



Red Hat OpenShift Container Storage 4.5

Red Hat OpenShift Container Storage の管理

クラスターおよびストレージ管理者の手順

Red Hat OpenShift Container Storage 4.5 Red Hat OpenShift Container Storage の管理

クラスターおよびストレージ管理者の手順

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/Managing_OpenShift_Container_Storage.ent file |.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本書では、OpenShift Container Storage クラスターを管理する方法について説明します。

目次

第1章 概要	5
第2章 OPENSIFT CONTAINER PLATFORM サービスのストレージの設定	6
2.1. OPENSIFT CONTAINER STORAGE を使用するためのイメージレジストリーの設定	6
2.2. OPENSIFT CONTAINER STORAGE を使用するためのモニタリングの設定	8
2.3. OPENSIFT CONTAINER STORAGE のクラスターロギング	11
2.3.1. 永続ストレージの設定	11
2.3.2. OpenShift Container Storage を使用するためのクラスターロギングの設定	12
第3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	15
第4章 ストレージノードのスケールリング	17
4.1. ストレージノードのスケールリングの要件	17
4.1.1. Red Hat OpenShift Container Storage のサポートされるデプロイメント	17
4.2. ストレージ容量のスケールアップ	18
4.2.1. AWS または VMware インフラストラクチャーの OpenShift Container Storage ノードへの容量の追加によるストレージのスケールアップ	18
4.2.2. ローカルストレージデバイスを使用した OpenShift Container Storage ノードへの容量の追加によるストレージのスケールアップ	20
4.3. ストレージ容量のスケールアウト	24
4.3.1. ノードの追加	24
4.3.1.1. AWS のインストーラーでプロビジョニングされるインフラストラクチャーへのノードの追加	24
4.3.1.2. AWS または VMware のユーザーによってプロビジョニングされるインフラストラクチャーへのノードの追加	25
4.3.1.3. ローカルストレージデバイスを使用したノードの追加	26
4.3.2. 新規ノードの追加の確認	28
4.3.3. ストレージ容量のスケールアップ	28
第5章 PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) の管理	29
5.1. OPENSIFT CONTAINER PLATFORM を使用するためのアプリケーション POD の設定	29
5.2. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) 要求ステータスの表示	30
5.3. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) 要求イベントの確認	31
5.4. PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) の拡張	31
5.5. 動的プロビジョニング	33
5.5.1. 動的プロビジョニングについて	33
5.5.2. OpenShift Container Storage の動的プロビジョニング	33
5.5.3. 利用可能な動的プロビジョニングプラグイン	34
第6章 CONTAINER STORAGE INTERFACE (CSI) コンポーネントの配置の管理	36
第7章 MULTICLOUD OBJECT GATEWAY	37
7.1. MULTICLOUD OBJECT GATEWAY について	37
7.2. アプリケーションの使用による MULTICLOUD OBJECT GATEWAY へのアクセス	37
7.2.1. ターミナルから Multicloud Object Gateway へのアクセス	37
7.2.2. MCG コマンドラインインターフェースからの Multicloud Object Gateway へのアクセス	39
7.3. MULTICLOUD OBJECT GATEWAY コンソールへのユーザーアクセスの許可	42
7.4. ハイブリッドまたはマルチクラウド用のストレージリソースの追加	43
7.4.1. 新規バックスタアの作成	43
7.4.2. MCG コマンドラインインターフェー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	45
7.4.2.1. AWS でサポートされるバックスタアの作成	45
7.4.2.2. IBM COS でサポートされるバックスタアの作成	47

7.4.2.3. Azure でサポートされるバックスタアの作成	48
7.4.2.4. GCP でサポートされるバックスタアの作成	50
7.4.2.5. ローカル永続ボリュームでサポートされるバックスタアの作成	51
7.4.3. s3 と互換性のある Multicloud Object Gateway バックスタアの作成	52
7.4.4. ユーザーインターフェー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	54
7.4.5. 新規バケットクラスの作成	56
7.5. 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	58
7.5.1. MCG コマンドラインインターフェー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	59
7.5.2. 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	59
7.5.3. ユーザーインターフェー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	60
7.6. MULTICLOUD OBJECT GATEWAY のバケットポリシー	61
7.6.1. バケットポリシーについて	61
7.6.2. バケットポリシーの使用	62
7.6.3. Multicloud Object Gateway での AWS S3 ユーザーの作成	63
7.7. OBJECT BUCKET CLAIM (オブジェクトバケット要求)	66
7.7.1. 動的 Object Bucket Claim (オブジェクトバケット要求)	66
7.7.2. コマンドラインインターフェースを使用した Object Bucket Claim (オブジェクトバケット要求) の作成	68
7.7.3. 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	71
7.7.4. Object Bucket Claim (オブジェクトバケット要求) のデプロイメントへの割り当て	74
7.7.5. OpenShift Web コンソールを使用したオブジェクトバケットの表示	75
7.7.6. Object Bucket Claim (オブジェクトバケット要求) の削除	76
7.8. 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	77
7.8.1. Multicloud Object Gateway での S3 エンドポイント	77
7.8.2. ストレージノードを使用したスケールアップ	77

第8章 RADOS OBJECT GATEWAY S3 エンドポイントへのアクセス 81

第9章 OPENSIFT CONTAINER STORAGE のストレージノードの置き換え 83

9.1. AWS にデプロイされる OPENSIFT CONTAINER STORAGE	83
9.1.1. ユーザーによって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 AWS ノードの置き換え	83
9.1.2. 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 AWS ノードの置き換え	85
9.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗した AWS ノードの置き換え	86
9.1.4. インストーラーでプロビジョニングされるインフラストラクチャーでの失敗した AWS ノードの置き換え	87
9.2. VMWARE にデプロイされる OPENSIFT CONTAINER STORAGE	88
9.2.1. ユーザーによって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 VMware ノードの置き換え	88
9.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗した VMware ノードの置き換え	90
9.3. ローカルストレージデバイスを使用した OPENSIFT CONTAINER STORAGE のデプロイ	91
9.3.1. ベアメタルインフラストラクチャーでのストレージノードの置き換え	91
9.3.1.1. ユーザーによって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	91
9.3.1.2. ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗したノードの置き換え	96
9.3.2. Amazon EC2 インフラストラクチャーでのストレージノードの置き換え	101
9.3.2.1. ユーザーによって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 Amazon EC2 ノードの置き換え	101
9.3.2.2. 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 Amazon EC2 ノードの置き換え	107
9.3.2.3. ユーザーによって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 Amazon EC2	

ノードの置き換え	112
9.3.2.4. インストーラーでプロビジョニングされるインフラストラクチャーでの失敗した AWS ノードの置き換え	118
9.3.3. VMWare インフラストラクチャーでのストレージノードの置き換え	124
9.3.3.1. VMware のユーザーによって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	124
9.3.3.2. VMware ユーザーによって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	129
第10章 ストレージデバイスの置き換え	135
10.1. AWS への OPENSIFT CONTAINER STORAGE の動的プロビジョニング	135
10.1.1. AWS のユーザーによってプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え	135
10.1.2. AWS のインストーラーでプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え	135
10.2. VMWARE への OPENSIFT CONTAINER STORAGE の動的プロビジョニング	136
10.2.1. VMware のユーザーによってプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え	136
10.3. ローカルストレージデバイスを使用した OPENSIFT CONTAINER STORAGE のデプロイ	141
10.3.1. Amazon EC2 インフラストラクチャーでの障害のあるストレージノードの置き換え	141
10.3.2. VMware およびベアメタルインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え	141
第11章 OPENSIFT CONTAINER STORAGE の更新	151
11.1. 内部モードでの OPENSIFT CONTAINER STORAGE の更新	151
11.1.1. 内部モードでの OpenShift Container Storage Operator の自動更新の有効化	151
11.1.2. 内部モードでの OpenShift Container Storage Operator の手動による更新	153
11.2. 外部モードでの OPENSIFT CONTAINER STORAGE の更新	155
11.2.1. 外部モードでの OpenShift Container Storage Operator の自動更新の有効化	155
11.2.2. 外部モードでの OpenShift Container Storage Operator の手動による更新	157
11.3. 非接続環境での更新の準備	159
11.3.1. ミラーレジストリーの認証情報の追加	159
11.3.2. Red Hat Operator カタログのビルドおよびミラーリング	161
11.3.3. Operator imageContentSourcePolicy を作成します。	161
11.3.4. redhat-operator CatalogSource の更新	162
11.3.5. 更新の継続	163

第1章 概要

OpenShift Container Storage の管理は、管理者が Red Hat OpenShift Container Storage クラスターの管理方法を理解するのに役立ちます。

ほとんどの管理タスクは単一リソースに重点を置いています。本書の章は、管理者が変更するリソース別に分けられています。

- [2章 OpenShift Container Platform サービスのストレージの設定](#) コアとなる OpenShift Container Platform サービスに OpenShift Container Storage を使用する方法を説明します。
- [3章 OpenShift Container Storage を使用した OpenShift Container Platform アプリケーションのサポート](#) OpenShift Container Storage を使用するように OpenShift Container Platform アプリケーションを設定する方法についての情報を提供します。
- [4章 ストレージノードのスケーリング](#) OpenShift Container Storage ノードのストレージ容量のスケーリングについての情報を提供します。
- [5章 Persistent Volume Claim \(永続ボリューム要求、PVC\) の管理](#) Persistent Volume Claim (永続ボリューム要求、PVC) の要求の管理とそれらの要求への対応の自動化に関する情報を提供します。
- [6章 Container Storage Interface \(CSI\) コンポーネントの配置の管理](#) 容認を設定してノードでコンテナストレージのインターフェースコンポーネントを起動する方法についての情報を提供します。
- [7章 Multicloud Object Gateway](#) Multicloud Object Gateway に関する情報を提供します。
- [9章 OpenShift Container Storage のストレージノードの置き換え](#) OpenShift Container Storage の AWS UPI、AWS IPI、および VMware UPI で動作するノードまたは障害のあるノードを置き換える方法を説明します。
- [10章 ストレージデバイスの置き換え](#) では、VMware インフラストラクチャーを動的にデプロイされた OpenShift Container Storage およびローカルストレージデバイスを使用してデプロイされた OpenShift Container Storage のデバイスを置き換える方法を説明します。
- [11章 OpenShift Container Storage の更新](#) OpenShift Container Storage クラスターのアップグレード手順を説明します。

第2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。

2.1. OPENSIFT CONTAINER STORAGE を使用するためのイメージレジストリーの設定

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

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



警告

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

前提条件

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

手順

1. 使用するイメージレジストリーの Persistent Volume Claim (永続ボリューム要求、PVC) を作成します。
 - a. OpenShift Web コンソールで、**Storage** → **Persistent Volume Claims** をクリックします。
 - b. **Project** を **openshift-image-registry** に設定します。
 - c. **Create Persistent Volume Claim** をクリックします。
 - i. 上記で取得した利用可能なストレージクラス一覧から、プロビジョナー **openshift-storage.cephfs.csi.ceph.com** で **Storage Class** を指定します。
 - ii. Persistent Volume Claim (永続ボリューム要求、PVC) の **Name** を指定します (例: **ocs4registry**)。
 - iii. **Shared Access (RWX)** の **Access Mode** を指定します。
 - iv. 100 GB 以上の **Size** を指定します。
 - v. **Create** をクリックします。

新規 Persistent Volume Claim (永続ボリューム要求、PVC) のステータスが **Bound** として一覧表示されるまで待機します。
2. クラスターのイメージレジストリーを、新規の Persistent Volume Claim (永続ボリューム要求、PVC) を使用するように設定します。
 - a. **Administration** → **Custom Resource Definitions** をクリックします。
 - b. **imageregistry.operator.openshift.io** グループに関連付けられた **Config** カスタムリソース定義をクリックします。
 - c. **Instances** タブをクリックします。
 - d. クラスターインスタンスの横にある **Action メニュー (⋮)** → **Edit Config** をクリックします。
 - e. イメージレジストリーの新規 Persistent Volume Claim (永続ボリューム要求、PVC) を追加します。
 - i. 以下を **spec:** の下に追加し、必要に応じて既存の **storage:** セクションを置き換えます。

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

以下に例を示します。

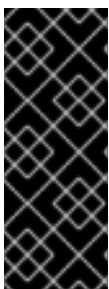
```
storage:
  pvc:
    claim: ocs4registry
```

- ii. **保存** 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**)。

2.2. OPENSIFT CONTAINER STORAGE を使用するためのモニタリングの設定

OpenShift Container Storage は、Prometheus および AlertManager で構成されるモニタリングスタックを提供します。

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



重要

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

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「[永続ストレージの設定](#)」の **Prometheus メトリクスデータの保持期間の設定** についてのサブセクションを参照してください。

前提条件

- OpenShift Web コンソールへの管理者アクセスがある。
- OpenShift Container Storage Operator が **openshift-storage** namespace にインストールされ、実行されている。OpenShift Web コンソールで、**Operators** → **Installed Operators** をクリックし、インストールされた Operator を表示します。

- モニタリング Operator が **openshift-monitoring** namespace にインストールされ、実行されている。OpenShift Web コンソールで、**Administration** → **Cluster Settings** → **Cluster Operators** をクリックしてクラスター 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。
 - a. 新規の **alertmanager-main-*** Pod をクリックし、Pod の詳細を表示します。
 - b. **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
ocs-alertmanager-claim-alertmanager-main-0	/alertmanager	alertmanager-db	ocs-alertmanager-claim-alertmanager-main-0	Read/Write	alertmanager

3. 新規 **prometheus-k8s-*** Pod が **Running** 状態が表示されることを確認します。
 - a. 新規 **prometheus-k8s-*** Pod をクリックし、Pod の詳細を表示します。
 - b. **Volumes** までスクロールダウンし、ボリュームに新規の Persistent Volume Claim (永続ボリューム要求、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	PVC ocs-prometheus-claim-prometheus-k8s-0	Read/Write	prometheus

2.3. 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。

2.3.1. 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](#)」に記載の **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。

2.3.2. OpenShift Container Storage を使用するためのクラスターロギングの設定

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



注記

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

前提条件

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

手順

1. OpenShift Web コンソールの左側のペインから **Administration → Custom Resource Definitions** をクリックします。
2. Custom Resource Definitions ページで、**ClusterLogging** をクリックします。
3. Custom Resource Definition Overview ページで、Actions メニューから **View Instances** を選択するか、または **Instances** タブをクリックします。
4. Cluster Logging ページで、**Create Cluster Logging** をクリックします。データを読み込むためにページを更新する必要がある場合があります。

