンソールの左側のペインで、**Operators** → **OperatorHub** をクリックします。
2. **Filter by keyword** テキストボックスまたはフィルター一覧を使用して、Operator の一覧から OpenShift Container Storage を検索します。
3. **OpenShift Container Storage** をクリックします。
4. **OpenShift Container Storage Operator** ページで、**Install** をクリックします。

5. **Install Operator** ページで、以下のオプションがデフォルトで選択されていることを確認します。
  - a. Channel を **stable-4.6**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. **Enable operator recommended cluster monitoring on this namespace**が選択されていることを確認します。この設定はクラスタのモニタリングに必要です。
  - e. **承認ストラテジー** を **Automatic** または **Manual** として選択している。承認ストラテジーはデフォルトで **Automatic** に設定されます。
    - **Approval Strategy** に **Automatic** を選択します。



#### 注記

Approval Strategy を **Automatic** として選択すると、新規インストール時、または OpenShift Container Storage の最新バージョンへの更新時に承認は必要ありません。

- i. **インストール** をクリックします。
  - ii. インストール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。
  - iii. **Operators** → **Installed Operators** をクリックします。
  - iv. **Project** が **openshift-storage** であることを確認します。デフォルトで、**プロジェクト** は **openshift-storage** です。
  - v. **OpenShift Container Storage** の **Status** が **Succeeded** に変更するまで待機します。
- **Approval Strategy** に **Manual** を選択します。



#### 注記

Approval Strategy を **Manual** として選択すると、新規インストール時、または OpenShift Container Storage の最新バージョンへの更新時に承認が必要になります。

- i. **インストール** をクリックします。
- ii. **Manual approval required** ページで、**Approve** または **View Installed Operators in namespace openshift-storage** のいずれかをクリックし、Operator をインストールできます。

**重要**

オプションのいずれかをクリックする前に、**Manual approval required** ページで、インストール計画がウィンドウに読み込まれるまで数分待機します。

**重要**

**Approve** をクリックする場合は、続行する前にインストール計画を確認する必要があります。

- **Approve** をクリックします。
  - OpenShift Container Storage Operator がインストールされている間に数分待機します。
  - **Installed operator - ready for use** ページで、**View Operator** をクリックします。
  - **Project** が **openshift-storage** であることを確認します。デフォルトで、プロジェクトは **openshift-storage** です。
  - **Operators** → **Installed Operators** をクリックします。
  - OpenShift Container Storage の **Status** が **Succeeded** に変更するまで待機します。
- **View Installed Operators in namespace openshift-storage** をクリックします。
  - **Installed Operators** ページで、**ocs-operator** をクリックします。
  - **Subscription Details** ページで、**Install Plan** リンクをクリックします。
  - **InstallPlan Details** ページで、**Preview Install Plan** をクリックします。
  - インストール計画を確認し、**Approve** をクリックします。
  - **Components** の **Status** が **Unknown** から **Created** または **Present** のいずれかに変更するまで待機します。
  - **Operators** → **Installed Operators** をクリックします。
  - **Project** が **openshift-storage** であることを確認します。デフォルトで、プロジェクトは **openshift-storage** です。
  - OpenShift Container Storage の **Status** が **Succeeded** に変更するまで待機します。

**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。
- **View Installed Operators in namespace openshift-storage** リンクをクリックし、OpenShift Container Storage Operator が **Installed Operators** ダッシュボードで **Status** を **Succeeded** として表示していることを確認します。

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

OpenShift Container Storage Operator を Red Hat OpenStack プラットフォームの OpenShift Container Platform にインストールした後に、OpenShift Container Storage クラスターサービスを新規に作成する必要があります。

### 前提条件

- 作業用の OpenShift Container Platform バージョン 4.5.4 以降にログインしている必要があります。
- OpenShift Container Storage Operator はインストールされている必要があります。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。

- 外部クラスターには、Red Hat Ceph Storage バージョン 4.2z1 以降が必要です。詳細は、この [Red Hat Ceph Storage リリースおよび対応する Ceph パッケージバージョンについてのナレッジベースのアーティクル](#) を参照してください。

Red Hat Ceph Storage クラスターを 4.1.1 以前のバージョンから最新リリースに更新し、これが新規にデプロイされたクラスターではない場合は、Red Hat Ceph Storage クラスターで CephFS プールのアプリケーションタイプを手動で設定し、外部モードで CephFS PVC の作成を有効にする必要があります。

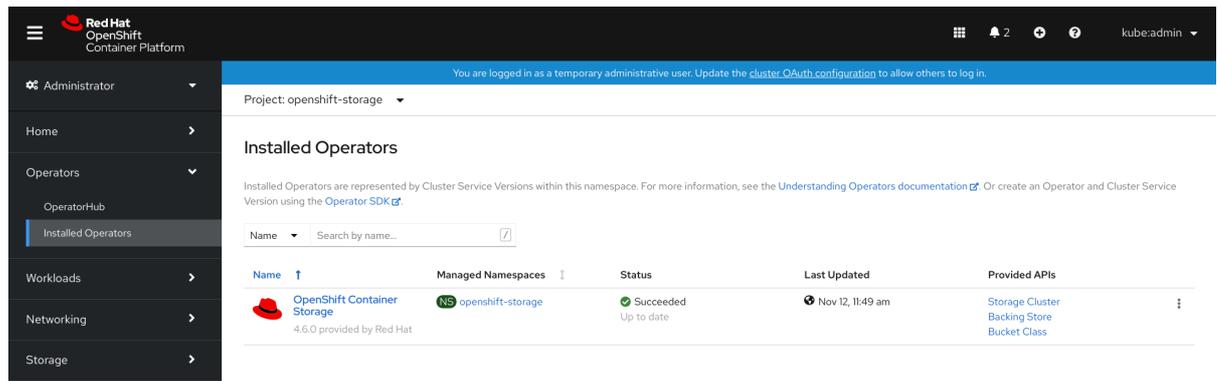
詳細は、[外部モードでの CephFS PVC の作成のトラブルシューティング](#) について参照してください。

- Red Hat Ceph Storage には Ceph Dashboard がインストールされ、設定されている必要があります。Ceph Manager Prometheus エクスポーターにポート **9283** を使用する必要があります。詳細は、[Ceph Dashboard のインストールおよびアクセス](#) について参照してください。
- 外部 RedHat CephStorage クラスターに対して PG Autoscaler オプションを有効にすることが推奨されます。詳細は、Red Hat Ceph Storage ドキュメントの [The placement group autoscaler](#) セクションを参照してください。
- 外部 Ceph クラスターには、既存の RBD プールを使用できるように事前に設定されている必要があります。これがない場合は、OpenShift Container Storage のデプロイメントに進む前に、Red Hat Ceph Storage の管理者に問い合わせてこれを作成してください。OpenShift Container Storage クラスターごとに別個のプールを使用することが推奨されます。

### 手順

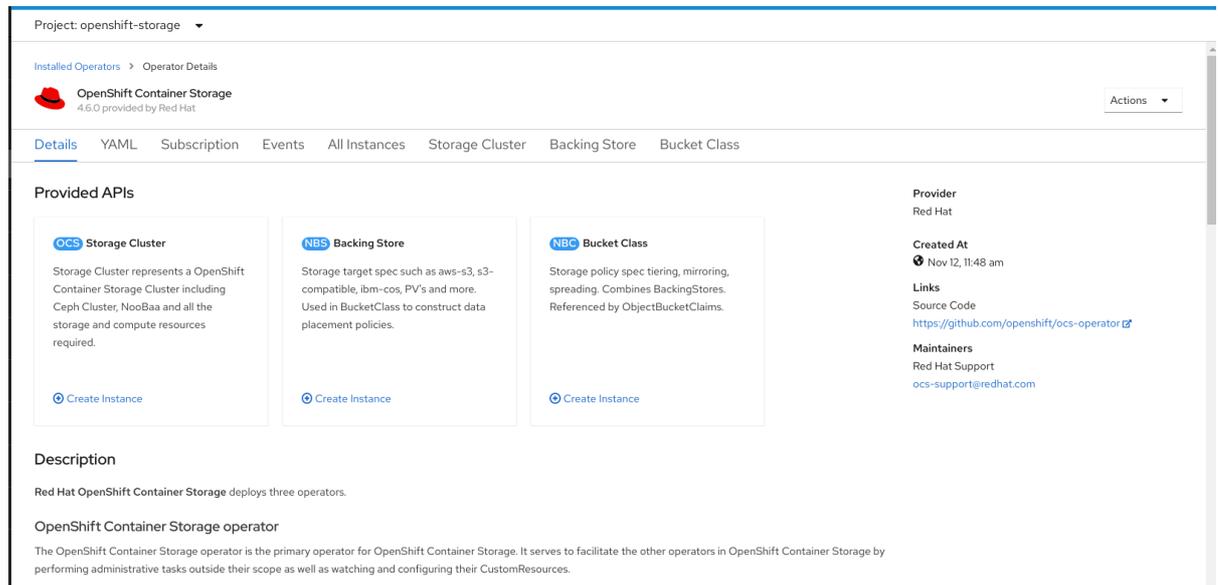
1. **Operators → Installed Operators** をクリックし、インストールされた Operator をすべて表示します。  
選択された **Project** が **openshift-storage** であることを確認します。

## 図2.1 OpenShift Container Storage Operator ページ



2. OpenShift Container Storage をクリックします。

## 図2.2 OpenShift Container Storage の Details タブ



3. Storage Cluster の **Create Instance** リンクをクリックします。
4. Mode を **External** に選択します。デフォルトでは、Internal はデプロイメントモードとして選択されます。

図2.3 Create Storage Cluster 形式の外部クラスターへの接続

Project: openshift-storage ▾

OpenShift Container Storage > Create Storage Cluster

## Create Storage Cluster

OCS runs as a cloud-native service for optimal integration with applications in need of storage, and handles the scenes such as provisioning and management.

Select Mode:  Internal  Internal - Attached Devices  External

### Connect to external cluster

Download [ceph-external-cluster-details-exporter.py](#) script and run on the RHCS cluster, then upload the results(JSON) in the External cluster metadata field. [Download Script](#)

**i** A bucket will be created to provide the OCS Service.

External cluster metadata \*

Upload Credentials file

5. Connect to external cluster セクションで、**Download Script** リンクをクリックして、Ceph クラスターの詳細を抽出するために python スクリプトをダウンロードします。
6. Red Hat Ceph Storage (RHCS) クラスターの詳細を抽出するには、RHCS 管理者に問い合わせた上で Red Hat Ceph Storage でダウンロードした python スクリプトを **admin key** を使用して実行します。
  - a. RHCS ノードで以下のコマンドを実行し、利用可能な引数の一覧を表示します。

```
# python3 ceph-external-cluster-details-exporter.py --help
```



### 重要

Red Hat Ceph Storage 4.x クラスターが Red Hat Enterprise Linux 7.x (RHEL 7.x) クラスターにデプロイされている場合は、**python3** ではなく **python** を使用します。



### 注記

MON コンテナ内 (コンテナ化されたデプロイメント) または MON ノード (rpm デプロイメント) からスクリプトを実行することもできます。

- b. RHCS クラスターから外部クラスターの詳細を取得するには、以下のコマンドを実行します。

```
# python3 ceph-external-cluster-details-exporter.py \
--rbd-data-pool-name <rbd block pool name> [optional arguments]
```

以下に例を示します。

```
# python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name ceph-rbd --
monitoring-endpoint xxx.xxx.xxx.xxx --monitoring-endpoint-port 9283 --rgw-endpoint
xxx.xxx.xxx.xxx:xxxx --run-as-user client.ocs
```

上記の例は、以下のようになります。

- **--rbd-data-pool-name** は、OpenShift Container Storage でブロックストレージを提供するために使用される必須のパラメーターです。
- **--rgw-endpoint** は任意です。OpenShift Container Storage の Ceph Rados Gateway でオブジェクトストレージをプロビジョニングする場合に、このパラメーターを指定します。<ip\_address>:<port> の形式でエンドポイントを指定します。
- **--monitoring-endpoint** は任意です。これは、OpenShift Container Platform クラスターから到達できるアクティブな **ceph-mgr** の IP アドレスです。指定しない場合には、値が自動的に入力されます。
- **--monitoring-endpoint-port** は任意です。これは **--monitoring-endpoint** で指定された **ceph-mgr** Prometheus エクスポートに関連付けられるポートです。指定しない場合には、値が自動的に入力されます。ポート **9283** のみが OpenShift Container Platform 4.6 でサポートされます。
- **--run-as-user** は、スクリプトで作成される Ceph ユーザーの名前を指定するために使用されるオプションのパラメーターです。このパラメーターを指定しないと、デフォルトのユーザー名 **client.healthchecker** が作成されます。新規ユーザーのパーミッションは以下のように設定されます。
  - caps: [mgr] はコマンド設定を許可します。
  - caps: [mon] は r を許可し、コマンド quorum\_status を許可し、コマンド version を許可します。
  - caps: [osd] allow rwx pool=**RGW\_POOL\_PREFIX.rgw.meta**, allow r pool=**.rgw.root**, allow rw pool=**RGW\_POOL\_PREFIX.rgw.control**, allow rx pool=**RGW\_POOL\_PREFIX.rgw.log**, allow x pool=**RGW\_POOL\_PREFIX.rgw.buckets.index**

python スクリプトを使用して生成された JSON 出力の例:

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

```

```
<admin-key>}}, {"name": "rook-csi-cephfs-node", "kind": "Secret", "data":
{"adminID": "csi-cephfs-node", "adminKey": "<admin-key>"}}, {"name": "cephfs",
"kind": "StorageClass", "data": {"fsName": "cephfs", "pool": "cephfs_data"}},
{"name": "ceph-rgw", "kind": "StorageClass", "data": {"endpoint":
"xxx.xxx.xxx.xxx:xxxx", "poolPrefix": "default"}}
```

- c. JSON 出力を **.json** 拡張のあるファイルに保存します。



### 注記

OpenShift Container Storage がシームレスに機能するには、JSON ファイルを使用してアップロードされるパラメーター (RGW エンドポイント、CephFS の詳細、RBD プールなど) が、ストレージクラスターの作成後も RHCS 外部クラスターで変更されないままであることを確認します。

7. **External cluster metadata** → **Browse** をクリックして、JSON ファイルを選択し、アップロードします。

JSON ファイルの内容が入力され、テキストボックスに表示されます。

図2.4 JSON ファイルの内容

External cluster metadata \*

Upload Credentials file Browse

```
ROOK CEPH MON ENDPOINTS = {"maxMonId": "0", "data": {"ceph-mon-node=xxx.xxx.xxx:xxxx", "mapping": {}}}
ROOK CEPH MON = {"mon-secret": "mon-secret", "fsid": "<fs-id>", "admin-secret": "admin-secret"}
ROOK CEPH OPERATOR CREDS = {"userKey": "<user-key>", "userID": "client.healthchecker"}
ROOK CSI RBD NODE = {"userKey": "<user-key>", "userID": "csi-rbd-node"}
CEPH RBD = {"pool": "ceph-rbd"}
MONITORING ENDPOINT = {"MonitoringPort": "xxxx", "MonitoringEndpoint": "xxx.xxx.xxx.xxx"}
ROOK CSI RBD PROVISIONER = {"userKey": "<user-key>", "userID": "csi-rbd-provisioner"}
ROOK CSI CEPHFS PROVISIONER = {"adminID": "csi-cephfs-provisioner", "adminKey": "<admin-key>"}
ROOK CSI CEPHFS NODE = {"adminID": "csi-cephfs-node", "adminKey": "<admin-key>"}
CEPHFS = {"pool": "cephfs_data", "fsName": "cephfs"}
CEPH RGW = {"endpoint": "xxx.xxx.xxx:xxxx", "poolPrefix": "default"}
```

8. **Create** をクリックします。

Create ボタンは、**.json** ファイルのアップロード後にのみ有効になります。

### 検証手順

- インストールされたストレージクラスターの最後の **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。
- OpenShift Container Storage、Pod および StorageClass が正常にインストールされていることを確認するには、[外部モードの OpenShift Container Storage インストールの確認](#) を参照してください。

## 2.3. 外部モードの OPENSIFT CONTAINER STORAGE インストールの確認

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。

### 2.3.1. Pod の状態の確認

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の詳細は、[表2.1「OpenShift Container Storage コンポーネントに対応する Pod」](#) を参照してください。
3. 以下の Pod が実行中であることを確認します。

表2.1 OpenShift Container Storage コンポーネントに対応する Pod

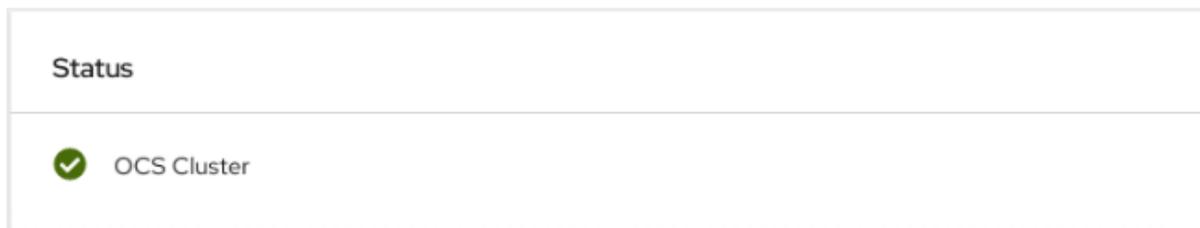
コンポーネント	対応する Pod
OpenShift Container Storage Operator	<ul style="list-style-type: none"> <li>● <b>ocs-operator-*</b> (任意のワーカーノードに 1Pod)</li> <li>● <b>ocs-metrics-exporter-*</b></li> </ul>
Rook-ceph Operator	<p><b>rook-ceph-operator-*</b></p> <p>(任意のワーカーノードに 1Pod)</p>
Multicloud Object Gateway	<ul style="list-style-type: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

コンポーネント	対応する Pod
CSI	<ul style="list-style-type: none"> <li>● <b>cephfs</b> <ul style="list-style-type: none"> <li>○ <b>csi-cephfsplugin-*</b> (各ワーカーノードに 1 Pod)</li> <li>○ <b>csi-cephfsplugin-provisioner-*</b> (ワーカーノードに分散する 2 Pod)</li> </ul> </li> </ul> <div style="display: flex; align-items: center; margin: 10px 0;">  <div> <p><b>注記</b></p> <p>MDS が外部クラスターにデプロイされていない場合、csi-cephfsplugin Pod は作成されません。</p> </div> </div> <ul style="list-style-type: none"> <li>● <b>rbd</b> <ul style="list-style-type: none"> <li>○ <b>csi-rbdplugin-*</b> (各ワーカーノードに 1 Pod)</li> <li>○ <b>csi-rbdplugin-provisioner-*</b> (ストレージノードに分散する 2 Pod)</li> </ul> </li> </ul>

### 2.3.2. OpenShift 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** 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2.5 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-external-storagecluster

**プロバイダー**

OpenStack

**モード**

外部

**バージョン**

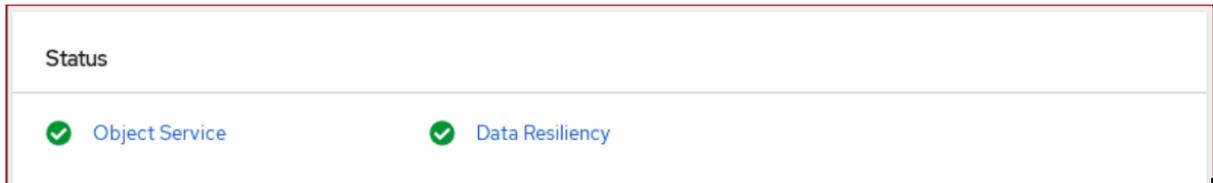
ocs-operator-4.6.0

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。

### 2.3.3. 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2.6 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適切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  
RADOS Object Gateway

#### プロバイダー

OpenStack

#### バージョン

ocs-operator-4.6.0



#### 注記

RADOS Object Gateway は、OpenShift Container Storage を外部モードでデプロイし、RADOS Object Gateway エンドポイントの詳細が含まれている場合にのみ表示されます。

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。

### 2.3.4. ストレージクラスが作成され、一覧表示されることの確認

- 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-external-storagecluster-ceph-rbd**
  - **ocs-external-storagecluster-ceph-rgw**

- **ocs-external-storagecluster-cephfs**
- **openshift-storage.noobaa.io**



#### 注記

- MDS が外部クラスターにデプロイされていない場合、**ocs-external-storagecluster-cephfs** ストレージクラスは作成されません。
- RGW が外部クラスターにデプロイされていない場合、**ocs-external-storagecluster-ceph-rgw** ストレージクラスは作成されません。

MDS および RGW についての詳細は、[Red Hat Ceph Storage のドキュメント](#) を参照してください。

### 2.3.5. Ceph クラスターが接続されていることの確認

以下のコマンドを実行して、OpenShift Container Storage クラスターが外部の Red Hat Ceph Storage クラスターに接続されているかどうかを確認します。

```
$ oc get cephcluster -n openshift-storage
```

NAME	DATADIRHOSTPATH	MONCOUNT	AGE	PHASE
ocs-external-storagecluster-cephcluster		31m15s	Connected	Cluster
connected successfully	HEALTH_OK			

### 2.3.6. ストレージクラスターの準備が整っていることを確認します。

以下のコマンドを実行して、ストレージクラスターが準備状態にあり、**External** オプションが **true** に設定されていることを確認します。

```
$ oc get storagecluster -n openshift-storage
```

NAME	AGE	PHASE	EXTERNAL	CREATED AT	VERSION
ocs-external-storagecluster	31m15s	Ready	true	2020-07-29T20:43:04Z	4.6.0

## 2.4. 外部モードでの OPENSIFT CONTAINER STORAGE のアンインストール

### 2.4.1. 外部モードでの OpenShift 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。OpenShift Container Storage をアンインストールしても、外部クラスターから RBD プールが削除されたり、外部の Red Hat Ceph Storage クラスターがアンインストールされたりしません。

#### 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`



### 注記

`uninstall.ocs.openshift.io/cleanup-policy` は外部モードには適用できません。

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2.2 `uninstall.ocs.openshift.io` uninstall annotations descriptions

アノテーション	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 <b>DataDirHostPath</b> 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 <b>DataDirHostPath</b> をクリーンアップし <b>ません</b> 。
<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 <b>一時停止</b> 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 <b>続行</b> 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、アンインストールモードを変更できます。

```
$ oc annotate storagecluster ocs-external-storagecluster uninstall.ocs.openshift.io/mode="forced" --
  overwrite
storagecluster.ocs.openshift.io/ocs-external-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。

## 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ンインストールを 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 により内部で使用される PVC および OBC 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
```

- 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>
```

クラスターに作成されているカスタムバックングストア、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
```

- 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。

5. 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>
```

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

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

7. **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 cephrbdmirrors.ceph.rook.io --wait=true --timeout=5m
```

8. 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。
  - a. **Home** → **Overview** をクリックし、ダッシュボードにアクセスします。
  - b. **Persistent Storage** および **Object Service** タブが **Cluster** タブの横に表示されないことを確認します。

## 2.4.2. OpenShift 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、モニターリングスタックを OpenShift Container Storage からクリーンアップします。

モニター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 モニターリングスタックを使用できるように設定されません。  
詳細は、[モニターリングスタックの設定](#) を参照してください。

### 手順

1. **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                                STATUS  VOLUME
CAPACITY ACCESS MODES STORAGECLASS  AGE
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-0 Bound  pvc-0d519c4f-
15a5-11ea-baa0-026d231574aa 40Gi  RWO          ocs-external-storagecluster-ceph-rbd
8d
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-1 Bound  pvc-
0d5a9825-15a5-11ea-baa0-026d231574aa 40Gi  RWO          ocs-external-
storagecluster-ceph-rbd 8d
persistentvolumeclaim/my-alertmanager-claim-alertmanager-main-2 Bound  pvc-
0d6413dc-15a5-11ea-baa0-026d231574aa 40Gi  RWO          ocs-external-
storagecluster-ceph-rbd 8d
persistentvolumeclaim/my-prometheus-claim-prometheus-k8s-0   Bound  pvc-0b7c19b0-
15a5-11ea-baa0-026d231574aa 40Gi  RWO          ocs-external-storagecluster-ceph-rbd
8d
persistentvolumeclaim/my-prometheus-claim-prometheus-k8s-1   Bound  pvc-0b8aed3f-
15a5-11ea-baa0-026d231574aa 40Gi  RWO          ocs-external-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-external-storagecluster-ceph-rbd
  prometheusK8s:
    volumeClaimTemplate:
      metadata:
        name: my-prometheus-claim
      spec:
        resources:
          requests:
            storage: 40Gi
        storageClassName: ocs-external-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. PVC を使用する Pod を一覧表示します。

この例では、PVC を使用していた **alertmanagerMain** および **prometheusK8s** Pod は **Terminating** 状態にあります。これらの Pod が OpenShift Container Storage PVC を使用しなくなった後に PVC を削除できます。