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

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

6. **保存** 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2.1 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3章 OPENSHIFT CONTAINER STORAGE を使用した OPENSHIFT CONTAINER PLATFORM アプリケーションのサ ポート

OpenShift Container Platform のインストール時に OpenShift 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

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変更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およびサブパス（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、PersistenVolumeClaim Overview ページでストレージクラス名を確認します。

第4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内部モードで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 Red Hat 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。

外部モードでストレージをスケーリングするには、[Red Hat Ceph Storage のドキュメント](#) を参照してください。

4.1. 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](#)
 - [ローカルストレージデバイス](#)
 - [容量のプランニング](#)



重要

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。

ストレージ領域が完全になくなる場合は、Red Hat カスタマーポータルにお問い合わせください。

4.1.1. Red Hat OpenShift Container Storage のサポートされるデプロイメント

- ユーザーによってプロビジョニングされるインフラストラクチャー：
 - Amazon Web Services (AWS)
 - VMware
 - ベアメタル
- 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：
 - Amazon Web Services (AWS)

4.2. ストレージ容量のスケールアップ

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ストレージ容量をスケールアップできます。

- ストレージデバイスの動的または自動プロビジョニングを使用する AWS または VMware インフラストラクチャーの場合は、を参照してください。 [「AWS または VMware インフラストラクチャーの OpenShift Container Storage ノードへの容量の追加によるストレージのスケールアップ」](#)
- ローカルストレージデバイスを使用したベアメタル、Amazon EC2 I3、または VMware インフラストラクチャーの場合は、を参照してください。 [「ローカルストレージデバイスを使用した OpenShift Container Storage ノードへの容量の追加によるストレージのスケールアップ」](#)

4.2.1. AWS または VMware インフラストラクチャーの OpenShift Container Storage ノードへの容量の追加によるストレージのスケールアップ

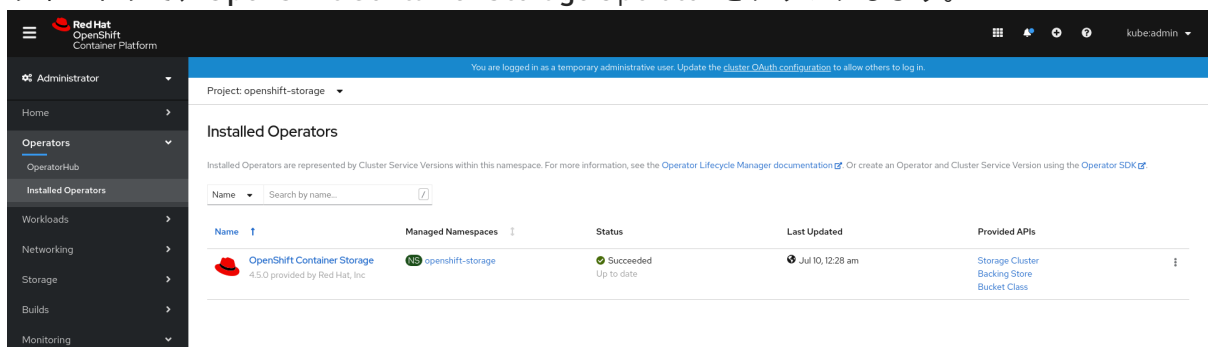
以下の手順を使用して、設定された Red Hat OpenShift Container Storage ワーカーノードにストレージ容量を追加し、パフォーマンスを強化します。

前提条件

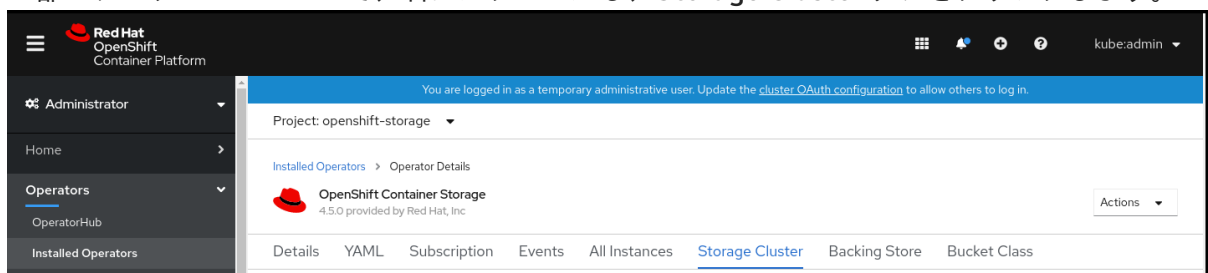
- 実行中の OpenShift Container Storage Platform
- OpenShift Web コンソールの管理者権限

手順

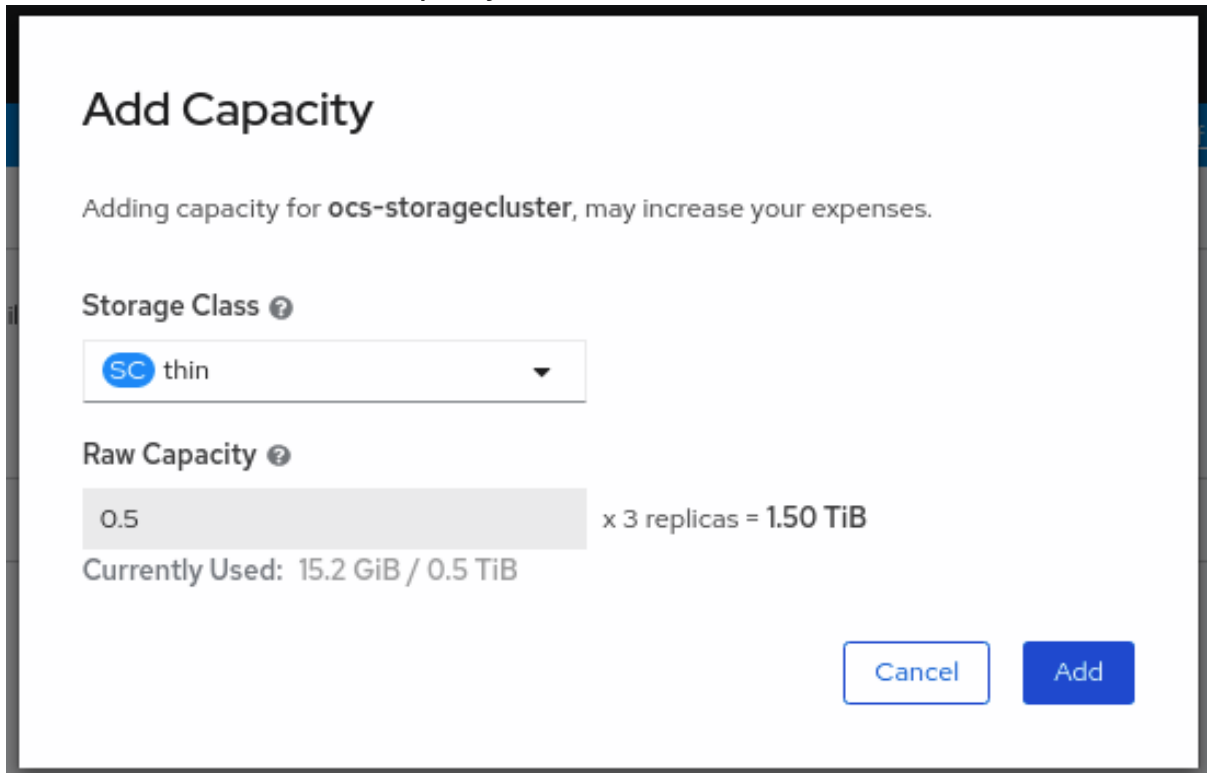
1. OpenShift Web コンソールに移動します。
2. 左側のナビゲーションバーの **Operators** をクリックします。
3. **Installed Operators** を選択します。
4. ウィンドウで、**OpenShift Container Storage Operator** をクリックします。



5. 上部のナビゲーションバーで、右にスクロールし、**Storage Cluster** タブをクリックします。



- 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。
- オプションメニューから **Add Capacity** を選択します。



ダイアログボックスから、要求される追加容量およびストレージクラスを設定できます。**Add capacity** には、インストール時に選択された容量が表示され、容量はこの増分値でのみ追加できます。AWS では、ストレージクラスは **gp2** に設定する必要があります。VMware では、ストレージクラスは **thin** に設定する必要があります。



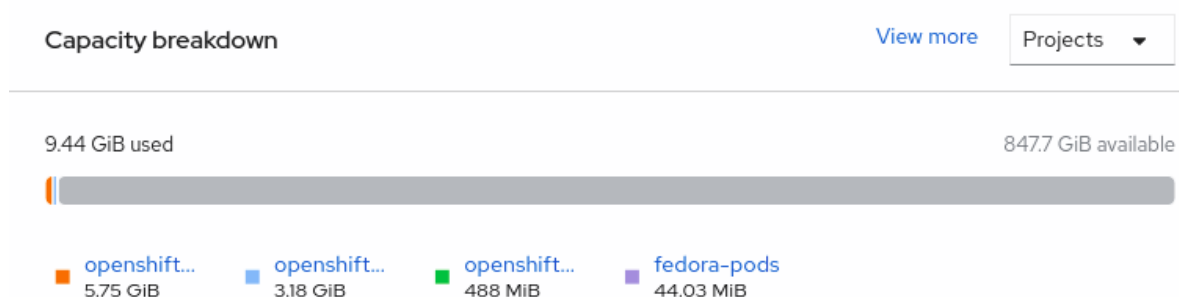
注記

効果的にプロビジョニングされる容量については、OpenShift Container Storage はレプリカ数の3を使用するため、**Raw Capacity** フィールドに入力する値の3倍の値にします。

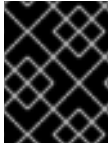
- 設定が終了したら、**Add** をクリックします。ストレージクラスターが **Ready** 状態になるまでに数分待機する必要がある場合があります。

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。



2. 容量は選択に応じて増大することに注意してください。



重要

OpenShift Container Storage 4.2 では、OSD またはノードの縮小によるクラスタの削減はサポートされていません。

4.2.2. ローカルストレージデバイスを使用した OpenShift Container Storage ノードへの容量の追加によるストレージのスケールアップ

以下の手順を使用して、ベアメタルおよび VMware インフラストラクチャーで設定されたローカルストレージベースの OpenShift Container Storage ワーカーノードにストレージ容量（追加のストレージデバイス）を追加します。



重要

Amazon EC2 I3 でのストレージのスケールアップはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。



注記

Amazon EC2 I3 インフラストラクチャーでは、利用可能な両方の NVMe デバイスを使用してデプロイメントが実行されるため、ノードを追加することが容量を追加するための唯一のオプションになります。

前提条件

- OpenShift Container Platform クラスタにログインしている必要があります。
- ローカルストレージ Operator がインストールされている必要があります。次の手順を使用します。以下を参照してください。
 - [ローカルストレージ Operator のベアメタルへのインストール](#)
 - [ローカルストレージ Operator の vSphere クラスタへのインストール](#)
- 3つの OpenShift Container Platform ワーカーノードが必要です。それらのノードには、元の OCS StorageCluster の作成に使用されたものと同じストレージタイプおよびサイズ (例: 2TB NVMe ドライブ) が割り当てられている必要があります。

手順

1. OpenShift Container Storage がインストールされている OpenShift Container Platform ノードにストレージ容量を追加するには、以下を実行する必要があります。
 - a. ワーカーノードごとに少なくとも1つのデバイスを追加するため、利用可能なデバイスの一意の **by-id** 識別子を見つけます。各デプロイメントガイドで説明されている利用可能なストレージデバイスを検索する手順に従ってください。



注記

このプロセスを、ストレージを追加する既存ノードのすべて (3 ノード以上) に対して実行するようにしてください。

- b. 一意のデバイス ID を **LocalVolume** カスタムリソース(CR)に追加します。

```
$ oc edit -n local-storage localvolume local-block
```

出力例:

```
spec:
  logLevel: Normal
  managementState: Managed
  nodeSelector:
    nodeSelectorTerms:
      - matchExpressions:
          - key: cluster.ocs.openshift.io/openshift-storage
            operator: In
            values:
              - ""
  storageClassDevices:
    - devicePaths:
        - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402P51P0GGN
        - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402LM1P0GGN
        - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402M21P0GGN
        - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402B71P0GGN # newly
      added device by-id
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402A31P0GGN # newly
      added device by-id
    - /dev/disk/by-id/nvme-INTEL_SSDPE2KX010T7_PHLF733402Q71P0GGN # newly
      added device by-id
  storageClassName: localblock
  volumeMode: Block
```

CR の編集後に変更を保存するようにしてください。

出力例:

```
localvolume.local.storage.openshift.io/local-block edited
```

この CR で、**by-id** を使用する新規デバイスが追加されていることを確認できます。それぞれの新規デバイスは、3 つのワーカーノードの 1 つの NVMe デバイスにマップされます。

- **nvme-INTEL_SSDPE2KX010T7_PHLF733402B71P0GGN**
- **nvme-INTEL_SSDPE2KX010T7_PHLF733402A31P0GGN**
- **nvme-INTEL_SSDPE2KX010T7_PHLF733402Q71P0GGN**

2. 新規に作成された PV を **localVolume** CR で使用される **storageclass** 名前で表示します。

```
$ oc get pv | grep localblock | grep Available
```

出力例:

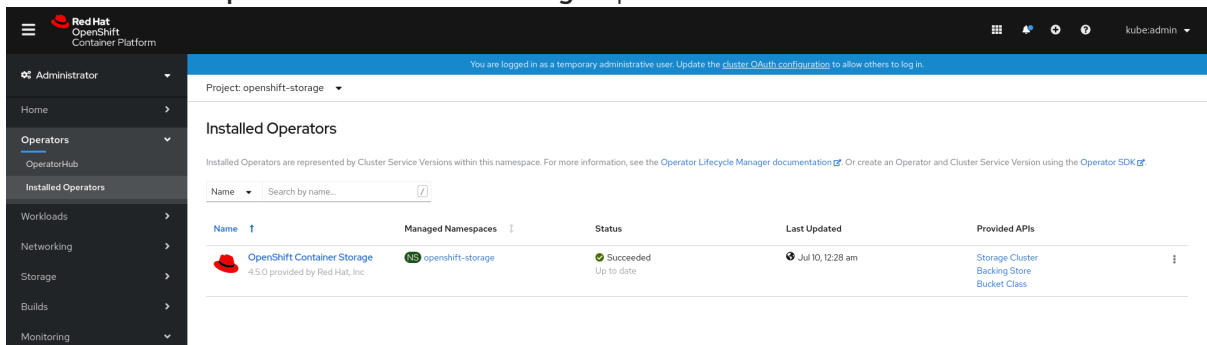
```

local-pv-5ee61dcc 931Gi RWO Delete Available localblock 2m35s
local-pv-b1fa607a 931Gi RWO Delete Available localblock 2m27s
local-pv-e971c51d 931Gi RWO Delete Available localblock 2m22s
...

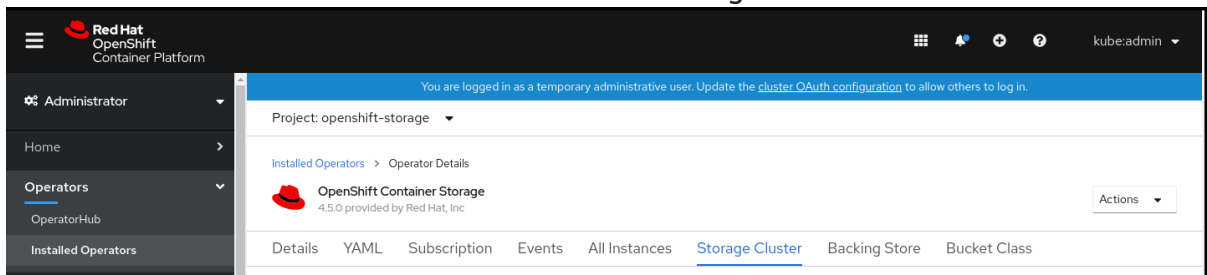
```

新しい OSD に使用されるサイズと同じサイズの 3 つの PV が利用可能です。

3. OpenShift Web コンソールに移動します。
4. 左側のナビゲーションバーの **Operators** をクリックします。
5. **Installed Operators** を選択します。
6. ウィンドウで、**OpenShift Container Storage Operator** をクリックします。



7. 上部のナビゲーションバーで、右にスクロールし、**Storage Cluster** タブをクリックします。



8. 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。
9. 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 localblock ▼

Available capacity: 2.73 TiB / 3 replicas

Cancel

Add

このダイアログボックスで、**Storage Class** 名を **localVolume** CR で使用される名前に設定します。表示される利用可能な容量は、ストレージクラスで利用可能なローカルディスクをベースとしています。

10. 設定が終了したら、**Add** をクリックします。ストレージクラスターが **Ready** 状態になるまでに数分待機する必要がある場合があります。
11. 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。

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

出力例:

```
NAME                READY STATUS RESTARTS AGE
rook-ceph-osd-0-77c4fdb758-qshw4 1/1 Running 0 1h
rook-ceph-osd-1-8645c5fbb6-656ks 1/1 Running 0 1h
rook-ceph-osd-2-86895b854f-r4gt6 1/1 Running 0 1h
rook-ceph-osd-3-dc7f787dd-gdnsz 1/1 Running 0 10m
rook-ceph-osd-4-554b5c46dd-hbf9t 1/1 Running 0 10m
rook-ceph-osd-5-5cf94c4448-k94j6 1/1 Running 0 10m
```

上記の例では、osd-3、osd-4、および osd-5 は、新たに OpenShift Container Storage クラスターに追加される Pod です。

```
$ oc get pvc -n openshift-storage |grep localblock
```

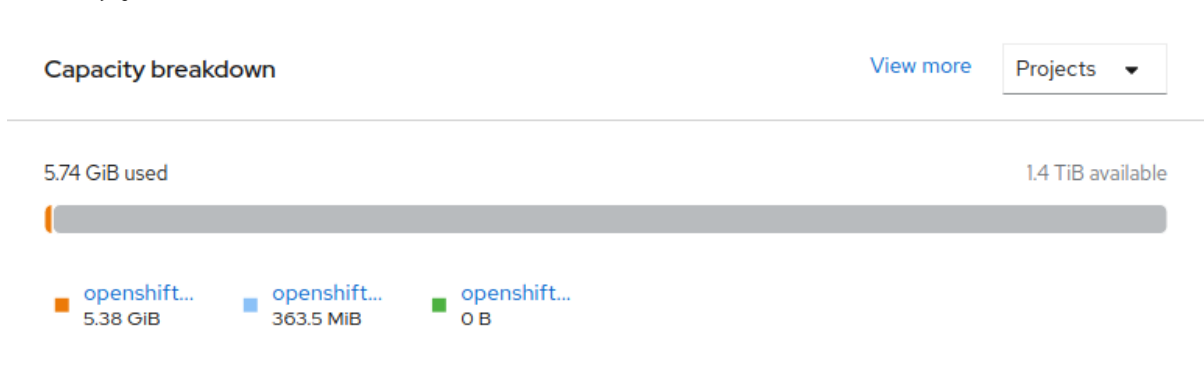
出力例:

```
ocs-deviceset-0-0-qc29m Bound local-pv-fc5562d3 931Gi RWO localblock 1h
ocs-deviceset-0-1-qdmrl Bound local-pv-b1fa607a 931Gi RWO localblock 10m
ocs-deviceset-1-0-mpwmk Bound local-pv-58cdd0bc 931Gi RWO localblock 1h
ocs-deviceset-1-1-85892 Bound local-pv-e971c51d 931Gi RWO localblock 10m
ocs-deviceset-2-0-rl47 Bound local-pv-29d8ad8d 931Gi RWO localblock 1h
ocs-deviceset-2-1-cgth2 Bound local-pv-5ee61dcc 931Gi RWO localblock 10m
```

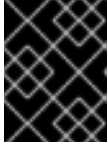
上記の例では、3つの新規 PVC が作成されています。

検証手順

1. **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。



重要

OpenShift Container Storage では、OSD またはノードの縮小によるクラスターの削減はサポートしていません。

4.3. 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、以下の手順を実行する必要があります。

- 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。
- ストレージ容量をスケールアップします。

4.3.1. ノードの追加

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ノードを追加できます。

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ストレージノードを追加できます。

- AWS のインストーラーでプロビジョニングされるインフラストラクチャーの場合は、[を参照してください。「AWS のインストーラーでプロビジョニングされるインフラストラクチャーへのノードの追加」](#)
- AWS または VMware のユーザーによってプロビジョニングされるインフラストラクチャーの場合は、[を参照してください。「AWS または VMware のユーザーによってプロビジョニングされるインフラストラクチャーへのノードの追加」](#)
- ベアメタル、Amazon EC2 I3、または VMware インフラストラクチャーの場合は、[を参照してください。「ローカルストレージデバイスを使用したノードの追加」](#)

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

前提条件

- OpenShift Container Platform (OCP) クラスター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が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たことを確認するには、「[新規ノードの追加の確認](#)」を参照してください。

4.3.1.2. AWS または VMware のユーザーによってプロビジョニングされるインフラストラクチャーへのノードの追加

前提条件

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

手順

1. AWS のユーザーによってプロビジョニングされるインフラストラクチャーまたは VMware のユーザーによってプロビジョニングされるインフラストラクチャーにノードを追加するかどうかに応じて、以下のそれぞれの手順を実行します。
 - AWS の場合:
 - a. 必要なインフラストラクチャーで新規 AWS マシンインスタンスを作成します。「[プラットフォーム要件](#)」を参照してください。
 - b. 新規 AWS マシンインスタンスを使用して新規 OpenShift Container Platform ノードを作成します。
 - VMware の場合:
 - a. 必要なインフラストラクチャーで vSphere に新規の仮想マシンを作成します。「[プラットフォーム要件](#)」を参照してください。
 - b. 新規の仮想マシンを使用して新規 OpenShift Container Platform ワーカーノードを作成します。
2. **Pending** 状態の OpenShift Container Storage に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

3. 新規ノードに必要なすべての OpenShift Container Storage CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

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. 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ースを使用する場合

- 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ースの使用

- 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=""
```



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たことを確認するには、「[新規ノードの追加の確認](#)」を参照してください。

4.3.1.3. ローカルストレージデバイスを使用したノードの追加

以下の手順を使用して、ベアメタル、Amazon EC2、および VMware インフラストラクチャーにノードを追加します。



重要

Amazon EC2 のストレージノードのスケーリングはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

前提条件

- OpenShift Container Platform (OCP) クラスターにログインしている必要があります。
- 3つの OpenShift Container Platform ワーカーノードが必要です。それらのノードには、元の OCS StorageCluster が作成されていたものと同じストレージタイプおよびサイズ（例：2TB NVMe ドライブ）が割り当てられている必要があります。

手順

- ベアメタル、Amazon EC2、または VMware インフラストラクチャーのどれにノードを追加するかどうかに応じて、以下の手順を実行します。
 - Amazon EC2 の場合
 - a. 必要なインフラストラクチャーで新規 Amazon EC2 I3 マシンインスタンスを作成します。「[Creating a MachineSet in AWS](#)」および「[Platform requirements](#)」を参照してください。

- b. 新規 Amazon EC2 I3 マシンインスタンスを使用して新規 OpenShift Container Platform ノードを作成します。
- VMware の場合:
 - a. 必要なインフラストラクチャーで vSphere に新規の仮想マシンを作成します。「[プラットフォーム要件](#)」を参照してください。
 - b. 新規の仮想マシンを使用して新規 OpenShift Container Platform ワーカーノードを作成します。
 - ベアメタルの場合:
 - a. 必要なインフラストラクチャーで新規のベアメタルマシンを取得します。「[プラットフォーム要件](#)」を参照してください。
 - b. 新規ベアメタルマシンを使用して新規 OpenShift Container Platform ノードを作成します。
2. **Pending** 状態の OpenShift Container Storage に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

3. 新規ノードに必要なすべての OpenShift Container Storage CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

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. 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ースを使用する場合

- 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ースの使用

- 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=""
```



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たことを確認するには、「[新規ノードの追加の確認](#)」を参照してください。

4.3.2. 新規ノードの追加の確認

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

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

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

- **csi-cephfsplugin-***
- **csi-rbdplugin-***

4.3.3. ストレージ容量のスケールアップ

ストレージの容量をスケールアップするには、[容量の追加によるストレージのスケールアップ](#)について参照してください。

第5章 PERSISTENT VOLUME CLAIM (永続ボリューム要求、PVC) の管理

5.1. 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**)。

5.2. 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。

5.3. 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。

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

OpenShift Container Storage 4.5 では、Persistent Volume Claim (永続ボリューム要求、PVC) をテクノロジープレビュー機能として拡張する機能が導入されました。これにより、永続ストレージリソースの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)。



重要

CSI ボリュームの拡張は、テクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

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



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。



注記

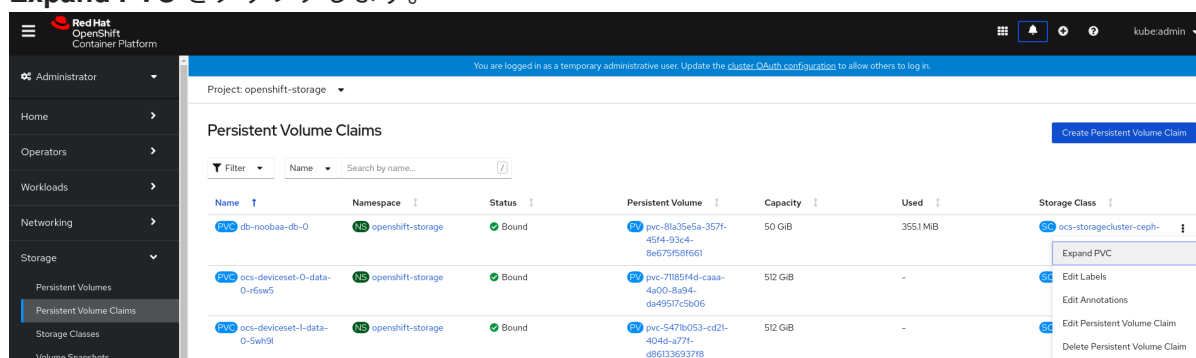
このテクノロジープレビュー機能は、OpenShift Container Storage バージョン 4.5 の新規インストールでのみ利用できます。これは、以前の OpenShift Container Storage リリースからアップグレードされたクラスターには適用されません。

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 *

50	GiB ▼
----	-------

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。

5.5. 動的プロビジョニング

5.5.1. 動的プロビジョニングについて

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

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

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

OpenShift Container Platform では、数多くのストレージタイプを永続ボリュームとして使用することができます。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。

5.5.2. 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 4.5 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 4.5 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。

5.5.3. 利用可能な動的プロビジョニングプラグイン

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

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

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



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6章 CONTAINER STORAGE INTERFACE (CSI) コンポーネントの配置の管理

各クラスターは、**infra** や **storage** ノードなどの数多くの専用ノードで構成されます。ただし、カスタムテイントを持つ **infra** ノードは、ノードで OpenShift Container Storage Persistent Volume Claims (永続ボリューム要求、PVC) を使用することができません。そのため、このようなノードを使用する必要がある場合は、容認を設定してノードで **csi-plugins** を起動することができます。詳細は、<https://access.redhat.com/solutions/4827161> を参照してください。

手順

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

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

2. **configmap** を表示して、追加された容認を確認します。

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

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

```
apiVersion: v1
data:
[...]
```

```
CSI_PLUGIN_TOLERATIONS: |
- effect: NoSchedule
  key: nodetype
  operator: Equal
  value: infra
- effect: NoSchedule
  key: node.ocs.openshift.io/storage
  operator: Exists
[...]
```

```
kind: ConfigMap
metadata:
[...]
```

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

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

例:

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

検証手順

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

第7章 MULTICLOUD OBJECT GATEWAY

7.1. 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。

7.2. 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。

RADOS Object Gateway S3 エンドポイントへのアクセスに関する詳細は、[8章 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。

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ー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ースからの Multicloud Object Gateway へのアクセス」](#)

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

手順

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

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

出力は以下のようになります。

```
Name:      noobaa
Namespace: openshift-storage
Labels:    <none>
Annotations: <none>
```

```

API Version: noobaa.io/v1alpha1
Kind:      NooBaa
Metadata:
  Creation Timestamp: 2019-07-29T16:22:06Z
  Generation:        1
  Resource Version:   6718822
  Self Link:          /apis/noobaa.io/v1alpha1/namespaces/openshift-storage/noobaas/noobaa
  UID:                019cfb4a-b21d-11e9-9a02-06c8de012f9e
Spec:
Status:
  Accounts:
    Admin:
      Secret Ref:
        Name:      noobaa-admin
        Namespace: openshift-storage
  Actual Image:  noobaa/noobaa-core:4.0
  Observed Generation: 1
  Phase:         Ready
  Readme:

```

Welcome to NooBaa!

Welcome to NooBaa!

NooBaa Core Version:
NooBaa Operator Version:

Lets get started:

1. Connect to Management console:

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

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

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

```
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。

7.2.2. MCG コマンドラインインターフェースからの 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ー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/soukSiA==

#-----#
#- 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7.1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 OCS 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
```

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

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

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

前提条件

- 実行中の OpenShift Container Storage Platform

手順

1. Multicloud Object Gateway へのアクセスを有効にします。
クラスターで以下の手順を実行します。
 - a. **cluster-admins** グループを作成します。


```
# oc adm groups new cluster-admins
```
 - b. グループを **cluster-admin** ロールにバインドします。


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

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

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

ここで、**<user-name>** は追加するユーザーの名前です。



注記

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

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

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

ここで、**<user-name>** は削除するユーザーの名前です。

検証手順

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

7.4. ハイブリッドまたはマルチクラウド用のストレージリソースの追加

7.4.1. 新規バックアップストアの作成

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

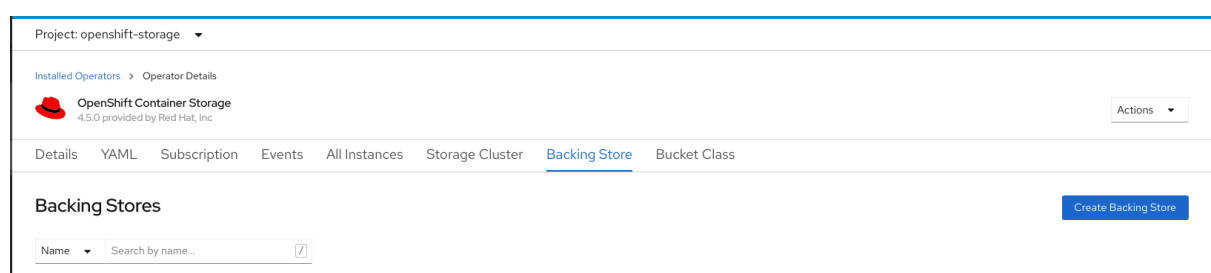
前提条件

- OpenShift への管理者アクセス。

手順

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

図7.1 バックアップストアタブのある OpenShift Container Storage Operator ページ



4. Create Backing Store をクリックします。

図7.2 Create Backing Store ページ

The screenshot shows the 'Create new Backing Store' page in the OpenShift Container Storage console. The page title is 'Create new Backing Store' and the subtitle is 'Storage targets that are used to store chunks of data on MCG buckets.' The form contains the following fields:

- Backing Store Name ***: Input field containing 'my-backingstore'. A note below says 'A unique name for the Backing Store within the project'.
- Provider ***: Dropdown menu showing 'AWS S3'.
- Region ***: Dropdown menu showing 'us-east-1'.
- Endpoint**: Input field.
- Secret ***: Dropdown menu showing 'Select Secret' and a 'Switch to Credentials' link.
- Target Bucket ***: Input field.

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

5. Create New Backing Store ページで、以下を実行します。

- 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ー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。

- 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

- 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。

7.4.2. MCG コマンドラインインターフェー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は、[「s3 と互換性のある Multicloud Object Gateway バックングストアの作成」](#) に進みます。

7.4.2.1. AWS でサポートされるバックングストアの作成

前提条件

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

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

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

手順

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

```
noobaa backingstore create <backingstore_name> --access-key=<AWS ACCESS KEY> --secret-key=<AWS SECRET ACCESS KEY> --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. **<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。

7.4.2.2. IBM COS でサポートされるバックングストアの作成

前提条件

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

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

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

手順

1. MCG コマンドラインインターフェースから、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。

7.4.2.3. Azure でサポートされるバックングストアの作成

前提条件

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

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

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

手順

1. MCG コマンドラインインターフェースから、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、バックリ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、バックリングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。出力は次のようになります。

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

YAML を使用してストレージリソースを追加することもできます。

1. 認証情報でシークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  AccountName: <AZURE ACCOUNT NAME ENCODED IN BASE64>
  AccountKey: <AZURE ACCOUNT KEY ENCODED IN BASE64>
```

- a. Base64 を使用して独自の Azure アカウント名およびアカウントキーを指定し、エンコードし、その結果を **<AZURE ACCOUNT NAME ENCODED IN BASE64>** および **<AZURE ACCOUNT KEY ENCODED IN BASE64>** に使用する必要があります。
 - b. **<backingstore-secret-name>** を一意の名前に置き換えます。
2. 特定のバックリングストアについて以下の YAML を適用します。

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

- a. **<blob-container-name>** を既存の Azure blob コンテナ名に置き換えます。この引数は、Multicloud Object Gateway に対して、バックングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。
- b. **<backingstore-secret-name>** を直前の手順で作成したシークレットの名前に置き換えます。

7.4.2.4. GCP でサポートされるバックングストアの作成

前提条件

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

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

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

手順

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

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

- a. **<backingstore_name>** を、バックングストアの名前に置き換えます。
- b. **<PATH TO GCP PRIVATE KEY JSON FILE>** を、この目的で作成された GCP プライベートキーへのパスに置き換えます。
- c. **<GCP bucket name>** を、既存の GCP オブジェクトストレージバケット名に置き換えます。この引数は、Multicloud Object Gateway に対して、バックングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。
出力は次のようになります。

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

YAML を使用してストレージリソースを追加することもできます。

1. 認証情報でシークレットを作成します。

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

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

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

前提条件

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

```

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

```

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

手順

1. MCG コマンドラインインターフェースから、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。

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

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ースから、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=http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-storage.svc.cluster.local:80
```

- 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 "rgw-resource"
INFO[0002] Created: Secret "backing-store-secret-rgw-resource"
```

YAML を使用してバック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: http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-
storage.svc.cluster.local:80
    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。

7.4.4. ユーザーインターフェー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

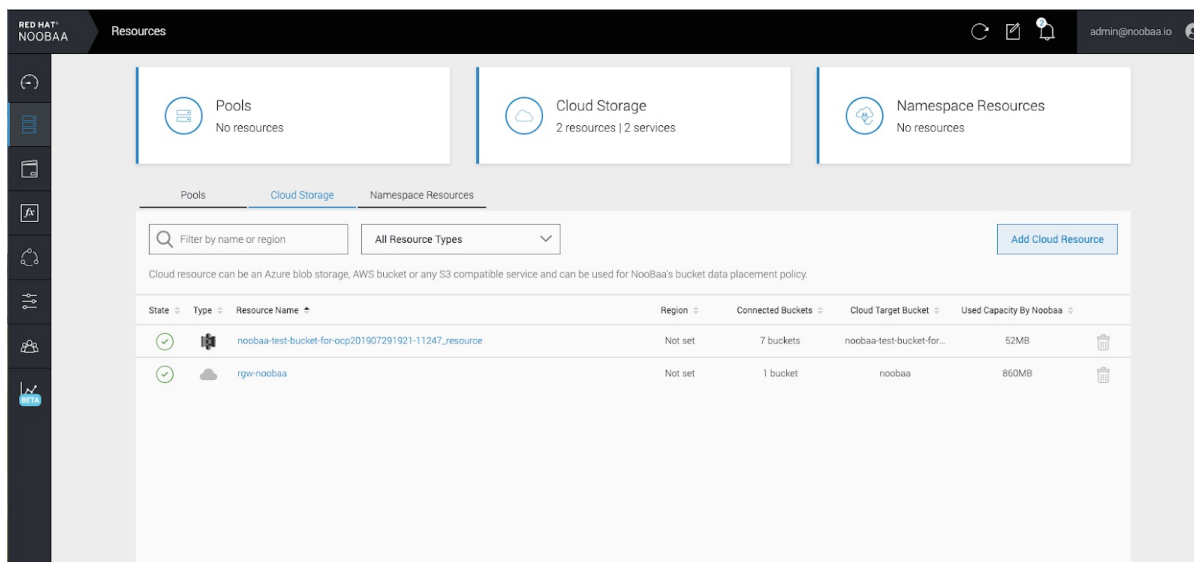
手順

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

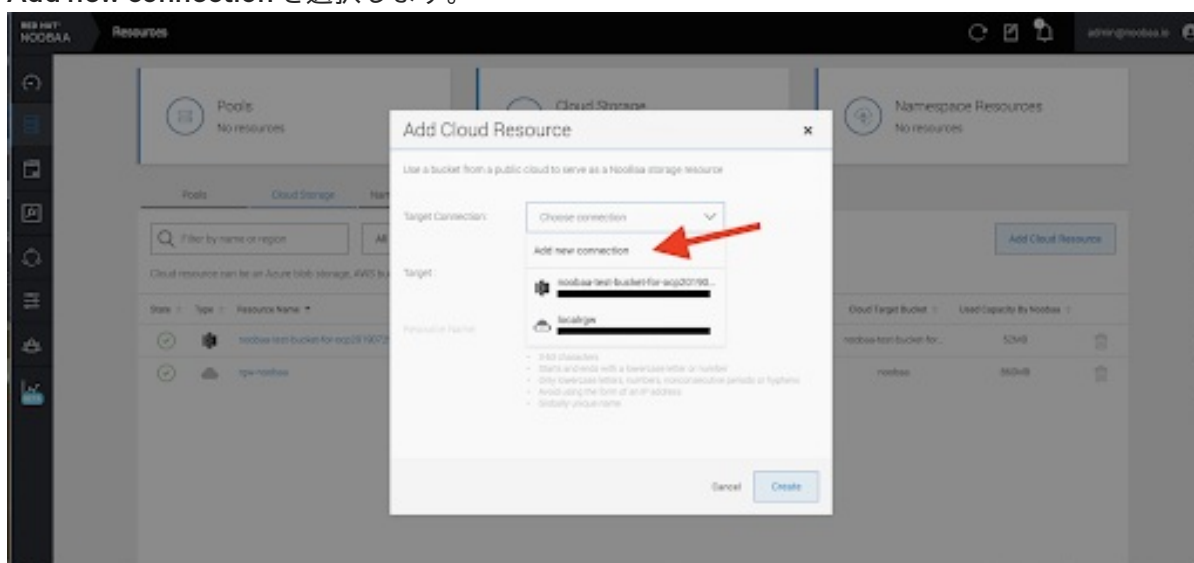
The screenshot shows the OpenShift Storage console interface. The main heading is "Overview". Below it, there are tabs for "Cluster", "Persistent Storage", and "Object Service". The "Object Service" tab is selected. The interface is divided into several sections:

- Details:** Service Name: OpenShift Container Storage (OCS), System Name: noobaa, Provider: VSphere, Version: ocs-operator.v4.5.0.
- Status:** Multi Cloud Object Gateway (green checkmark), Data Resiliency (green checkmark).
- Capacity breakdown:** View more, Projects (dropdown), Not enough usage data.
- Object Data Reduction:** Efficiency Ratio: 1:1, Savings: No Savings.
- Buckets:** 1 NooBaa Bucket (0 Objects), 0 Object Buckets (0 Objects), 0 Object Bucket Claims (0 Objects).
- Data Consumption:** Providers (dropdown), I/O Operations (dropdown), I/O Operations count (bar chart showing 60).
- Activity:** Ongoing: There are no ongoing activities. Recent Events: 18:09 Backing store mode: OPTIMAL.

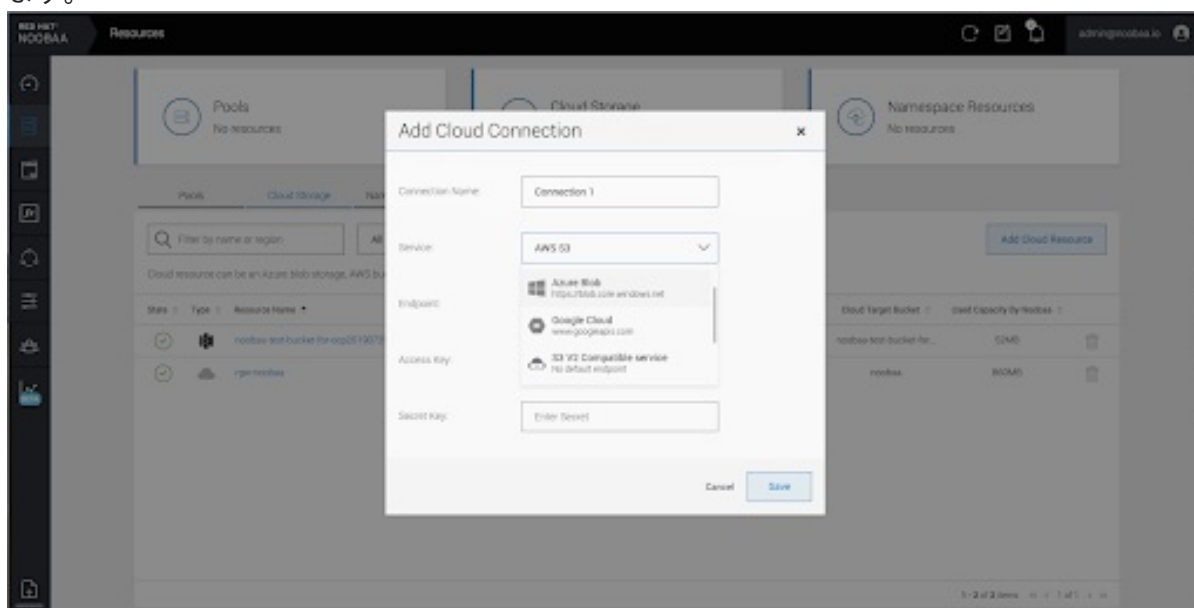
2. 以下に強調表示されているように左側にある **Resources** タブを選択します。設定する一覧から、**Add Cloud Resource** を選択します。