```

$ oc get pod,pvc -n openshift-monitoring

```

NAME	READY	STATUS	RESTARTS	AGE
pod/alertmanager-main-0	3/3	Terminating	0	10h
pod/alertmanager-main-1	3/3	Terminating	0	10h
pod/alertmanager-main-2	3/3	Terminating	0	10h
pod/cluster-monitoring-operator-84cd9df668-zhjfn	1/1	Running	0	18h
pod/grafana-5db6fd97f8-pmtbf	2/2	Running	0	10h
pod/kube-state-metrics-895899678-z2r9q	3/3	Running	0	10h
pod/node-exporter-4njxv	2/2	Running	0	18h
pod/node-exporter-b8ckz	2/2	Running	0	11h
pod/node-exporter-c2vp5	2/2	Running	0	18h
pod/node-exporter-cq65n	2/2	Running	0	18h
pod/node-exporter-f5sm7	2/2	Running	0	11h
pod/node-exporter-f852c	2/2	Running	0	18h
pod/node-exporter-l9zn7	2/2	Running	0	11h
pod/node-exporter-ngbs8	2/2	Running	0	18h
pod/node-exporter-rv4v9	2/2	Running	0	18h
pod/openshift-state-metrics-77d5f699d8-69q5x	3/3	Running	0	10h
pod/prometheus-adapter-765465b56-4tbxx	1/1	Running	0	10h
pod/prometheus-adapter-765465b56-s2qq2	1/1	Running	0	10h
pod/prometheus-k8s-0	6/6	Terminating	1	9m47s
pod/prometheus-k8s-1	6/6	Terminating	1	9m47s
pod/prometheus-operator-cbfd89f9-ldnwc	1/1	Running	0	43m
pod/telemeter-client-7b5ddb4489-2xfpz	3/3	Running	0	10h

NAME	STATUS	VOLUME	
CAPACITY	ACCESS MODES	STORAGECLASS	AGE

```

persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-0 Bound pvc-
2eb79797-1fed-11ea-93e1-0a88476a6a64 40Gi RWO ocs-external-
storagecluster-ceph-rbd 19h
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-1 Bound pvc-
2eb79797-1fed-11ea-93e1-0a88476a6a64 40Gi RWO ocs-external-
storagecluster-ceph-rbd 19h
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-2 Bound pvc-2ec6a9cf-
1fed-11ea-93e1-0a88476a6a64 40Gi RWO ocs-external-storagecluster-ceph-rbd
19h
persistentvolumeclaim/ocs-prometheus-claim-prometheus-k8s-0 Bound pvc-3162a80c-
1fed-11ea-93e1-0a88476a6a64 40Gi RWO ocs-external-storagecluster-ceph-rbd
19h
persistentvolumeclaim/ocs-prometheus-claim-prometheus-k8s-1 Bound pvc-
316e99e2-1fed-11ea-93e1-0a88476a6a64 40Gi RWO ocs-external-
storagecluster-ceph-rbd 19h

```

5. 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
```

### 2.4.3. OpenShift Container Storage からの OpenShift 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は、[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:
  emptyDir: {}
.
.
.

```

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
```

#### 2.4.4. OpenShift 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 を使用するように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3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。

### 3.1. 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。

## 第4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されるインフラストラクチャーによって異なります。



### 警告

これらのサービスに十分なストレージ容量があることを常に確認してください。これらの重要なサービスのストレージ領域が不足すると、クラスターは動作しなくなり、復元が非常に困難になります。

Red Hat は、これらのサービスのキュ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。

### 4.1. 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。



### 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**)。

## 4.2. 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 AlertManager 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 × トリクスデータの保持期間の編集](#) 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。
- モニターリング 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** をクリックしてクラスター 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** 状態が表示されることを確認します。
- Workloads** → **Pods**に移動します。
  - 新規の **alertmanager-main-\*** Pod をクリックし、Podの詳細を表示します。
  - 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)

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

3. 新規 **prometheus-k8s-\*** Pod が **Running** 状態が表示されることを確認します。
- 新規 **prometheus-k8s-\*** Pod をクリックし、Podの詳細を表示します。
  - Volumes** までスクロールダウンし、ボリュームに新規の Persistent Volume Claim (永続ボリューム要求、PVC) のいずれかに一致する **Type ocs-prometheus-claim** があることを確認します (例: **ocs-prometheus-claim-prometheus-k8s-0**)。

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

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	ocs-prometheus-claim-prometheus-k8s-0	Read/Write	prometheus

## 4.3. 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、[クラスターロギングのデプロイ](#) を参照してください。

OpenShift Container Platform の初回のデプロイメントでは、OpenShift 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 ドキュメントで [クラスターロギング Curator](#) について参照してください。

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

### 4.3.1. 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、[クラスターロギングの設定](#) を参照してください。

### 4.3.2. 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せん。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) の状態が表示されることを確認します。

図4.1 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-9r624biv-1	openshift-logging	Bound	pvc-8993013d-1a6e-11ea-8d2f-027b4eaf61a	200G
elasticsearch-elasticsearch-cdm-9r624biv-2	openshift-logging	Bound	pvc-89947c90-1a6e-11ea-8d2f-027b4eaf61a	200G
elasticsearch-elasticsearch-cdm-9r624biv-3	openshift-logging	Bound	pvc-8995f557-1a6e-11ea-8d2f-027b4eaf61a	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、PersistenVolumeClaim Overview ページでストレージクラス名を確認します。

## 注記

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

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

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

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

## 注記

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5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6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ルすると、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 (RHOCP) ノードには OpenShift Container Storage サブスクリプションが必要ですが、RHOCP 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。

### 6.1. 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 RHOCP エンタイトルメントを使用しないようにするには、**infra** ノードロールのラベルが必要です。**infra** ノードロールラベルは、OpenShift 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ル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

RHOCP サブスクリプションコストが適用されないように、ラベルは RHOCP ノードを **infra** ノードとして識別します。テイントは、OpenShift 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ル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: ""
```

### 6.2. 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ーノードをプロビジョニングするように設定されます。現時点で、OpenShift 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ーノードに必要な適切なテイントおよびラベルを持つノードを作成します。これは OpenShift 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: ""
```

## 6.3. インフラストラクチャーノードの手動作成

マシン API が環境内でサポートされない場合にのみ、ラベルはノードに直接適用される必要があります。手動作成では、OpenShift 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7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。

### 7.1. 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。

### 7.2. RED HAT OPENSTACK PLATFORM インフラストラクチャー上の OPENSIFT CONTAINER STORAGE ノードへの容量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。

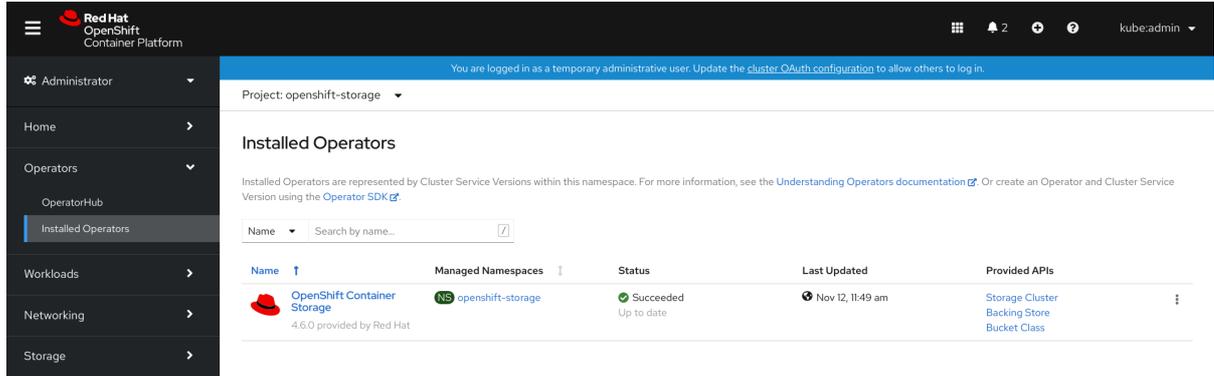
#### 前提条件

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

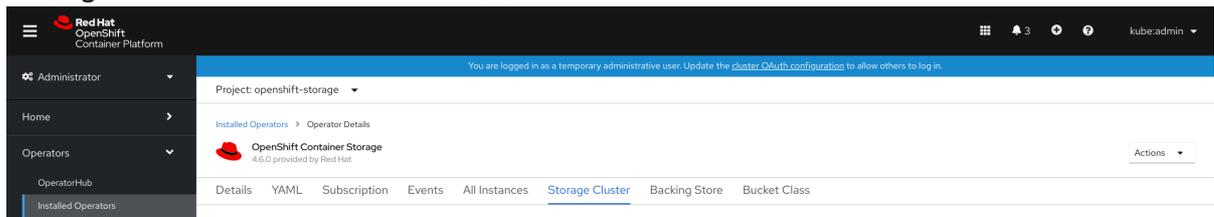
- デプロイメント時にプロビジョニングされたストレージクラス以外のストレージクラスを使用してスケールするには、最初に追加のストレージクラスを定義します。詳細は、[ストレージクラスの作成](#)を参照してください。

## 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。

# Add Capacity

Adding capacity for **ocs-storagecluster**, may increase your expenses.

**Storage Class** ?

SC standard

**Raw Capacity** ?

2 x 3 replicas = 6 TiB

Currently Used: Not available

Cancel Add

7. ストレージクラスを選択します。

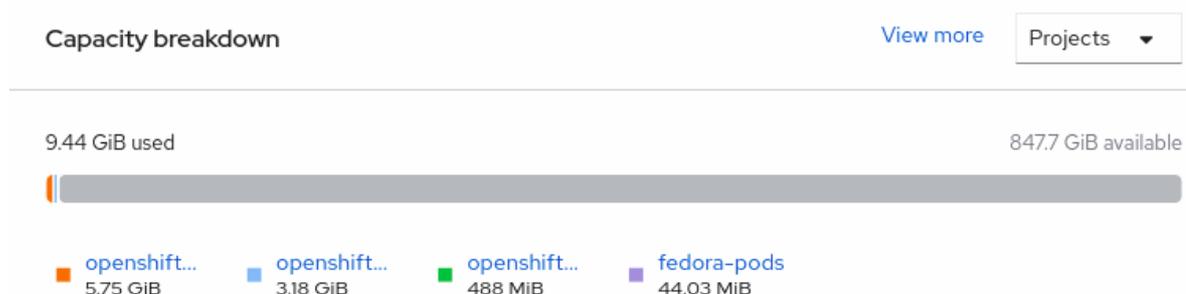
デプロイメント時に生成されるデフォルトのストレージクラスを使用している場合は、ストレージクラスを **standard** に設定します。他のストレージクラスを作成している場合は、適切なものを選択します。

**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、**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。

- 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。

- 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) クラスタでデータ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-qrgqm
```

- 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。

## 7.3. 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。

### 前提条件

- OpenShift Container Platform (RHOC) クラスタにログインしている必要があります。

## 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。

### 7.3.1. 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-\***

### 7.3.2. 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8章 MULTICLOUD OBJECT GATEWAY

### 8.1. 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。

### 8.2. 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8.1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。

### 8.2.1. 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".
```

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

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

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

## 2. Test S3 client:

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

**1** NOOBAA\_ACCESS\_KEY=\$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r '.data.AWS\_ACCESS\_KEY\_ID|@base64d')

**2** NOOBAA\_SECRET\_KEY=\$(kubectl get secret noobaa-admin -n openshift-storage -o json | jq -r '.data.AWS\_SECRET\_ACCESS\_KEY|@base64d')  
 alias s3='AWS\_ACCESS\_KEY\_ID=\$NOOBAA\_ACCESS\_KEY  
 AWS\_SECRET\_ACCESS\_KEY=\$NOOBAA\_SECRET\_KEY aws --endpoint https://localhost:10443 --no-verify-ssl s3'  
 s3 ls

### Services:

#### Service Mgmt:

##### External DNS:

```
https://noobaa-mgmt-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a3406079515be11eaa3b70683061451e-1194613580.us-east-
```

```
2.elb.amazonaws.com:443
```

##### Internal DNS:

```
https://noobaa-mgmt.openshift-storage.svc:443
```

##### Internal IP:

```
https://172.30.235.12:443
```

##### Node Ports:

```
https://10.0.142.103:31385
```

##### Pod Ports:

```
https://10.131.0.19:8443
```

#### serviceS3:

##### External DNS: **3**

```
https://s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
https://a340f4e1315be11eaa3b70683061451e-943168195.us-east-2.elb.amazonaws.com:443
```

##### Internal DNS:

```
https://s3.openshift-storage.svc:443
```

##### Internal IP:

```
https://172.30.86.41:443
```

##### Node Ports:

```
https://10.0.142.103:31011
```

##### Pod Ports:

```
https://10.131.0.19:6443
```

- 1** アクセスキー (AWS\_ACCESS\_KEY\_ID 値)
- 2** シークレットアクセスキー (AWS\_SECRET\_ACCESS\_KEY 値)
- 3** MCG エンドポイント



## 注記

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

## 8.2.2. 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/soulkSiA==

```

```
#-----#
```

```
#- 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8.2 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

```

## 8.3. ハイブリッドまたはマルチクラウド用のストレージリソースの追加

### 8.3.1. 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

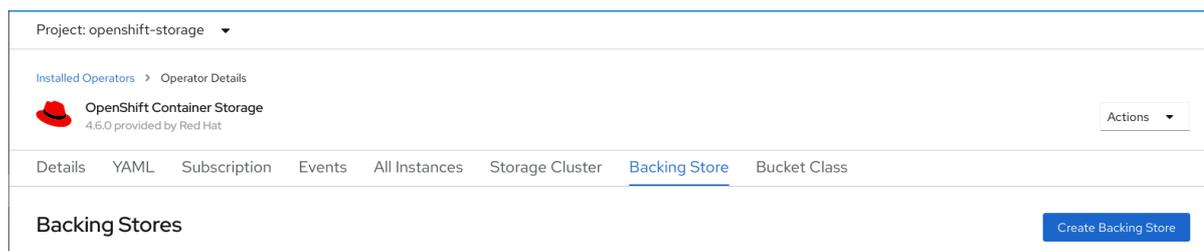
- OpenShift への管理者アクセス。

#### 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。

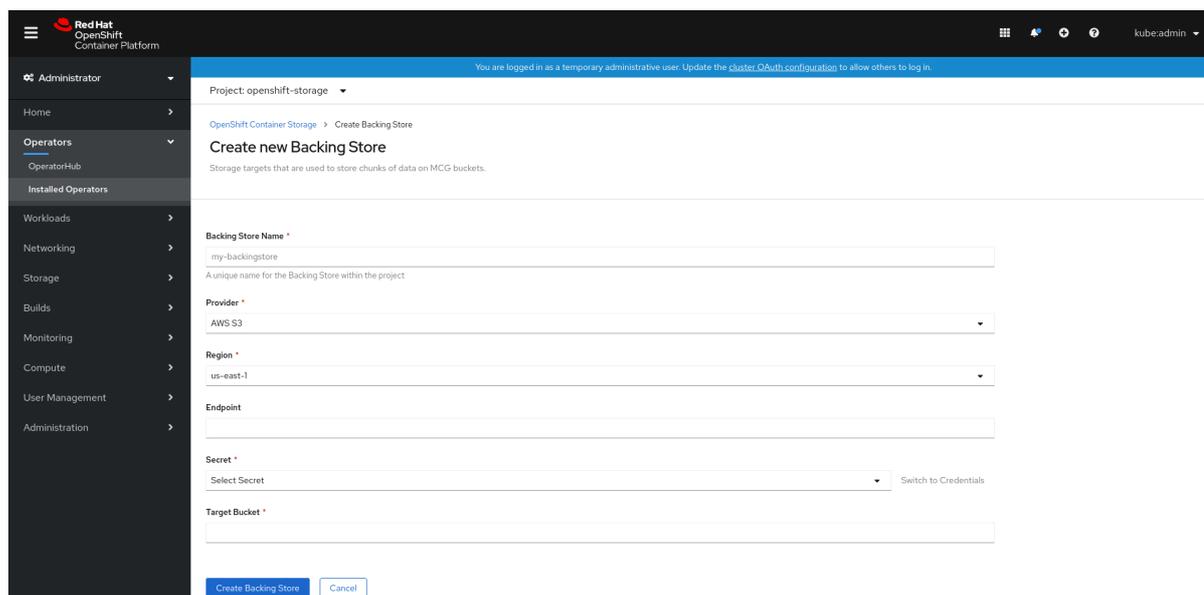
- OpenShift Container Storage Operator ページで右側にスクロールし、**Backing Store** タブをクリックします。

図8.1 バッキングストアタブのある OpenShift Container Storage Operator ページ



- Create Backing Store** をクリックします。

図8.2 Create Backing Store ページ



- Create New Backing Store ページで、以下を実行します。
  - Backing Store Name** を入力します。
  - Provider** を選択します。
  - Region** を選択します。
  - Endpoint** を入力します。これはオプションです。
  - 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。

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

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

MCG で使用できるバッ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。

#### 8.3.2.1. 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](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>
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: noobaa
      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。

### 8.3.2.2. 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](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>

```

- 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。

### 8.3.2.3. 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](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。

- 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

```

- <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。
- <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。

### 8.3.2.4. 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](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

- 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>

```

- <backingstore\_name>** を、バックングストアの名前に置き換えます。
- <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。
- <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。

### 8.3.2.5. 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

#### 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](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。
- 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。

- 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。

### 8.3.3. 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
```

- 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

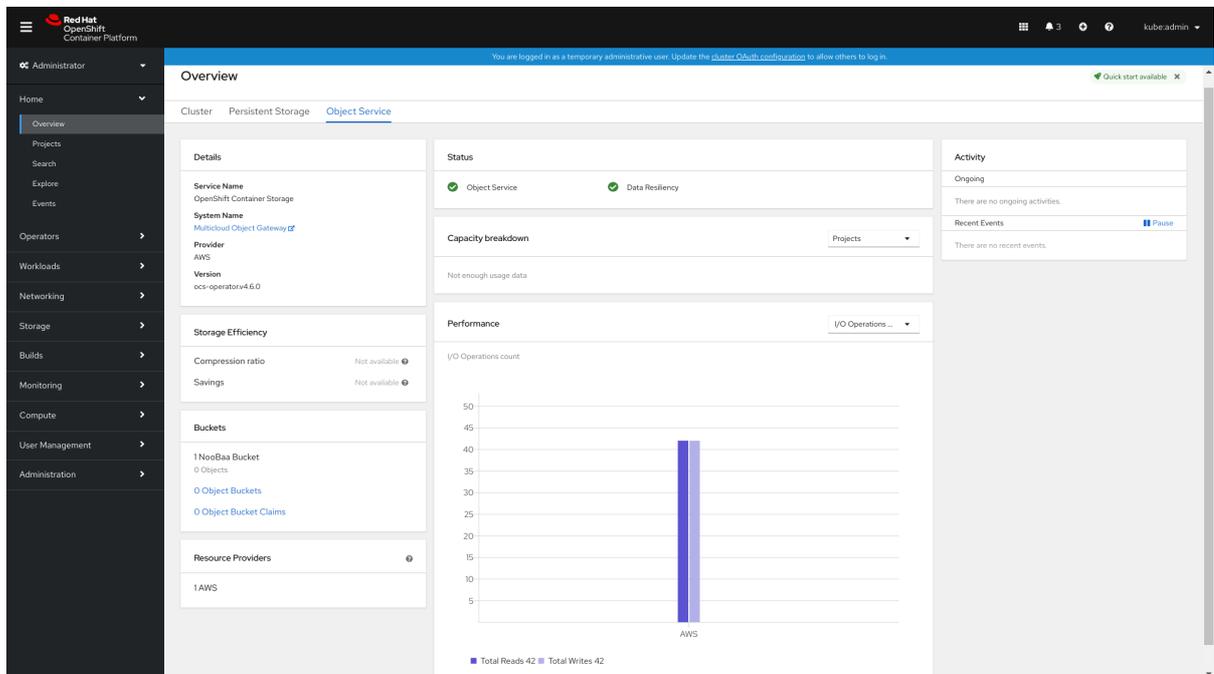
```

- 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。

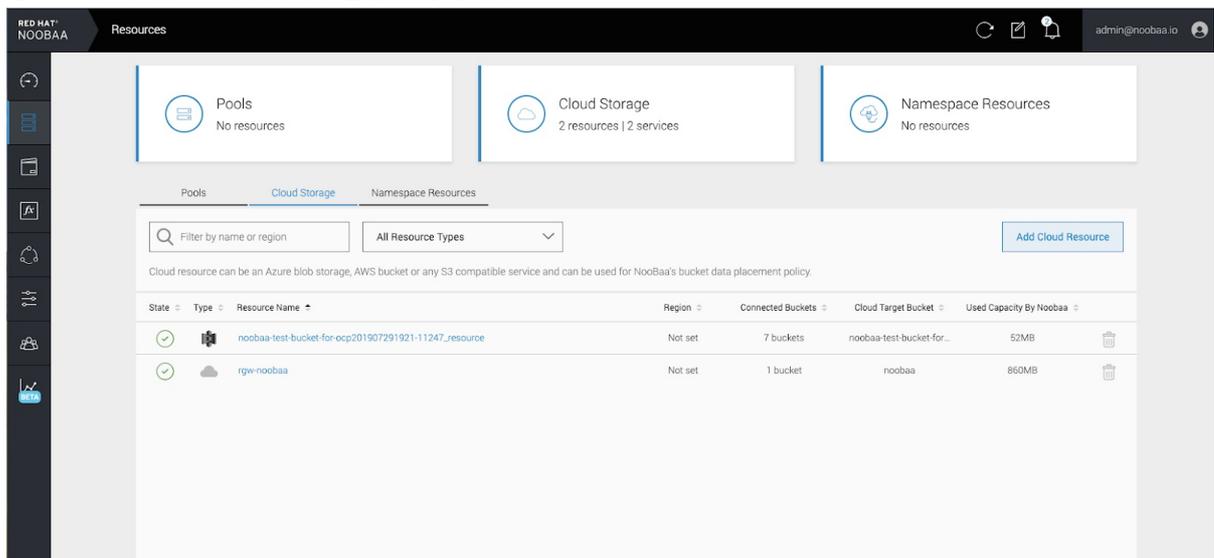
### 8.3.4. ユーザーインターフェイスを使用したハイブリッドおよびマルチクラウドのストレージリソースの追加

#### 手順

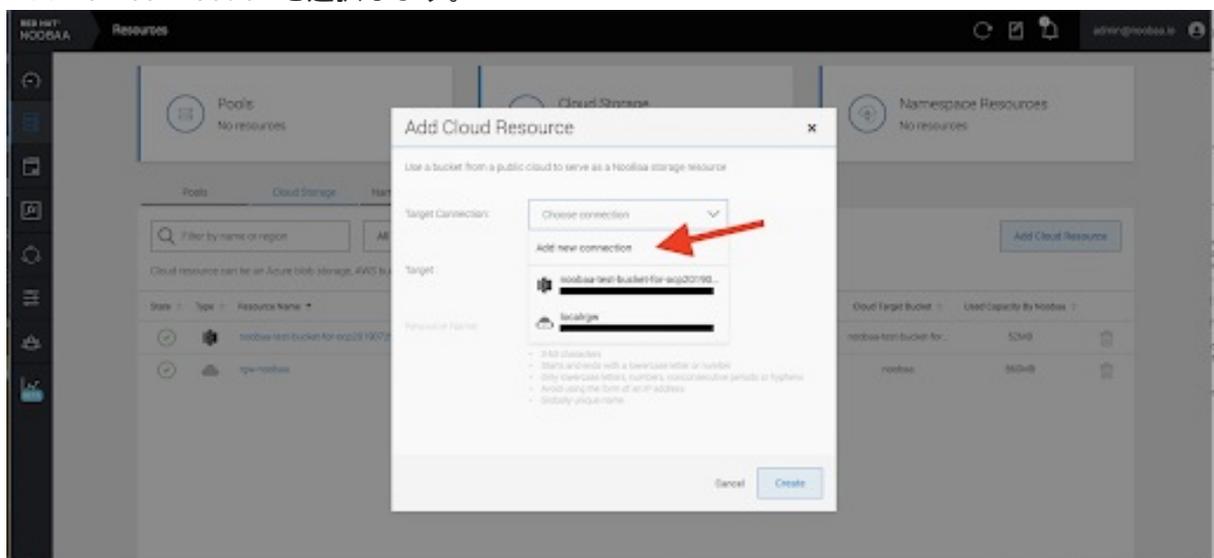
1. OpenShift Storage コンソールで、**Overview** → **Object Service** → に移動し、**Multicloud Object Gateway** リンクを選択します。