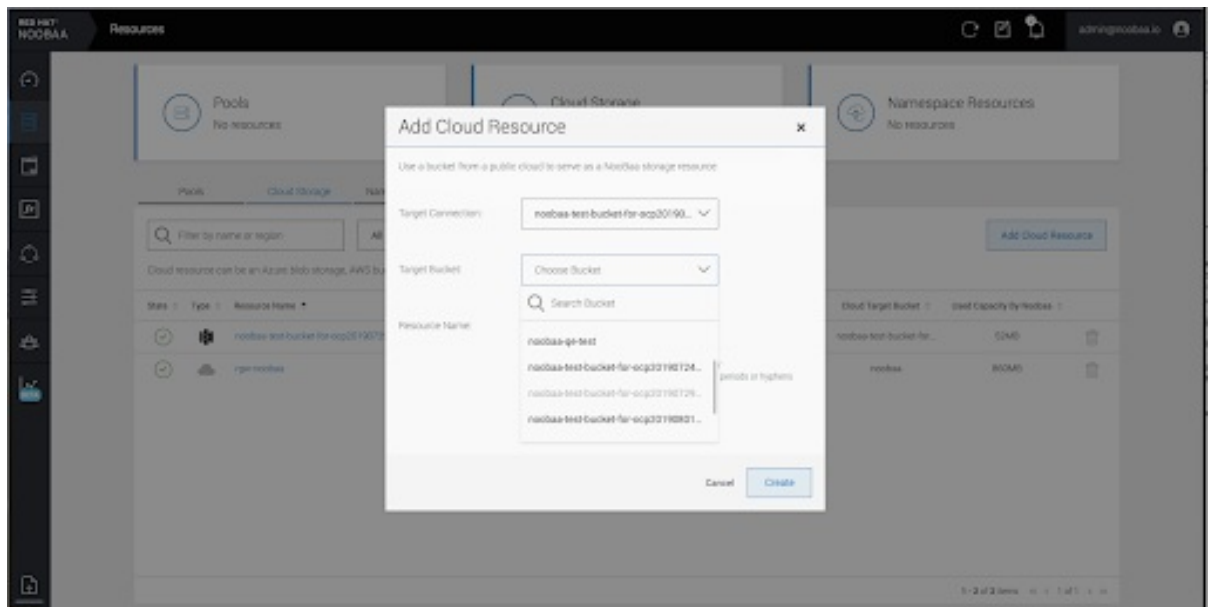
3. Add new connection を選択します。



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。

7.4.5. 新規バケットクラスの作成

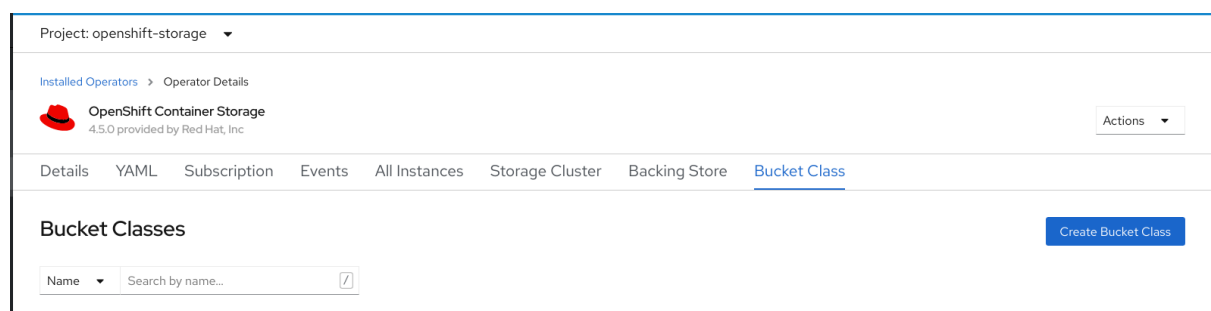
バケットクラスは、OBC (Object Bucket Class) の階層ポリシーおよびデータ配置を定義するバケットのクラスを表す CRD です。

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

手順

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

図7.3 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7.4 Create Bucket Class 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.

1 General

2 Placement Policy

3 Backing Store

4 Review

What is a Bucket Class?
An MCG Bucket's data location is determined by a policy called a Bucket Class
[Learn More](#)

Bucket Class Name *
my-multi-cloud-mirror
A unique name for the Bucket Class within the project.

Description(Optional)

Next Back Cancel

- 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7.5 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.

1 General

2 Placement Policy

3 Backing Store

4 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、**新規バックイングストアを作成**することもできます。

図7.6 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"/> NBS 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つ以上のバックングストアを選択する必要があります。

- Bucket Class 設定を確認し、確認します。

図7.7 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](#)

- [Create Bucket Class](#) をクリックします。

検証手順

- Operators → Installed Operators をクリックします。
- OpenShift Container Storage Operator をクリックします。
- 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。

7.5. 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イ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ー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ー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](#)」

7.5.1. MCG コマンドラインインターフェー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ー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
```

7.5.2. 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。

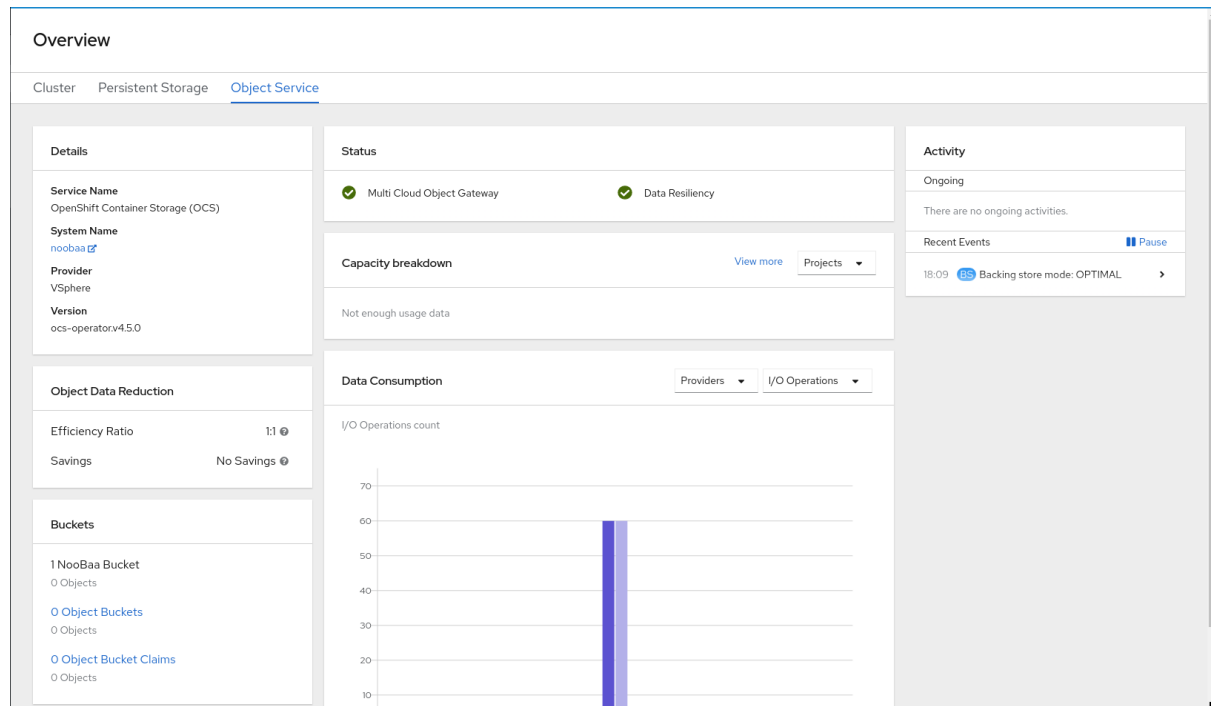
■

additionalConfig:
bucketclass: mirror-to-aws

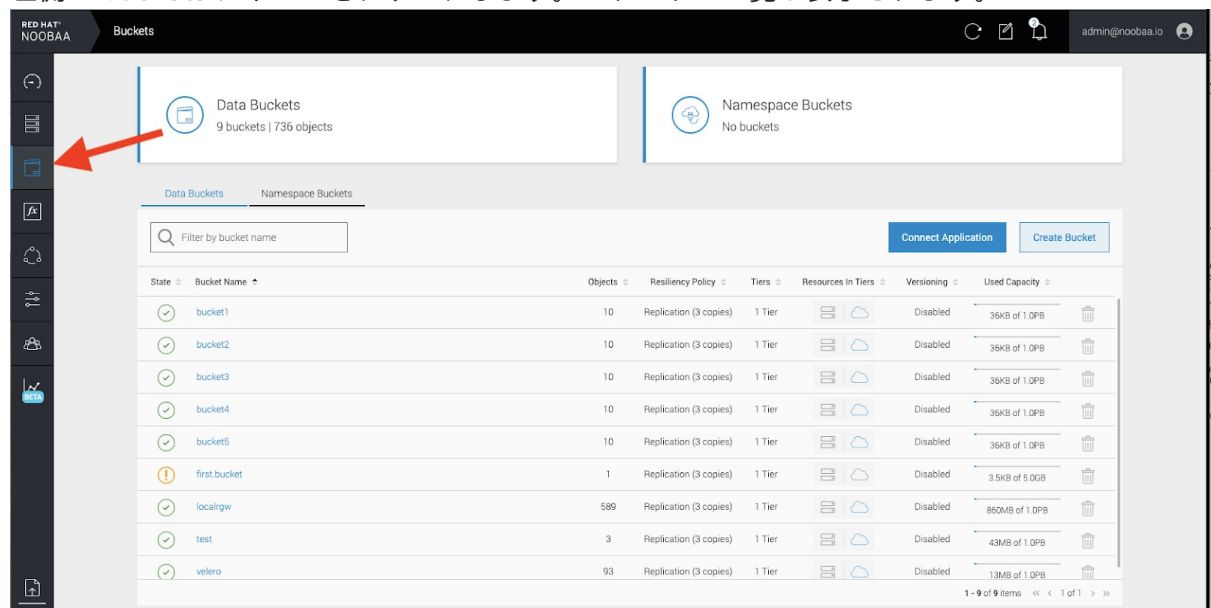
OBC についての詳細は、「Object Bucket Claim (オブジェクトバケット要求)」を参照してください。

7.5.3. ユーザーインターフェー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

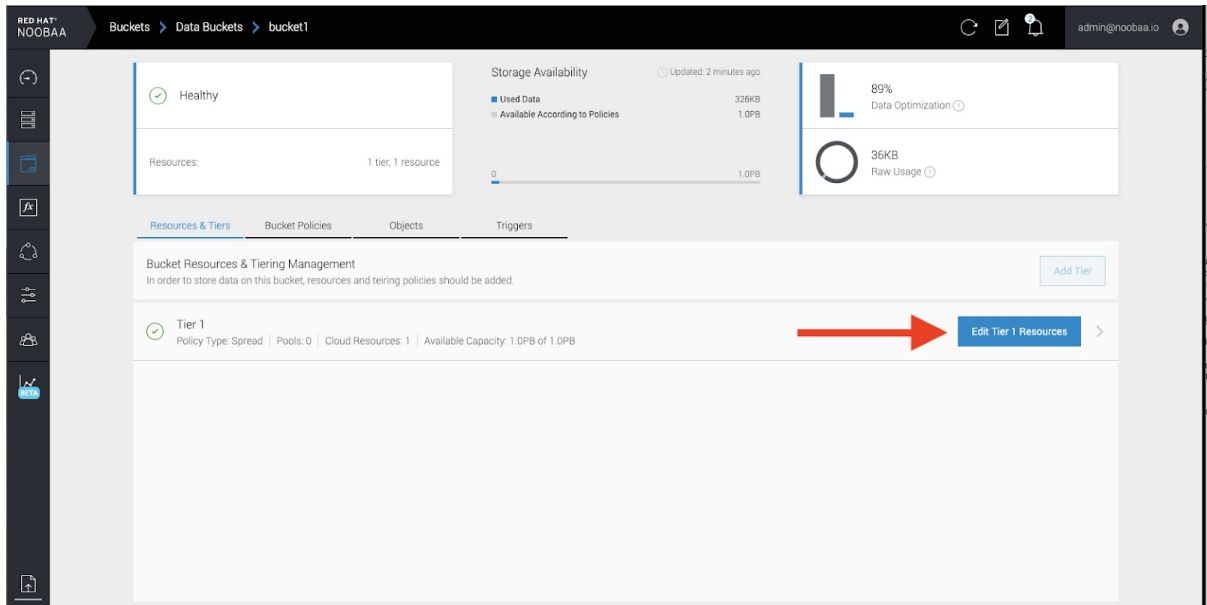
1. OpenShift Storage コンソールで、**Overview** → **Object Service** → に移動し、**noobaa** リンクを選択します。



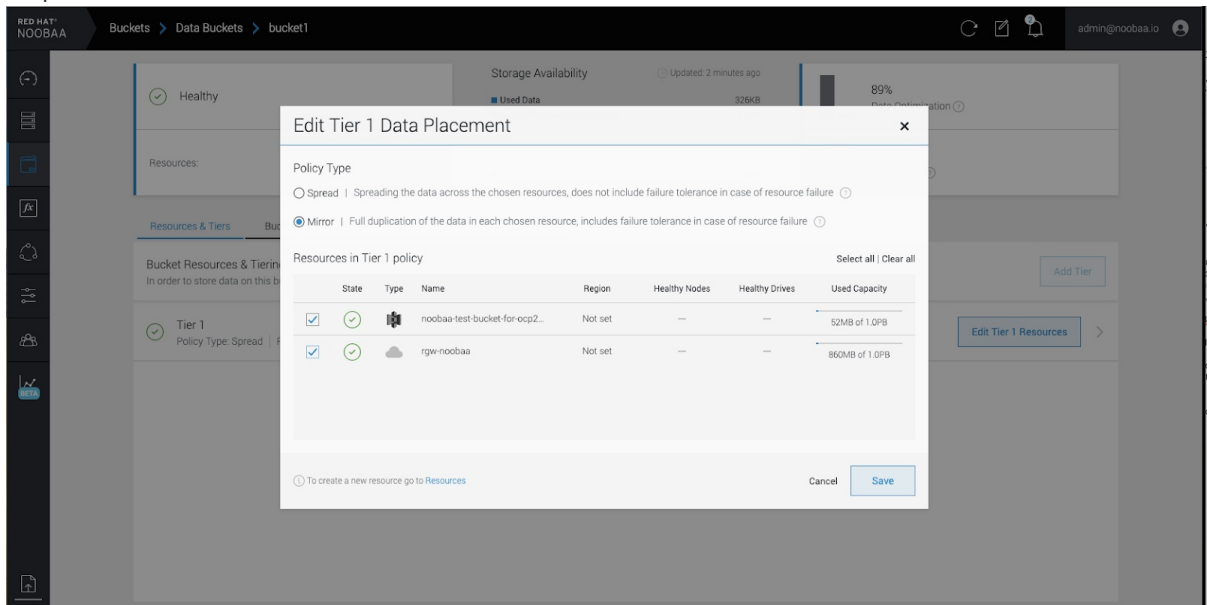
2. 左側の **buckets** アイコンをクリックします。バケットの一覧が表示されます。



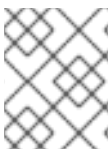
3. 更新するバケットをクリックします。
4. **Edit Tier 1 Resources** をクリックします。



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. **保存** 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。

7.6. 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。

7.6.1. 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。

7.6.2. 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。それらの構成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。

7.6.3. 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、**noobaa** リンクを選択します。

Overview

Cluster Persistent Storage **Object Service**

Details

- Service Name: OpenShift Container Storage (OCS)
- System Name: noobaa
- Provider: VSphere
- Version: ocs-operatorv4.5.0

Status

- Multi Cloud Object Gateway
- Data Resiliency

Capacity breakdown

View more Projects

Not enough usage data

Object Data Reduction

- Efficiency Ratio: 1:1
- Savings: No Savings

Buckets

- 1 NooBaa Bucket: 0 Objects
- Object Buckets: 0 Objects
- Object Bucket Claims: 0 Objects

Data Consumption

Providers I/O Operations

I/O Operations count

70
60
50
40
30
20
10

Activity

Ongoing

There are no ongoing activities.

Recent Events **Pause**

18:09 Backing store mode: OPTIMAL

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

RED HAT NOOBAA Accounts

Accounts

Filter by account name

Create Account

Account Name	Access Type	Default Resource
admin@noobaa.io	Administrator	noobaa-default-backing-store
kube.admin (Current user)	Administrator	noobaa-default-backing-store

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

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

7.7. 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ースを使用した 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。

7.7.1. 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> -o yaml
```