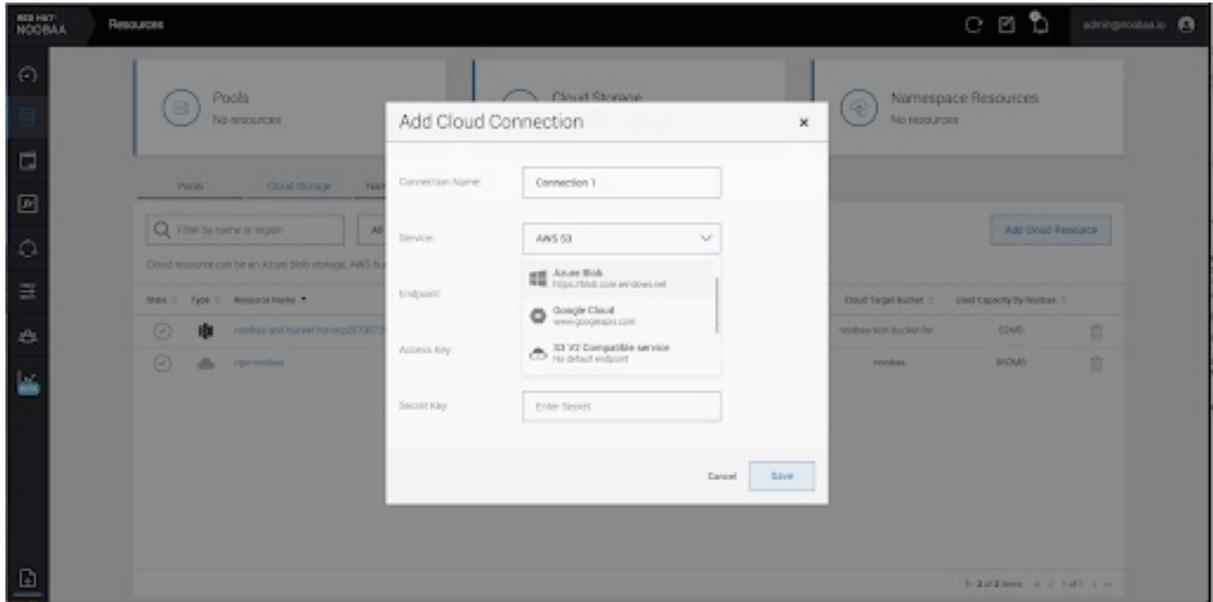
2. 以下に強調表示されているように左側にある **Resources** タブを選択します。設定する一覧から、**Add Cloud Resource** を選択します。



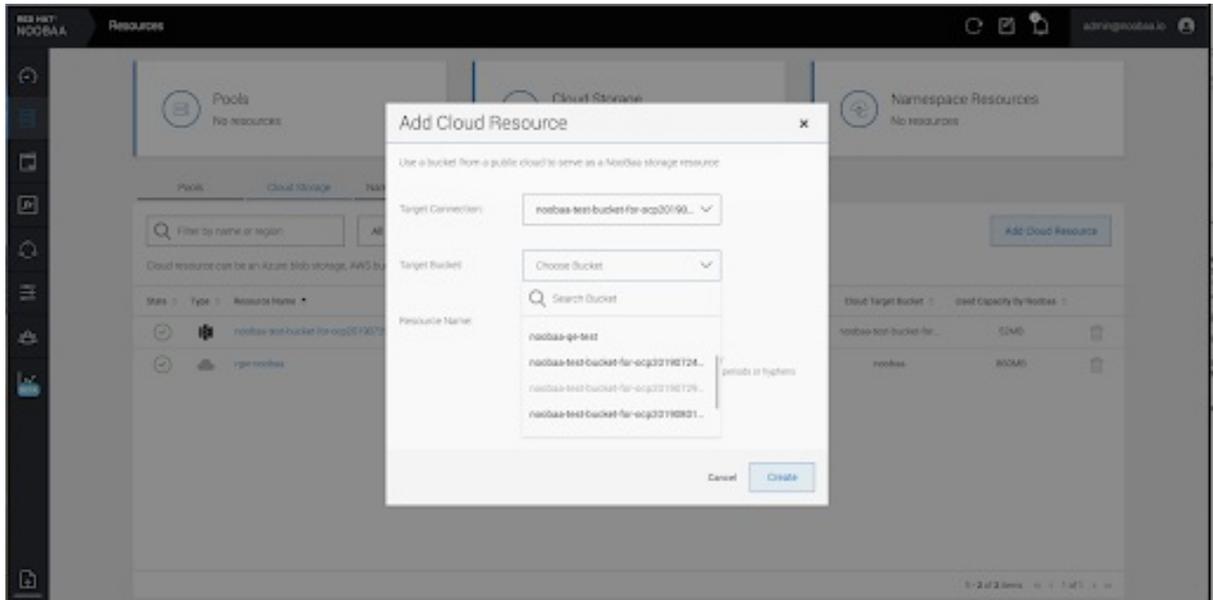
3. **Add new connection** を選択します。



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。

## 8.3.5. 新規バケットクラスの作成

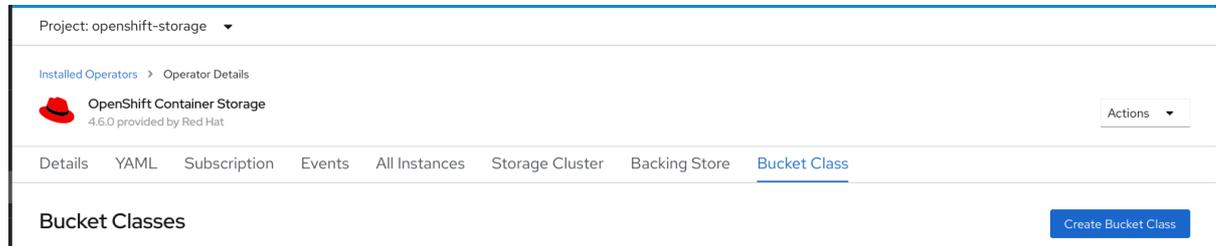
バケットクラスは、OBC (Object Bucket Class) の階層ポリシーおよびデータ配置を定義するバケットのクラスを表す CRD です。

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

### 手順

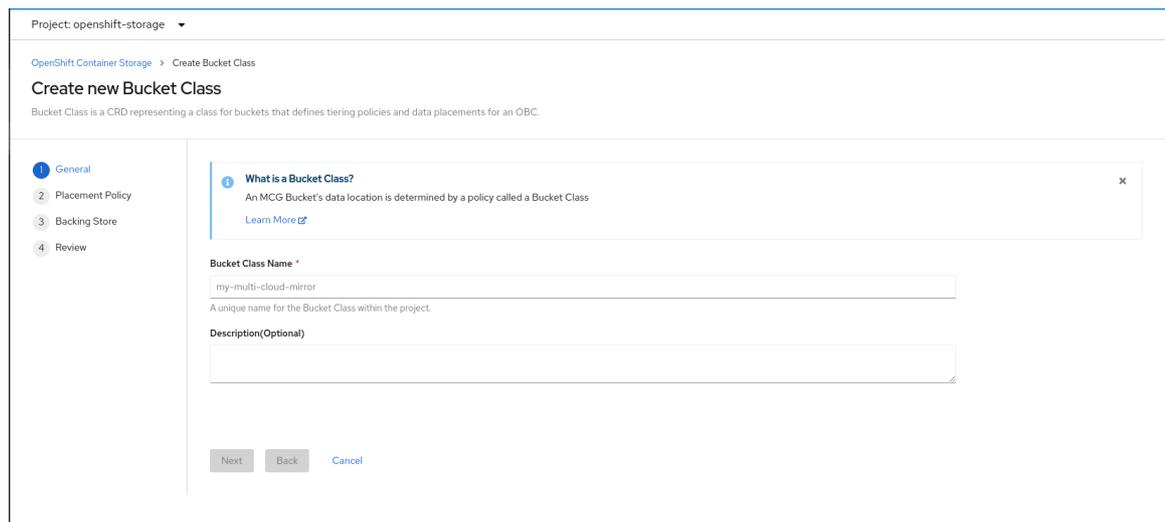
1. OpenShift Web コンソールの左側のペインで **Operators** → **Installed Operators** をクリックし、インストールされた Operator を表示します。
2. **OpenShift Container Storage Operator** をクリックします。
3. OpenShift Container Storage Operator ページで右側にスクロールし、**Bucket Class** タブをクリックします。

図8.3 Bucket Class タブのある OpenShift Container Storage Operator ページ



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。

図8.4 Create Bucket Class ページ



- 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。

図8.5 階層 1 - Policy Type 選択ページ

Project: openshift-storage

OpenShift Container Storage > Create Bucket Class

### Create new Bucket Class

Bucket Class is a CRD representing a class for buckets that defines tiering policies and data placements for an OBC.

- General
- Placement Policy
- Backing Store
- Review

#### Tier 1 - Policy Type

- Spread  
Spreading the data across the chosen resources. By default, a replica of one copy is used and does not include failure tolerance in case of resource failure.
- Mirror  
Full duplication of the data in each chosen resource. By default, a replica of one copy per location is used. Includes failure tolerance in case of resource failure.

[Add Tier](#)

[Next](#) [Back](#) [Cancel](#)

- c. Tier 1 - Policy Type で Spread を選択した場合、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。または、[新規バックキングストアを作成](#) することもできます。

図8.6 階層 1 - Backing Store 選択ページ

Project: openshift-storage

OpenShift Container Storage > Create Bucket Class

### Create new Bucket Class

Bucket Class is a CRD representing a class for buckets that defines tiering policies and data placements for an OBC.

- General
- Placement Policy
- Backing Store
- Review

#### Tier 1 - Backing Store (Spread)

[Create Backing Store](#)

Select at least 1 Backing Store resource \*

Search Backing Store

Name	BucketName	Type	Region
<input checked="" type="checkbox"/> <b>NBS</b> noobaa-default-backing-store	nb.1589272586147.apps.ebondare-dc25.q...	awsS3	us-east-2

1 resources selected

[Next](#) [Back](#) [Cancel](#)



## 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。

- a. Bucket Class 設定を確認し、確認します。

図8.7 バケットクラス設定の確認ページ

Project: openshift-storage

OpenShift Container Storage > Create Bucket Class

### Create new Bucket Class

Bucket Class is a CRD representing a class for buckets that defines tiering policies and data placements for an OBC.

- General
- Placement Policy
- Backing Store
- Review

#### Review and confirm Bucket Class settings

Bucket Class name  
ocs-01-spread

Placement Policy Details  
Tier 1: Spread  
Selected Backing Store: noobaa-default-backing-store

[Create Bucket Class](#) [Back](#) [Cancel](#)

- b. **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。

## 8.4. NAMESPACE バケットの設定

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 S3 API エンドポイント](#) について参照してください。

### 8.4.1. 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。

## 8.4.2. 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。

## 8.4.3. 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。

### 8.4.4. 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 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](#)

## 8.5. ハイブリッドおよびマルチクラウドバケットのデータのミラーリング

Multicloud Object Gateway (MCG) 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](#)」

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

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 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
```

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

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

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  name: hybrid-class
  labels:
    app: noobaa
spec:
  placementPolicy:
    tiers:
      - tier:
          mirrors:
            - mirror:
                spread:
                  - cos-east-us
            - mirror:
                spread:
                  - noobaa-test-bucket-for-ocp201907291921-11247_resource
```

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。

### 8.5.3. 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** リンクを選択します。

The screenshot shows the 'Overview' page for the 'Object Service' in the Red Hat OpenShift Container Platform Administrator console. The page is divided into several sections:

- Details:** Service Name: OpenShift Container Storage; System Name: Multicloud Object Gateway; Provider: AWS; Version: ocs-operatorv4.6.0.
- Status:** Object Service and Data Resiliency are both in a 'Good' state.
- Capacity breakdown:** Shows 'Not enough usage data'.
- Performance:** A bar chart shows I/O Operations count for 'AWS' with a value of 42. The legend indicates 'Total Reads 42' and 'Total Writes 42'.
- Activity:** Shows 'Ongoing' and 'Recent Events' sections, both indicating no activity.

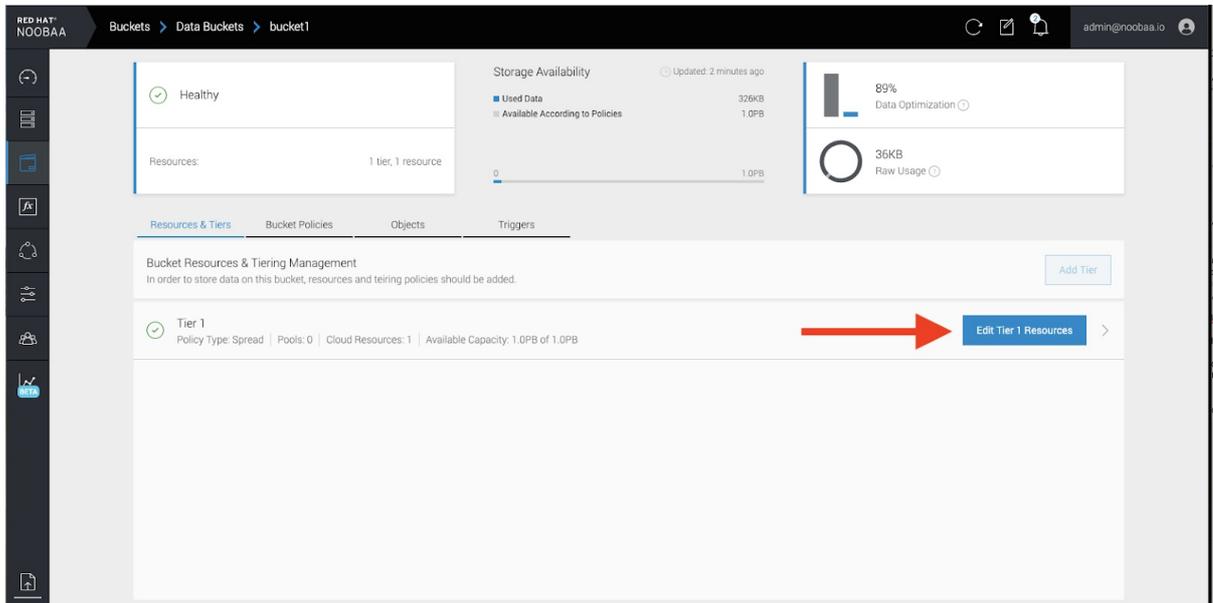
2. 左側の buckets アイコンをクリックします。バケットの一覧が表示されます。

The screenshot shows the 'Buckets' page in the Red Hat OpenShift Container Platform Administrator console. A red arrow points to the 'Buckets' icon in the left sidebar. The main content area displays a table of Data Buckets and Namespace Buckets.