- 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

7.7.2. コマンドラインインターフェースを使用した 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ースを使用して 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ー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ースを使用して、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ー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。

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

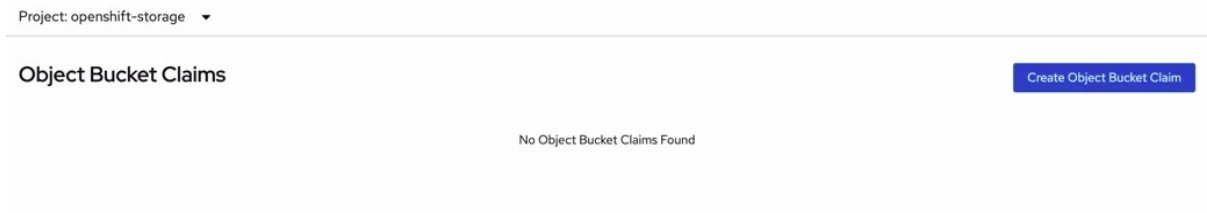
OpenShift Web コンソールを使用して Object Bucket Claim (オブジェクトバケット要求) を作成できます。

前提条件

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

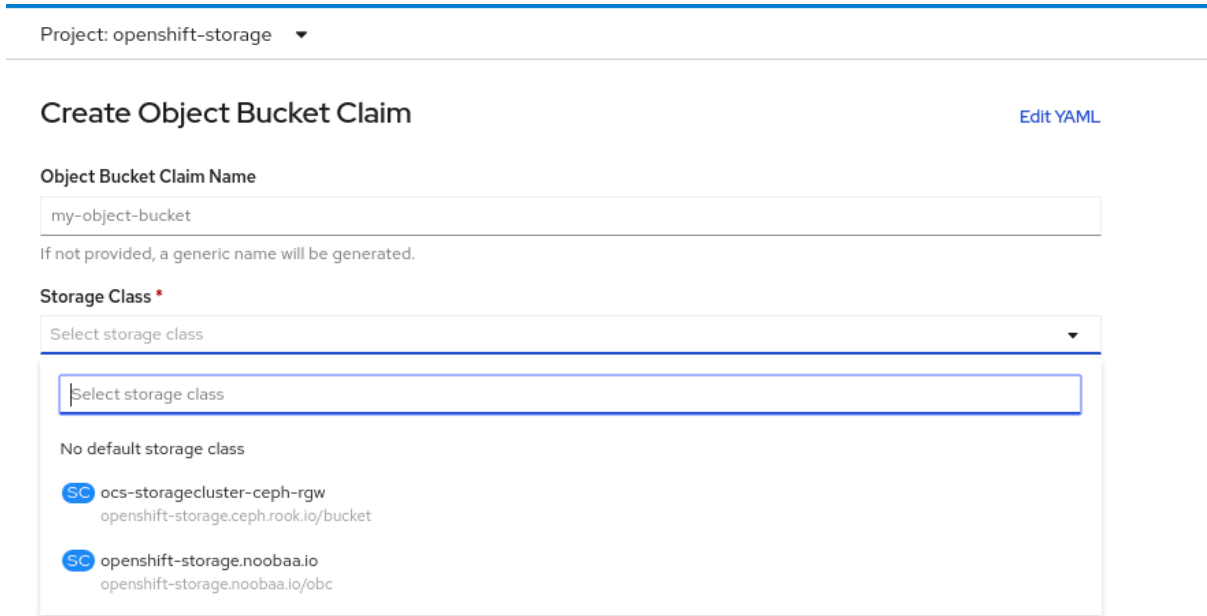
手順

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



- Object Bucket Claim (オブジェクトバケット要求) の名前を入力し、ドロップダウンメニューから、内部または外部かのデプロイメントに応じて適切なストレージクラスとバケットクラスを選択します。

内部モード



デプロイメント後に作成された以下のストレージクラスを使用できます。

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

外部モード

Project: openshift-storage ▼

Create Object Bucket Claim

[Edit YAML](#)

Object Bucket Claim Name

If not provided, a generic name will be generated.

Storage Class *

No default storage class

- SC **ocs-external-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。

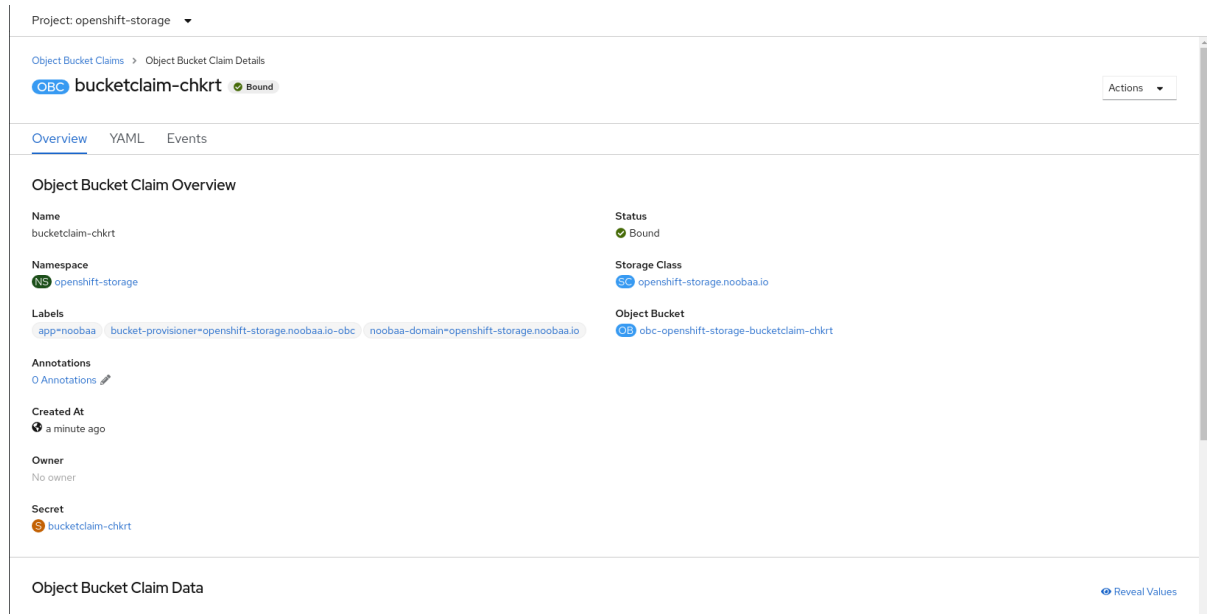
- **ocs-external-storagecluster-ceph-rgw** は Ceph Object Gateway (RGW) を使用します。
- **openshift-storage.noobaa.io** は Multicloud Object Gateway を使用します。



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)」

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

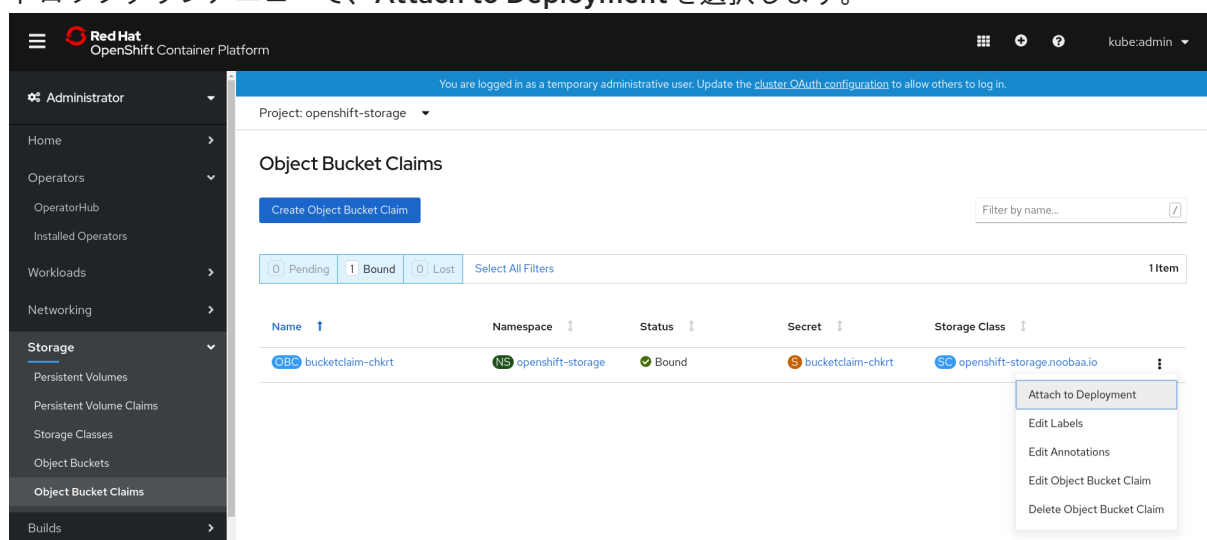
Object Bucket Claim (オブジェクトバケット要求、OBC) は作成後に、特定のデプロイメントに割り当てることができます。

前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

1. 左側のナビゲーションバーで **Storage** → **Object Bucket Claims** をクリックします。
2. 作成した OBC の横にあるアクションメニュー (✓) をクリックします。
3. ドロップダウンメニューで、**Attach to Deployment** を選択します。



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

関連情報

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

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

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

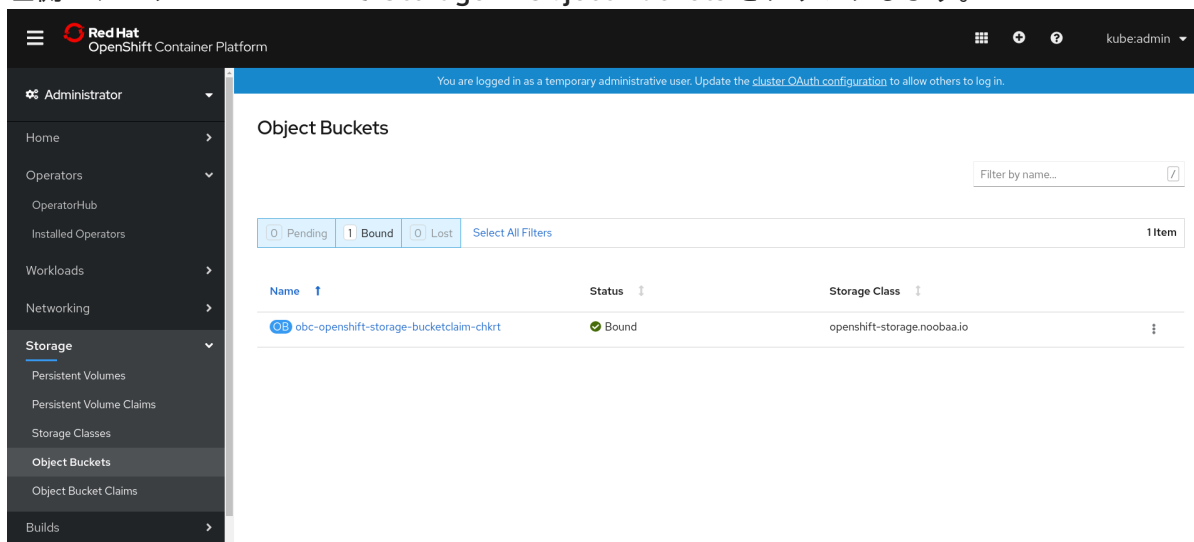
前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

オブジェクトバケットの詳細を表示するには、以下を実行します。

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



特定の OBC の詳細ページに移動し、**Resource** リンクをクリックして、その OBC のオブジェクトバケットを表示します。

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

Object Buckets > Object Bucket Details

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

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

Object Bucket Overview

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

関連情報

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

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

前提条件

- OpenShift Web コンソールへの管理者アクセス。

手順

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

Red Hat OpenShift Container Platform

You are logged in as a temporary administrative user. Update the `cluster OAuth` configuration to allow others to log in.

Project: openshift-storage

Object Bucket Claims

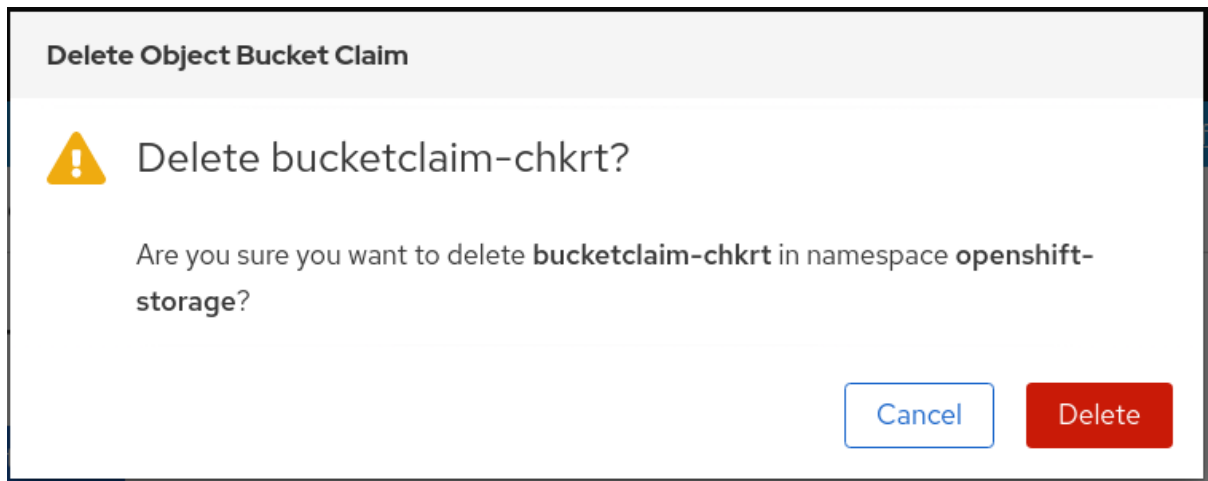
Create Object Bucket Claim Filter by name...

0 Pending 1 Bound 0 Lost Select All Filters 1 Item

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

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

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



4. **Delete** をクリックします。

関連情報

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

7.8. 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 エンドポイントをスケールリングして簡単に対応できます。これはテクノロジープレビュー機能です。

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

重要

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は、テクノロジープレビュー機能です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

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

7.8.1. 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。

7.8.2. ストレージノードを使用したスケーリング

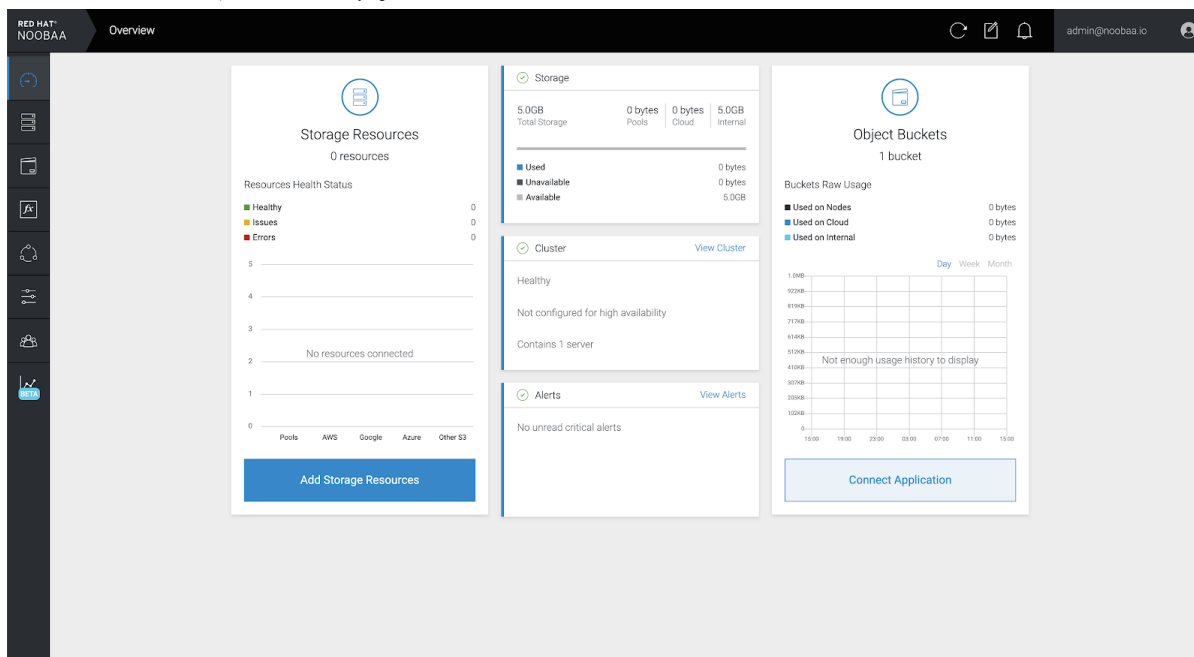
前提条件

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

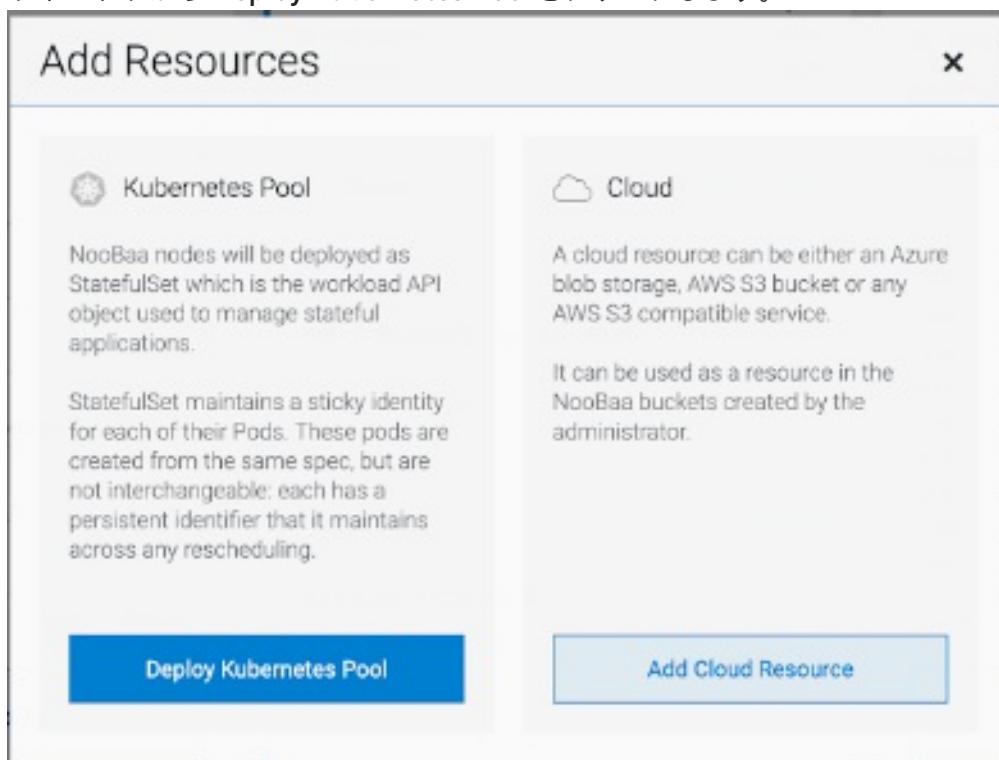
Multicloud Object Gateway のストレージノードは1つ以上の永続ボリュームに割り当てられた NooBaa デーモンコンテナであり、ローカルオブジェクトサービスデータストレージに使用されます。NooBaa デーモンは Kubernetes ノードにデプロイできます。これは、StatefulSet Pod で構成される Kubernetes プールを作成して実行できます。

手順

1. Mult-Cloud Object Gateway ユーザーインターフェースの **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。

The screenshot shows the 'Deploy Kubernetes Pool' dialog box with the 'Create Pool' step selected. It includes a progress indicator with three steps: 1. Create Pool (active), 2. Configure, and 3. Review. A warning message states: 'Kubernetes nodes will be deployed in a kubernetes pool type, and cannot be re-assigned later on to other resources.' Below this is a text input field for 'Kubernetes Pool Name:' with the placeholder 'Type here'. A list of requirements is provided:

- 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

At the bottom, there is a note: 'If you wish to scale up an existing kubernetes pool go to Resources > Pools'. There are 'Cancel' and 'Next' buttons.

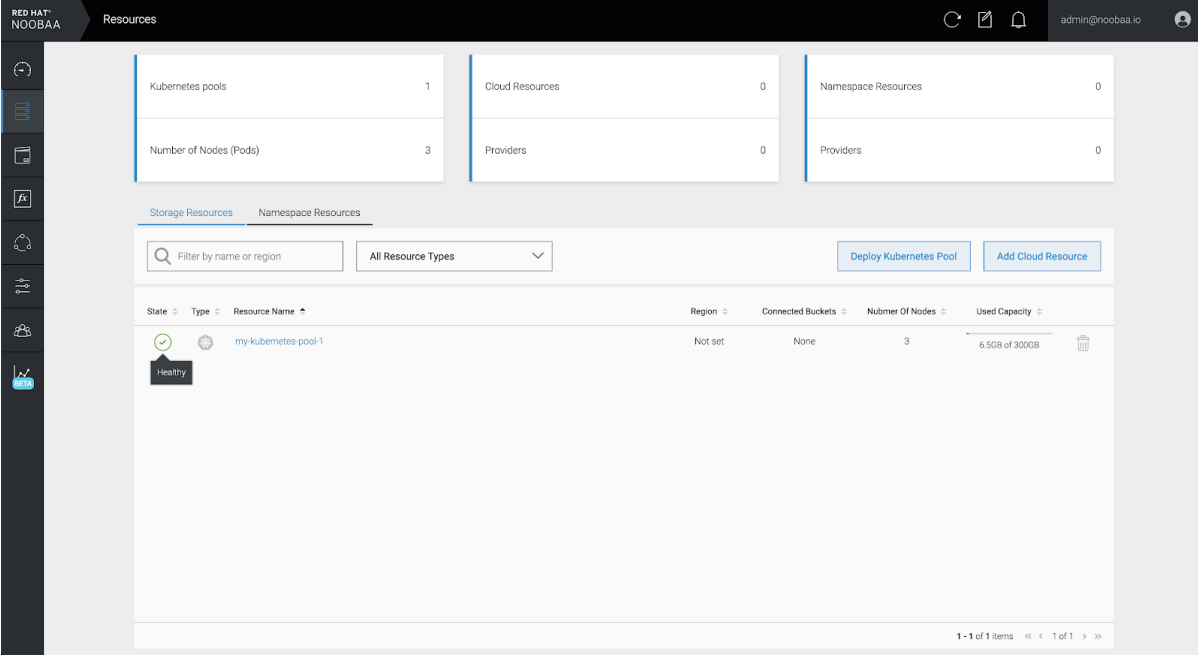
4. **Configure**手順で、要求される Pod 数と各 PV のサイズを設定します。新規 Pod ごとに、1つの PV が作成されます。

The screenshot shows the 'Deploy Kubernetes Pool' dialog box with the 'Configure' step selected. The progress indicator shows: 1. Create Pool (completed), 2. Configure (active), and 3. Review. A warning message states: '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.' Below this are two input fields: 'Number of Nodes (pods):' with a value of 3 and a spinner icon, and 'Node PV Size:' with a value of 100 and a dropdown menu set to GB. A note below the PV size field says: 'This cannot be changed later on'. At the bottom, there is a note: 'For each new node one PV will be created'. There are 'Previous' and 'Next' buttons.

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。

- 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 page. At the top, there are three summary cards: 'Kubernetes pools' (1), 'Cloud Resources' (0), and 'Namespace Resources' (0). Below these, a table shows 'Number of Nodes (Pods)' as 3 and 'Providers' as 0. The main section is titled 'Storage Resources' and includes a search bar, a dropdown for 'All Resource Types', and buttons for 'Deploy Kubernetes Pool' and 'Add Cloud Resource'. A table below lists resources with columns for State, Type, Resource Name, Region, Connected Buckets, Number Of Nodes, and Used Capacity. One resource, 'my-kubernetes-pool-1', is shown with a 'Healthy' status, 'Not set' region, 'None' buckets, 3 nodes, and 6.5GB of 300GB capacity.