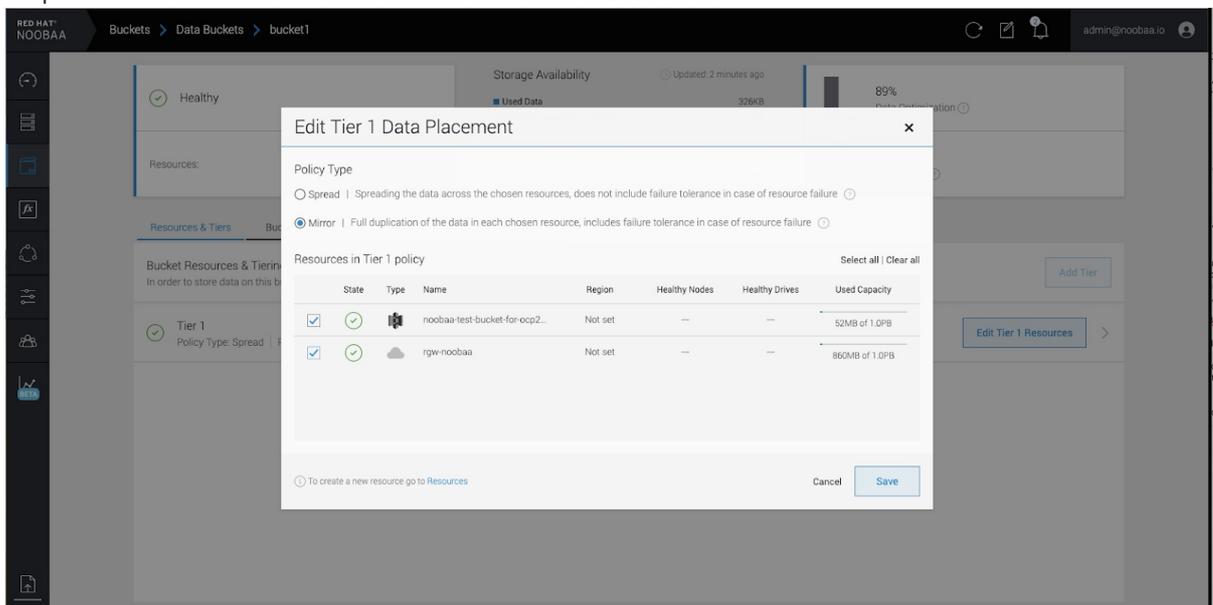
State	Bucket Name	Objects	Resiliency Policy	Tiers	Resources In Tiers	Versioning	Used Capacity
✓	bucket1	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket2	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket3	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket4	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
✓	bucket5	10	Replication (3 copies)	1 Tier		Disabled	36KB of 1.0PB
⚠	first.bucket	1	Replication (3 copies)	1 Tier		Disabled	3.5KB of 5.0GB
✓	localgw	589	Replication (3 copies)	1 Tier		Disabled	860MB of 1.0PB
✓	test	3	Replication (3 copies)	1 Tier		Disabled	43MB of 1.0PB
✓	velero	93	Replication (3 copies)	1 Tier		Disabled	13MB of 1.0PB

3. 更新するバケットをクリックします。

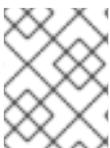
4. Edit Tier 1 Resources をクリックします。



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。以下の例では、prem Ceph RGW と AWS 間でデータのミラーリングをします。



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。

## 8.6. 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。

### 8.6.1. 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。

## 8.6.2. 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"
      ]
    }
  ]
}
```

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。それらの設定要素および使用方法についての詳細は、[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せん。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。

## 8.6.3. 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。

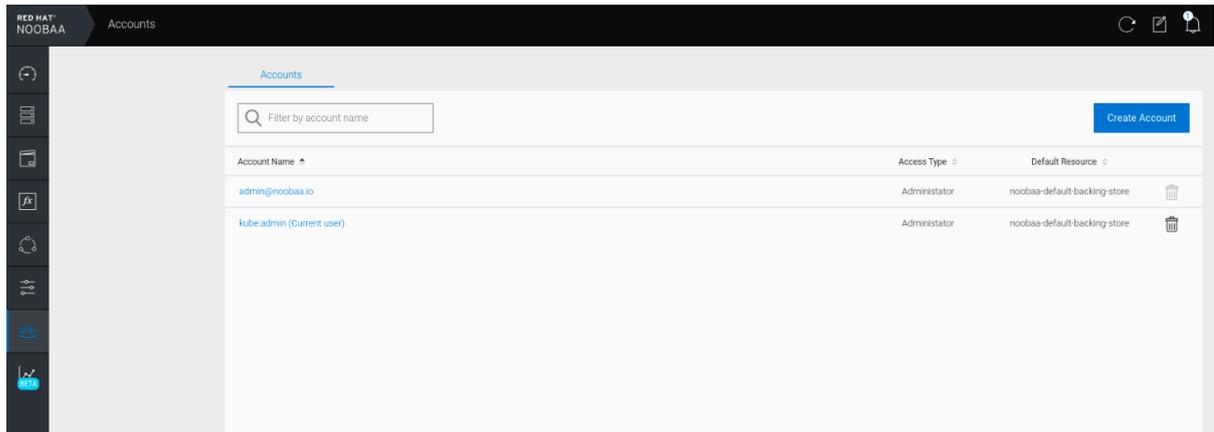
### 手順

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。

The screenshot shows the OpenShift Storage console interface. The main content area is titled 'Overview' and is divided into several sections:

- Details:** Shows Service Name (OpenShift Container Storage), System Name (Multicloud Object Gateway), Provider (AWS), and Version (ocs-operator@4.6.0).
- Status:** Displays two green checkmarks for 'Object Service' and 'Data Resiliency'.
- Capacity breakdown:** Shows 'Not enough usage data'.
- Storage Efficiency:** Shows 'Compression ratio' and 'Savings' as 'Not available'.
- Buckets:** Lists 1 NooBaa Bucket (0 Objects), 0 Object Buckets, and 0 Object Bucket Claims.
- Resource Providers:** Lists 1 AWS.
- Performance:** Includes a bar chart for 'I/O Operations count' showing 42 Total Reads and 42 Total Writes for AWS.
- Activity:** Shows 'Ongoing' and 'Recent Events' sections, both indicating no activity.

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:

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

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

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

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

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

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

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

永続ボリュームと同様に、Object Bucket Claim (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

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

コマンドラインインターフェイスを使用して Object Bucket Claim(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:
  Wi9kcFluSWxHRzIWaFlzNk1hc0xma2JXcjM1MVhqa051SIBleXpmOQ==
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。

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

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

#### 前提条件

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

#### 手順

1. OpenShift Web コンソールにログインします。
2. 左側のナビゲーションバーで **Storage** → **Object Bucket Claims** をクリックします。
3. **Create Object Bucket Claim** をクリックします。

Project: openshift-storage ▼

## Object Bucket Claims

[Create Object Bucket Claim](#)

No Object Bucket Claims Found

4. 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

Project: openshift-storage ▼

## Create Object Bucket Claim

[Edit YAML](#)

## Object Bucket Claim Name

If not provided, a generic name will be generated.

## Storage Class \*

Select storage class ▼

No default storage class

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

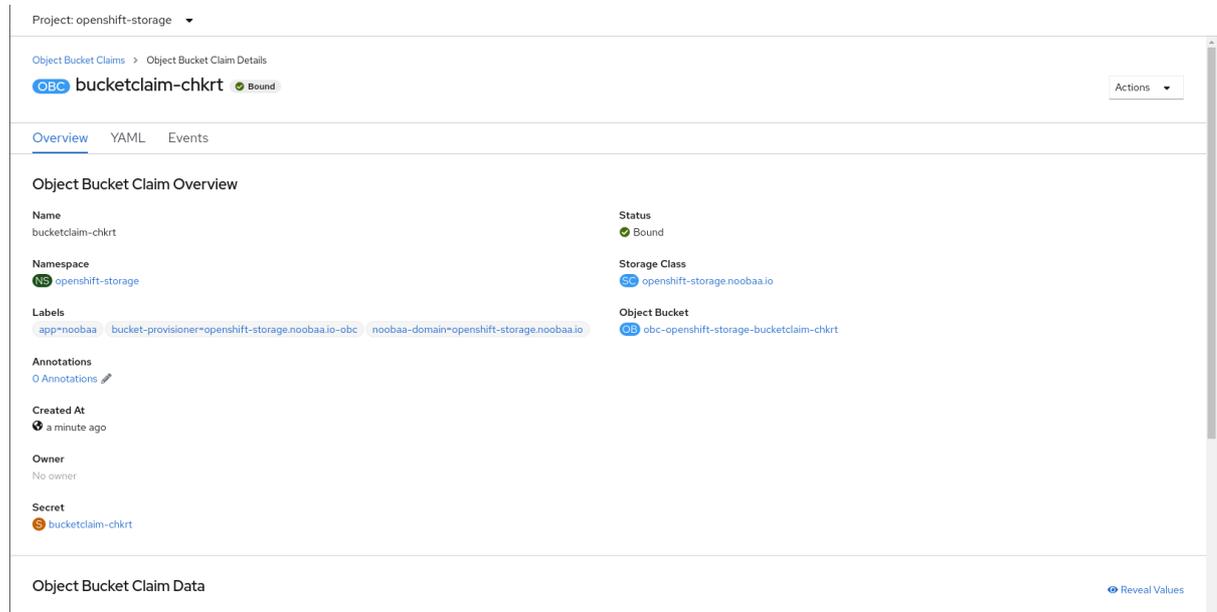
- **ocs-external-storagecluster-ceph-rgw** は Ceph Object Gateway (RGW) を使用します。
- **openshift-storage.noobaa.io** は Multicloud Object Gateway を使用します。



### 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)」

## 8.8. 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 エンドポイントの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

### 8.8.1. 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。

### 8.8.2. ストレージノードを使用したスケーリング

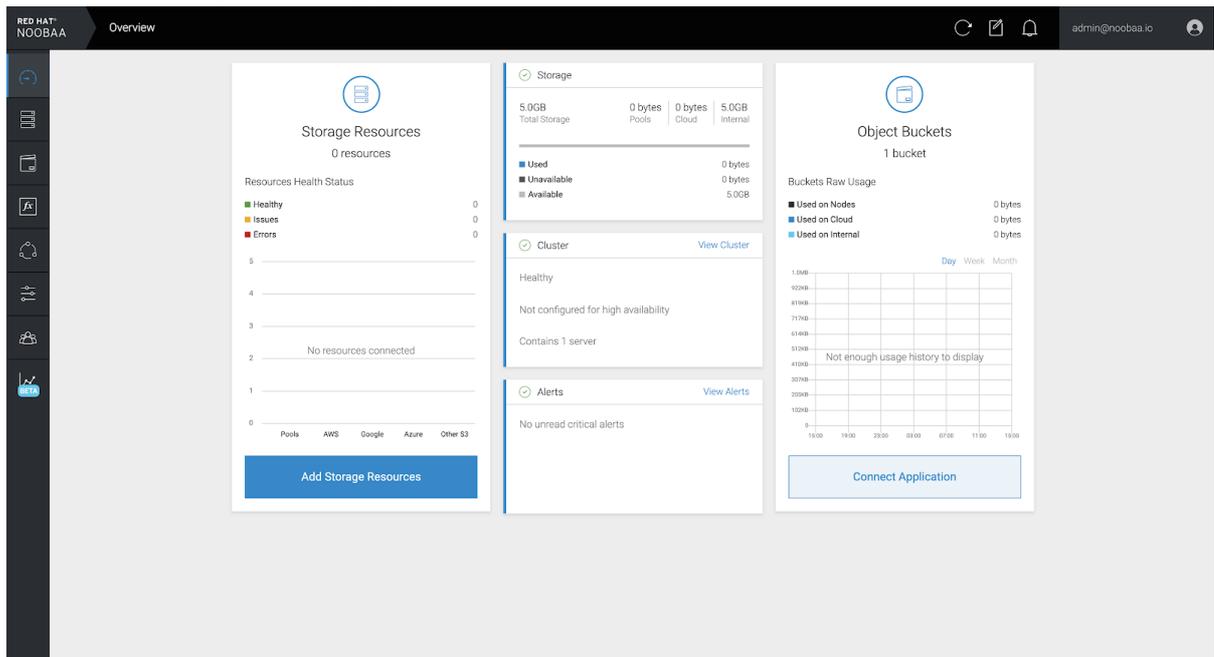
#### 前提条件

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

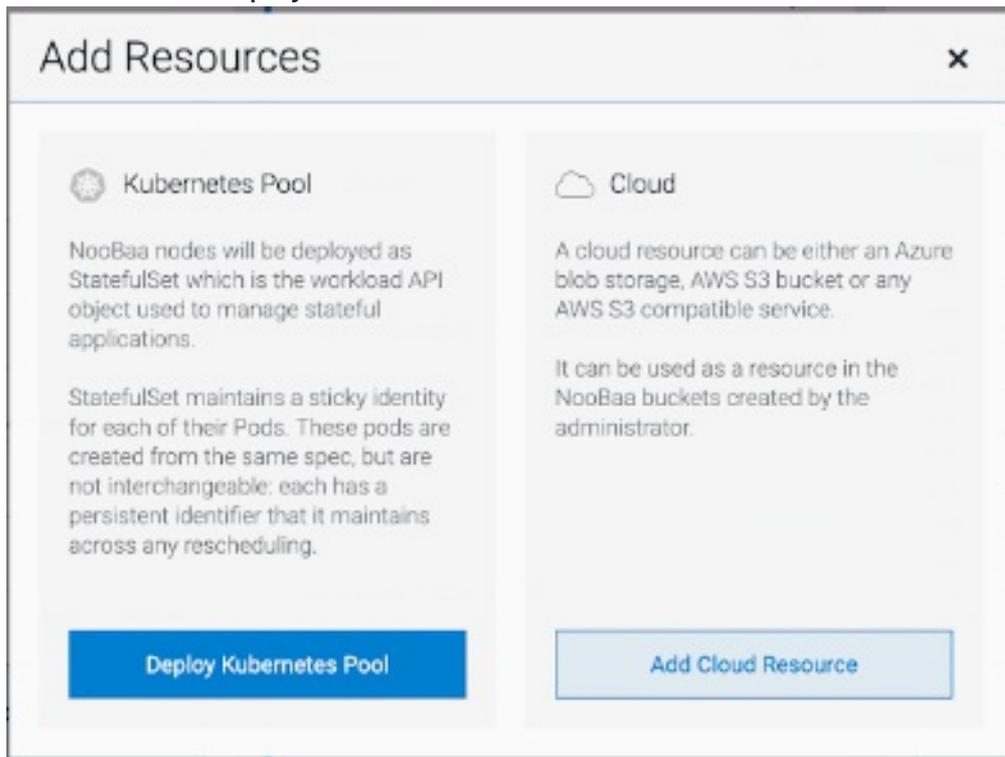
Multicloud Object Gateway のストレージノードは 1 つ以上の永続ボリュームに割り当てられた NooBaa デモンコンテナであり、ローカルオブジェクトサービスデータストレージに使用されます。NooBaa デモンは Kubernetes ノードにデプロイできます。これは、StatefulSet Pod で設定される Kubernetes プールを作成して実行できます。

## 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):

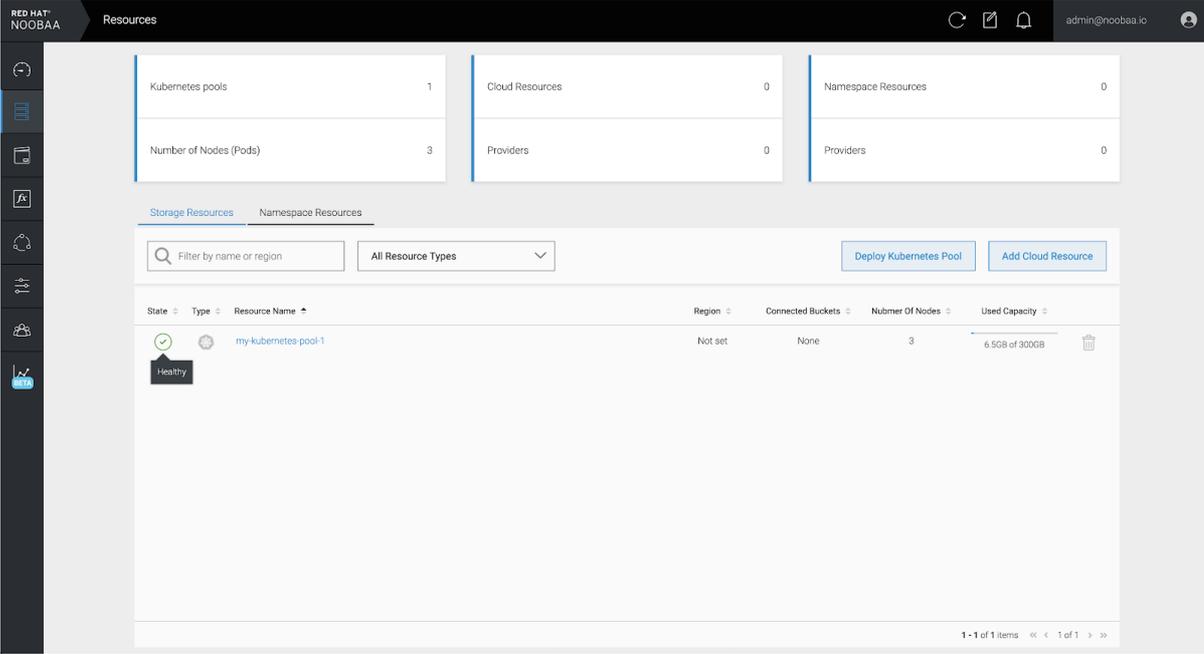
Node PV Size:     
This cannot be changed later on

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。



The screenshot displays the Red Hat NOOBAA Resources management interface. At the top, there are three summary cards: 'Kubernetes pools' (1), 'Cloud Resources' (0), and 'Namespace Resources' (0). Below these, a table lists 'Number of Nodes (Pods)' as 3 and 'Providers' as 0. The main area is titled 'Storage Resources' and contains a search bar, a filter dropdown set to 'All Resource Types', and two buttons: 'Deploy Kubernetes Pool' and 'Add Cloud Resource'. A table below shows the resource details:

State	Type	Resource Name	Region	Connected Buckets	Number Of Nodes	Used Capacity
Healthy		my.kubernetes.pool-1	Not set	None	3	6.5GB of 300GB

The interface also includes a sidebar with navigation icons and a footer indicating '1 - 1 of 1 items'.

## 第9章 永続ボリューム要求の管理

### 9.1. 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**)。

## 9.2. 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。

### 9.3. 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。

### 9.4. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) の拡張

OpenShift Container Storage 4.6 では、Persistent Volume Claim (永続ボリューム要求、PVC) を拡張する機能が導入され、永続ストレージリソース管理の柔軟性が向上します。

拡張は、以下の永続ボリュームでサポートされます。

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



## 警告

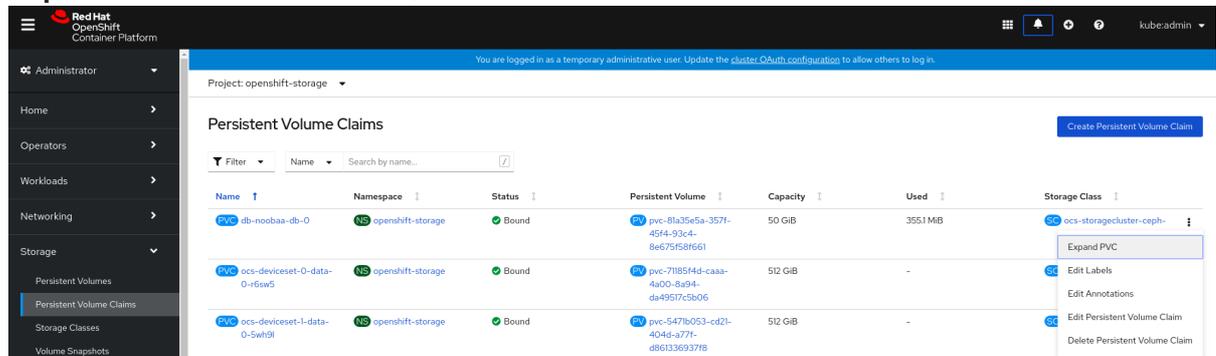
OSD および MON PVC の拡張機能は Red Hat によってサポートされていません。

## 前提条件

- OpenShift Web コンソールへの管理者アクセス。

## 手順

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



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

## Expand Persistent Volume Claim

Increase the capacity of claim `db-noobaa-db-0`. This can be a time-consuming process.

Size \*

Cancel

Expand

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



## 注記

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

## 9.5. 動的プロビジョニング

### 9.5.1. 動的プロビジョニングについて

StorageClass リソースオブジェクトは、要求可能なストレージを記述し、分類するほか、要求に応じて動的にプロビジョニングされるストレージのパラメーターを渡すための手段を提供します。

StorageClass オブジェクトは、さまざまなレベルのストレージおよびストレージへのアクセスを制御するための管理メカニズムとしても機能します。クラスター管理者 (**cluster-admin**) またはストレージ管理者 (**storage-admin**) は、ユーザーが基礎となるストレージボリュームソースに関する詳しい知識なしに要求できる StorageClass オブジェクトを定義し、作成します。

OpenShift Container Storage 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 Storage 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。

### 9.5.2. 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。

### 9.5.3. 利用可能な動的プロビジョニングプラグイン

OpenShift Container Storage 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	<b>kubernetes.io/cinder</b>	
AWS Elastic Block Store (EBS)	<b>kubernetes.io/aws-ebs</b>	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 <b>Key=kubernetes.io/cluster/&lt;cluster_name&gt;,Value=&lt;cluster_id&gt;</b> のタグを付けます。ここで、 <b>&lt;cluster_name&gt;</b> および <b>&lt;cluster_id&gt;</b> 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	<b>kubernetes.io/azure-disk</b>	
Azure File	<b>kubernetes.io/azure-file</b>	<b>persistent-volume-binder</b> 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)	<b>kubernetes.io/gce-pd</b>	マルチゾーン設定では、GCE プロジェクトごとに OpenShift Container Storage 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。

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
VMware vSphere	<b>kubernetes.io/vsphere-volume</b>	



### 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10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リングすることはできません。

### 10.1. 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

- 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。

1. OpenShift Web コンソールで、**Storage** → **Persistent Volume Claims** をクリックします。
2. ボリュームのスナップショットを作成するには、以下のいずれかを実行します。
  - 必要な PVC の横にある Action メニュー (⋮) → **Create Snapshot** をクリックします。
  - スナップショットを作成する PVC をクリックし、**Actions** → **Create Snapshot** をクリックします。
3. ボリュームスナップショットの **Name** を入力します。
4. ドロップダウンリストから **Snapshot Class** を選択します。
5. **Create** をクリックします。作成されるボリュームスナップショットの Details ページにリダイレクトされます。

Volume Snapshots ページで以下を実行します。

1. OpenShift Web コンソールで **Storage** → **Volume Snapshots** をクリックします。
2. **Volume Snapshots** ページで、**Create Volume Snapshot** をクリックします。

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

### 検証手順

- PVC の **Details** ページに移動し、**Volume Snapshots** タブをクリックしてボリュームスナップショットの一覧を表示します。新規スナップショットが一覧表示されていることを確認します。
- OpenShift Web コンソールで **Storage** → **Volume Snapshots** をクリックします。新規スナップショットが一覧表示されていることを確認します。
- ボリュームスナップショットが **Ready** 状態になるまで待機します。

## 10.2. 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. 新規 PVC として復元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 (⋮) → **Restore as new PVC** をクリックします。
4. 新規 PVC の名前を入力します。
5. **Storage Class** 名を選択します。



### 注記

(Rados Block Device (RBD) の場合) 親 PVC と同じプールが指定されるストレージクラスを選択する必要があります。

6. **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 ページの使用

1. OpenShift Web コンソールで **Storage** → **Volume Snapshots** をクリックします。
2. 必要なボリュームスナップショットの横にある Action Menu (⋮) → **Restore as new PVC** をクリックします。
3. 新規 PVC の名前を入力します。
4. **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。

5. **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。



#### 注記

ボリュームスナップショットの復元時に、PVC は親 PVC が存在する場合にのみ、親 PVC のアクセスモードで作成されます。それ以外の場合は、PVC は ReadWriteOnce (RWO) アクセスモードでのみ作成されます。現時点で、OpenShift Web コンソールを使用してアクセスモードを指定することはできません。ただし、YAML を使用して CLI からアクセスモードを指定できます。詳細は、[ボリュームスナップショットの復元](#) について参照してください。

### 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。

## 10.3. ボリュームスナップショットの削除

### 前提条件

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

### 手順

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

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

Volume Snapshots ページで以下を実行します。

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

#### 検証手順

- 削除されたボリュームスナップショットが PVC の詳細ページの **Volume Snapshots** タブにないことを確認します。
- **Storage → Volume Snapshots** をクリックし、削除されたボリュームスナップショットが一覧表示されていないことを確認します。

## 第11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。

### 11.1. 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. **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。



#### 注記

クローンは、親 PVC のアクセスモードで作成されます。現時点で、OpenShift Web コンソール UI を使用してアクセスモードを指定することはできません。ただし、YAML を使用して CLI からアクセスモードを指定できます。詳細は、[CSI ボリュームクローンのプロビジョニング](#) について参照してください。

5. 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12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。

- 「Red Hat OpenStack Platform 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」
- 「Red Hat OpenStack Platform 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障害のあるノードの置き換え」

### 12.1. RED HAT OPENSTACK PLATFORM 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、Red Hat OpenStack Platform 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) クラスタでデータ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。  
直前の手順で特定された新規ノードごとに、以下を実行します。

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

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

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

```
$ lsblk
```

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

## 12.2. RED HAT OPENSTACK PLATFORM 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障害のあるノードの置き換え

以下の手順に従って、OpenShift Container Storage の Red Hat OpenStack Platform 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。

#### 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=""
```

9. [オプション]: 障害のある Red Hat OpenStack Platform インスタンスが自動的に削除されない場合には、Red Hat OpenStack Platform コンソールからインスタンスを終了します。

## 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) クラスターでデータ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。  
直前の手順で特定された新規ノードごとに、以下を実行します。

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

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

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

```
$ lsblk
```

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

## 第13章 ストレージデバイスの置き換え

### 13.1. RED HAT OPENSTACK PLATFORM 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障害のあるストレージデバイスの置き換え

以下の手順を使用して、Red Hat OpenStack Platform にデプロイされた OpenShift Container Storage のストレージデバイスを置き換えます。この手順は、新規ボリュームで新規の Persistent Volume Claim(永続ボリューム要求、PVC)を作成し、古いオブジェクトストレージデバイス (OSD) を削除するのに役立ちます。

#### 手順

1. 置き換える必要がある OSD と、その OSD がスケジュールされている OpenShift Container Platform ノードを特定します。

```
$ oc get -n openshift-storage pods -l app=rook-ceph-osd -o wide
```

出力例:

```
rook-ceph-osd-0-6d77d6c7c6-m8xj6 0/1 CrashLoopBackOff 0 24h 10.129.0.16
compute-2 <none> <none>
rook-ceph-osd-1-85d99fb95f-2svc7 1/1 Running 0 24h 10.128.2.24 compute-
0 <none> <none>
rook-ceph-osd-2-6c66cdb977-jp542 1/1 Running 0 24h 10.130.0.18 compute-
1 <none> <none>
```

この例では、**rook-ceph-osd-0-6d77d6c7c6-m8xj6** を置き換える必要があり、**compute-2** は OSD がスケジュールされる OpenShift Container platform ノードです。



#### 注記

置き換える OSD が正常である場合、Pod のステータスは **Running** になります。

2. 置き換えられる OSD の OSD デプロイメントをスケールダウンします。

```
# osd_id_to_remove=0
# oc scale -n openshift-storage deployment rook-ceph-osd-${osd_id_to_remove} --replicas=0
```

ここで、**osd\_id\_to\_remove** は **rook-ceph-osd** 接頭辞の直後にくる Pod 名の整数です。この例では、デプロイメント名は **rook-ceph-osd-0** です。

出力例:

```
deployment.extensions/rook-ceph-osd-0 scaled
```

3. **rook-ceph-osd** Pod が停止していることを確認します。

```
# oc get -n openshift-storage pods -l ceph-osd-id=${osd_id_to_remove}
```

出力例:

```
No resources found.
```



### 注記

**rook-ceph-osd** Pod が **terminating** 状態にある場合は、**force** オプションを使用して Pod を削除します。

```
# oc delete pod rook-ceph-osd-0-6d77d6c7c6-m8xj6 --force --grace-period=0
```

出力例:

```
warning: Immediate deletion does not wait for confirmation that the running
resource has been terminated. The resource may continue to run on the
cluster indefinitely.
pod "rook-ceph-osd-0-6d77d6c7c6-m8xj6" force deleted
```

4. 新規 OSD を追加できるようにクラスターから古い OSD を削除します。

a. 古い **ocs-osd-removal** ジョブを削除します。

```
$ oc delete -n openshift-storage job ocs-osd-removal-${osd_id_to_remove}
```

出力例:

```
job.batch "ocs-osd-removal-0" deleted
```

b. **openshift-storage** プロジェクトを変更します。

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

c. クラスターから以前の OSD を削除します。

```
$ oc process -n openshift-storage ocs-osd-removal -p
FAILED_OSD_IDS=${osd_id_to_remove} |oc create -n openshift-storage -f -
```



### 警告

この手順により、OSD はクラスターから完全に削除されます。**osd\_id\_to\_remove** の正しい値が指定されていることを確認します。

5. **ocs-osd-removal** Pod のステータスをチェックして、OSD が正常に削除されたことを確認します。**Completed** のステータスで、OSD の削除ジョブが正常に完了したことを確認します。

```
# oc get pod -l job-name=ocs-osd-removal-${osd_id_to_remove} -n openshift-storage
```



## 注記

**ocs-osd-removal** が失敗し、Pod が予想される **Completed** の状態にない場合、追加のデバッグのために Pod ログを確認します。以下に例を示します。

```
# oc logs -l job-name=ocs-osd-removal-${osd_id_to_remove} -n openshift-storage --tail=-1
```

6. 暗号化がインストール時に有効にされている場合は、それぞれの OpenShift Container Storage ノードから削除された 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。

- a. **ocs-osd-removal-job** Pod のログから、置き換えられた OSD の PVC 名を取得します。

```
$ oc logs -l job-name=ocs-osd-removal-job -n openshift-storage --tail=-1 |egrep -i 'pvc|deviceset'
```

以下に例を示します。

```
2021-05-12 14:31:34.666000 I | cephosd: removing the OSD PVC "ocs-deviceset-xxxx-xxx-xxx-xxx"
```

- b. 手順 #1 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、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
```

- ii. 直前の手順で特定された PVC 名に基づいて関連するデバイス名を検索します。

```
sh-4.4# dmsetup ls| grep <pvc name>
ocs-deviceset-xxx-xxx-xxx-xxx-block-dmccrypt (253:0)
```

- iii. マップ済みデバイスを削除します。

```
$ cryptsetup luksClose --debug --verbose ocs-deviceset-xxx-xxx-xxx-xxx-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. **ocs-osd-removal** ジョブを削除します。

```
# oc delete -n openshift-storage job ocs-osd-removal-${osd_id_to_remove}
```

出力例:

```
job.batch "ocs-osd-removal-0" deleted
```

## 検証手順

1. 新しい OSD が実行されていることを確認します。

```
# oc get -n openshift-storage pods -l app=rook-ceph-osd
```

出力例:

```
rook-ceph-osd-0-5f7f4747d4-snshw          1/1   Running   0      4m47s
rook-ceph-osd-1-85d99fb95f-2svc7        1/1   Running   0      1d20h
rook-ceph-osd-2-6c66cdb977-jp542        1/1   Running   0      1d20h
```

2. **Bound** 状態の新しい PVC が作成されていることを確認します。

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

出力例:

```
NAME                                STATUS VOLUME                                CAPACITY ACCESS
MODES STORAGECLASS                    AGE
db-noobaa-db-0                      Bound  pvc-b44ebb5e-3c67-4000-998e-304752deb5a7 50Gi
RWO  ocs-storagecluster-ceph-rbd        6d
ocs-deviceset-0-data-0-gwb5l        Bound  pvc-bea680cd-7278-463d-a4f6-3eb5d3d0defe
512Gi RWO  standard                               94s
ocs-deviceset-1-data-0-w9pjm        Bound  pvc-01aded83-6ef1-42d1-a32e-6ca0964b96d4
```

```
512Gi RWO standard 6d
ocs-deviceset-2-data-0-7bxcq Bound pvc-5d07cd6c-23cb-468c-89c1-72d07040e308
512Gi RWO standard 6d
```

3. (オプション) クラスタでデータ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-
qrgqm
```

- 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。

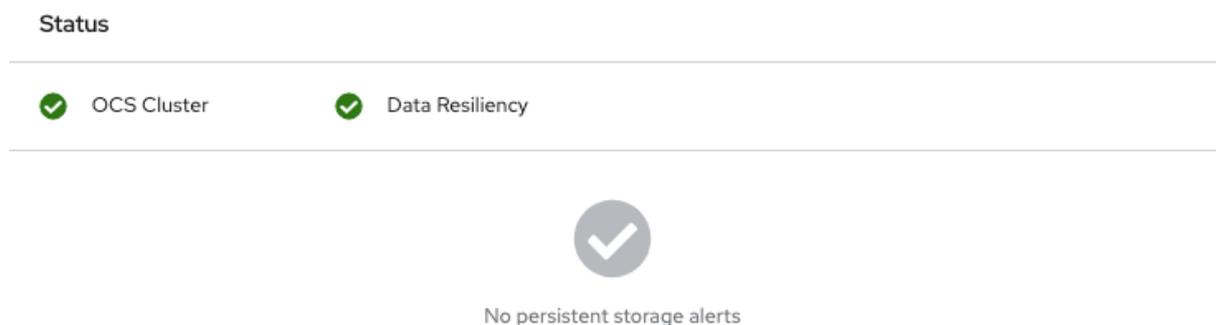
```
$ oc debug node/<node name>
$ chroot /host
```

- 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
```

4. OpenShift Web コンソールにログインし、ストレージダッシュボードを表示します。

図13.1 デバイスの置き換え後の OpenShift Container Platform ストレージダッシュボードの OSD ステータス



## 第14章 OPENSIFT CONTAINER STORAGE の更新

### 14.1. 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 (4.5 と 4.6 間などのマイナーリリース間、または 4.6.0 と 4.6.1 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 [クラスターの更新](#) ドキュメントに従って **OpenShift Container Platform** を更新します。
2. **OpenShift Container Storage** を更新します。
  - a. 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。
    - 更新のために非接続またはプロキシ環境を準備するには、[Operator Lifecycle Manager](#) を制限されたネットワーク で使用するための Operator ガイドを参照してください。
    - [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。

### 14.2. 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 ドキュメントを参照してください: [Updating an Operator catalog image](#)

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。

## 14.2.1. 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": "*****",
    }
  }
}
```

### 14.2.2. 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:v4.6 \
  --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
```

### 14.2.3. 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。

### 14.2.4. 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](#)

## 14.3. 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。

### 14.3.1. 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 4.5.X または 4.6.Y に更新する場合、[クラスターの更新](#) を参照してください。
- Red Hat OpenShift Container Storage チャンネルを **stable-4.5** から **stable-4.6** に切り替えます。チャンネルの詳細は、[OpenShift Container Storage](#) 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: 4.5 から 4.6 に更新) にのみチャンネルを切り換える必要があり、4.6 のバッチの更新間に更新する場合 (例: 4.6.0 から 4.6.1 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 バージョンがチャン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。

#### 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 4.5 から 4.6 に更新された後も、**Version** フィールドには依然として 4.5 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。

## 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。

### 14.3.2. 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 4.5.X または 4.6.Y に更新する場合、[クラスターの更新](#) を参照してください。
- Red Hat OpenShift Container Storage チャンネルを **stable-4.5** から **stable-4.6** に切り替えます。チャンネルの詳細は、[OpenShift Container Storage](#) 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: 4.5 から 4.6 に更新) にのみチャンネルを切り換える必要があり、4.6 のバッチの更新間に更新する場合 (例: 4.6.0 から 4.6.1 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。

### 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 4.5 から 4.6 に更新された後も、**Version** フィールドには依然として 4.5 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。

### 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。