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

第8章 RADOS OBJECT GATEWAY S3 エンドポイントへのアクセス

ユーザーは、RADOS Object Gateway (RGW) エンドポイントに直接アクセスできます。

前提条件

- 実行中の OpenShift Container Storage Platform

手順

1. **oc get service** コマンドを実行して RGW サービス名を取得します。

```
$ oc get service
```

NAME	TYPE
rook-ceph-rgw-ocs-storagecluster-cephobjectstore	ClusterIP

CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
172.30.99.207	<none>	80/TCP	4d15h

2. **oc expose** コマンドを実行して RGW サービスを公開します。

```
$ oc expose svc/<RGW service name> --hostname=<route name>
```

<RGW-service name> を直前の手順の RGW サービス名に置き換えます。

<route name> を RGW サービス用に作成するルートに置き換えます。

以下に例を示します。

```
$ oc expose svc/rook-ceph-rgw-ocs-storagecluster-cephobjectstore --hostname=rook-ceph-rgw-ocs.ocp.host.example.com
```

3. **oc get route** コマンドを実行して **oc expose** が成功し、RGW ルートがあることを確認します。

```
$ oc get route
```

NAME	HOST/PORT	PATH
rook-ceph-rgw-ocs-storagecluster-cephobjectstore	rook-ceph-rgw-ocsocp.host.example.com	

SERVICES	PORT	TERMINATION	WILDCARD
rook-ceph-rgw-ocs-storagecluster-cephobjectstore	http		<none>

検証

- **ENDPOINT** を確認するには、以下のコマンドを実行します。

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

<ENDPOINT> を、上記の手順 3 のコマンドから取得したルートに置き換えます。

以下に例を示します。

```
$ aws s3 --no-verify-ssl --endpoint http://rook-ceph-rgw-ocs.ocp.host.example.com ls
```

注記

デフォルトユーザー **ocs-storagecluster-cephobjectstoreuser** のアクセスキーおよびシークレットを取得するには、以下のコマンドを実行します。

- アクセスキー:

```
$ oc get secret rook-ceph-object-user-ocs-storagecluster-cephobjectstore-ocs-storagecluster-cephobjectstoreuser -o yaml | grep -w "AccessKey:" | head -n1 | awk '{print $2}' | base64 --decode
```

- シークレットキー:

```
$ oc get secret rook-ceph-object-user-ocs-storagecluster-cephobjectstore-ocs-storagecluster-cephobjectstoreuser -o yaml | grep -w "SecretKey:" | head -n1 | awk '{print $2}' | base64 --decode
```

第9章 OPENSIFT CONTAINER STORAGE のストレージノードの置き換え

OpenShift Container Storage 4.3 では、ノード置き換えを、以下のデプロイメントで動作するノードについてプロアクティブに実行し、失敗したノードについてリアクティブに実行することができます。

- Amazon Web Services (AWS)
 - ユーザーによってプロビジョニングされるインフラストラクチャー
 - 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
- VMware
 - ユーザーによってプロビジョニングされるインフラストラクチャー
- ローカルストレージデバイスの場合
 - ベアメタル
 - Amazon EC2 I3
 - VMware
- 外部モードでストレージノードを置き換える場合は、[Red Hat Ceph Storage のドキュメント](#) を参照してください。

9.1. AWS にデプロイされる OPENSIFT CONTAINER STORAGE

9.1.1. ユーザーによって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 AWS ノードの置き換え

以下の手順に従って、AWS のユーザーによって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を置き換えます。

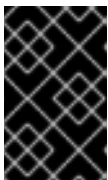
手順

1. 置き換える必要のあるノードを特定します。
2. 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>
```

3. 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。

- 以下のコマンドを使用してノードを削除します。

```
$ oc delete nodes <node_name>
```

- 必要なインフラストラクチャーで新規 AWS マシンインスタンスを作成します。「[プラットフォーム要件](#)」を参照してください。
- 新規 AWS マシンインスタンスを使用して新規 OpenShift Container Platform ノードを作成します。
- Pending** 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

- 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

- 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。
- 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ースを使用する場合

- 新規ノードについて、**Action Menu (!)** → **Edit Labels** をクリックします。
- 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ースの使用

- 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

- 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
```

- 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-***
- 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。
- 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。

9.1.2. 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 AWS ノードの置き換え

以下の手順を使用して、AWS 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。

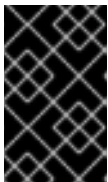
手順

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。

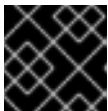
```
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets
```



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ースを使用する場合

- 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ースの使用

- 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. 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](#)。

9.1.3. ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗した AWS ノードの置き換え

以下の手順に従って、OpenShift Container Storage 4.3 の AWS のユーザーによってプロビジョニングされるインフラストラクチャー (UPI) で動作しない障害のあるノードを置き換えます。

手順

1. 置き換える必要のあるノードの AWS マシンインスタンスを特定します。
2. AWS にログインし、特定された AWS マシンインスタンスを終了します。
3. 必要なインフラストラクチャーで新規 AWS マシンインスタンスを作成します。「[プラットフォーム要件](#)」を参照してください。
4. 新規 AWS マシンインスタンスを使用して新規 OpenShift Container Platform ノードを作成します。
5. **Pending** 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

6. 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

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ースを使用する場合

- 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ースの使用

- 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

- 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
```

- 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-***
- 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。
- 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](#)。

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

以下の手順に従って、OpenShift Container Storage の AWS 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

- OpenShift Web コンソールにログインし、**Compute** → **Nodes** をクリックします。
- 障害のあるノードを特定し、その **Machine Name** をクリックします。
- Actions** → **Edit Annotations** をクリックし、**Add More** をクリックします。
- machine.openshift.io/exclude-node-draining** を追加し、**Save** をクリックします。
- Actions** → **Delete Machine** をクリックしてから、**Delete** をクリックします。
- 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ースを使用する場合

- 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ースの使用

- 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. 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
```

10. **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-***
11. 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。
12. 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。

9.2. VMWARE にデプロイされる OPENSIFT CONTAINER STORAGE

9.2.1. ユーザーによって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 VMware ノードの置き換え

以下の手順に従って、VMware のユーザーによってプロビジョニングされるインフラストラクチャー (UPI) で動作するノードを置き換えます。

手順

1. 置き換える必要があるノードとその仮想マシンを特定します。
2. 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>
```

3. 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
```

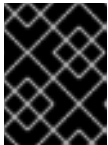
**重要**

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。

- 以下のコマンドを使用してノードを削除します。

```
$ oc delete nodes <node_name>
```

- VSphere にログインし、特定された仮想マシンを終了します。

**重要**

仮想マシンはインベントリからのみ削除し、ディスクから削除しないでください。

- 必要なインフラストラクチャーで vSphere に新規の仮想マシンを作成します。「[プラットフォーム要件](#)」を参照してください。
- 新規の仮想マシンを使用して新規 OpenShift Container Platform ワーカーノードを作成します。
- Pending** 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

- 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

- 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。
- 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ースを使用する場合

- 新規ノードについて、**Action Menu (⋮)** → **Edit Labels** をクリックします。
- 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ースの使用

- 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

- 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. 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。

9.2.2. ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗した VMware ノードの置き換え

以下の手順に従って、VMware のユーザーによってプロビジョニングされるインフラストラクチャー (UPI) で失敗したノードを置き換えます。

手順

1. 置き換える必要があるノードとその仮想マシンを特定します。
2. 以下のコマンドを使用してノードを削除します。

```
$ oc delete nodes <node_name>
```

3. vSphere にログインし、特定された仮想マシンを終了します。



重要

仮想マシンはインベントリからのみ削除し、ディスクから削除しないでください。

4. 必要なインフラストラクチャーで vSphere に新規の仮想マシンを作成します。「[プラットフォーム要件](#)」を参照してください。
5. 新規の仮想マシンを使用して新規 OpenShift Container Platform ワーカーノードを作成します。
6. **Pending** 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

7. 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

8. **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。
9. 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ースを使用する場合

- 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ースの使用

- 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. 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](#)。

9.3. ローカルストレージデバイスを使用した OPENSIFT CONTAINER STORAGE のデプロイ

9.3.1. ベアメタルインフラストラクチャーでのストレージノードの置き換え

- 稼働中のノードを置き換えるには、[こちら](#)を参照してください。「ユーザーによって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」
- 障害のあるノードを置き換えるには、[こちら](#)を参照してください。「ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗したノードの置き換え」

9.3.1.1. ユーザーによって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

前提条件

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

手順

1. ノードを特定し、置き換えるノードのラベルを取得します。ラックラベルをメモします。

```
$ oc get nodes --show-labels | grep <node_name>
```

- 置き換えるノードで実行されている mon (ある場合) およびオブジェクトストレージデバイス (OSD) Pod を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

- 先の手順で特定された Pod のデプロイメントをスケールダウンします。以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

- 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>
```

- ノードをドレイン (解放) します。

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

- ノードを削除します。

```
$ oc delete node <node_name>
```

- 必要なインフラストラクチャーで新規のベアメタルマシンを取得します。[クラスターのベアメタルへのインストール](#)について参照してください。

- 新規ベアメタルマシンを使用して新規 OpenShift Container Platform ノードを作成します。

- Pending** 状態の OpenShift Container Storage に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

- 新規ノードに必要なすべての OpenShift Container Storage CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

- OpenShift Web コンソールで **Compute** → **Nodes** をクリックし、新規ノードが **Ready** 状態にあるかどうかを確認します。

- 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ースを使用する場合

- 新規ノードについて、**Action Menu** (⋮) → **Edit Labels** をクリックします。
- 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ースの使用

- 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=""
```

- これらのワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。

- 新規ディスクエントリーを **LocalVolume** CR に追加します。

LocalVolume CR を編集し、障害のある **デバイス /dev/disk/by-id/{id}** を削除またはコメントアウトし、新規の **/dev/disk/by-id/{id}** を追加します。この例では、新しいデバイスは **/dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB89THF49128A** です。

```
# oc get -n local-storage localvolume
NAME      AGE
local-block 25h
```

```
# oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
  - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY81260978128A
  - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY80440W5U128A
  - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB85AABDE128A
  - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB89THF49128A
storageClassName: localblock
volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

- localblock** と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```
local-pv-3e8964d3          931Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-2-0-79j94 localblock
local-pv-414755e0          931Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-1-0-959rp localblock
local-pv-b481410          931Gi  RWO      Delete    Available
localblock                 3m24s
local-pv-d9c5cbd6          931Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-0-0-nvs68 localblock
```

- 障害のあるノードに関連付けられた PV を削除します。

- 置き換える OSD に関連付けられた **DeviceSet** を特定します。

```
# osd_id_to_remove=0
# oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

ここで、**osd_id_to_remove** は **rook-ceph-osd** プレフィックスの直後にくる 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:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

- b. PVC に関連付けられた PV を特定します。

```
# oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

NAME	STATUS	VOLUME	CAPACITY	ACCESS	MODES
STORAGECLASS	AGE				
ocs-deviceset-0-0-nvs68	Bound	local-pv-d9c5cbd6	931Gi	RWO	localblock
	24h				

この例では、関連付けられた PV は **local-pv-d9c5cbd6** です。

- c. PVC を削除します。

```
# oc delete pvc <pvc-name> -n openshift-storage
```

- d. PV を削除します。

```
# oc delete pv local-pv-d9c5cbd6
```

出力例:

```
persistentvolume "local-pv-d9c5cbd6" deleted
```

15. 失敗した OSD をクラスターから削除します。

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

16. **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
```

17. OSD Pod デプロイメントおよび crashcollector Pod デプロイメントを削除します。

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=<old_node_name> -n openshift-storage
```

18. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS  RESTARTS  AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running  0         1d20h
```

- a. **rook-ceph-operator** を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS  RESTARTS  AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running  0         66s
```

新規 OSD および **mon** の作成には、Operator が再起動するまでに数分かかる場合があります。

19. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

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。

新たな増分の **mon** が作成され、**Running** 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-c-64556f7659-c2ngc          1/1   Running   0      6h14m
rook-ceph-mon-d-7c8b74dc4d-tt6hd         1/1   Running   0      4h24m
rook-ceph-mon-e-57fb8c657-wg5f2         1/1   Running   0      162m
```

OSD と Mon が **Running** 状態になるまで数分かかる場合があります。

4. 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](#)。

9.3.1.2. ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗したノードの置き換え

前提条件

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

手順

1. ノードを特定し、置き換えるノードのラベルを取得します。ラックラベルをメモします。

```
$ oc get nodes --show-labels | grep <node_name>
```

2. 置き換えるノードで実行されている mon (ある場合) およびオブジェクトストレージデバイス (OSD) Pod を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

3. 先の手順で特定された Pod のデプロイメントをスケールダウンします。以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
```

```
┆ --replicas=0 -n openshift-storage
```

4. ノードにスケジューリング対象外 (unschedulable) のマークを付けます。

```
┆ $ oc adm cordon <node_name>
```

5. Terminating 状態の Pod の削除

```
┆ $ oc get pods -A -o wide | grep -i <node_name> | awk '{if ($4 == "Terminating") system ("oc -
┆ n " $1 " delete pods " $2 " --grace-period=0 " " --force ")}'
```

6. ノードをドレイン (解放) します。

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

7. ノードを削除します。

```
┆ $ oc delete node <node_name>
```

8. 必要なインフラストラクチャーで新規のベアメタルマシンを取得します。『クラスタのベアメタルへのインストール』を参照してください。

9. 新規ベアメタルマシンを使用して新規 OpenShift Container Platform ノードを作成します。

10. **Pending** 状態の OpenShift Container Storage に関連する証明書署名要求 (CSR) の有無を確認します。

```
┆ $ oc get csr
```

11. 新規ノードに必要なすべての OpenShift Container Storage CSR を承認します。

```
┆ $ oc adm certificate approve <Certificate_Name>
```

12. OpenShift Web コンソールで **Compute → Nodes** をクリックし、新規ノードが **Ready** 状態にあるかどうかを確認します。

13. 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ースを使用する場合

- 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ースの使用

- 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=""
```

14. これらのワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。

- a. 新規ディスクエントリーを **LocalVolume** CR に追加します。

LocalVolume CR を編集し、障害のある **デバイス /dev/disk/by-id/{id}** を削除またはコメントアウトし、新規の **/dev/disk/by-id/{id}** を追加します。この例では、新しいデバイスは **/dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB89THF49128A** です。

```
# oc get -n local-storage localvolume
NAME      AGE
local-block 25h
```

```
# oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
- /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY81260978128A
- /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY80440W5U128A
- /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB85AABDE128A
- /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB89THF49128A
storageClassName: localblock
volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

- b. **localblock** と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```
local-pv-3e8964d3          931Gi   RWO          Delete       Bound
openshift-storage/ocs-deviceset-2-0-79j94 localblock 25h
local-pv-414755e0          931Gi   RWO          Delete       Bound
openshift-storage/ocs-deviceset-1-0-959rp localblock 25h
local-pv-b481410          931Gi   RWO          Delete       Available
localblock                 3m24s
local-pv-d9c5cbd6          931Gi   RWO          Delete       Bound
openshift-storage/ocs-deviceset-0-0-nvs68 localblock
```

15. 障害のあるノードに関連付けられた PV を削除します。

- a. 置き換える OSD に関連付けられた **DeviceSet** を特定します。

```
# osd_id_to_remove=0
# oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

ここで、**osd_id_to_remove** は **rook-ceph-osd** プレフィックスの直後にくる 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:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

- b. PVC に関連付けられた PV を特定します。

```
# oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
NAME                STATUS      VOLUME          CAPACITY  ACCESS MODES
STORAGECLASS  AGE
ocs-deviceset-0-0-nvs68  Bound  local-pv-d9c5cbd6  931Gi    RWO          localblock
24h
```

この例では、関連付けられた PV は **local-pv-d9c5cbd6** です。

- c. PVC を削除します。

```
# oc delete pvc <pvc-name> -n openshift-storage
```

- d. PV を削除します。

```
# oc delete pv local-pv-d9c5cbd6
```

出力例:

```
persistentvolume "local-pv-d9c5cbd6" deleted
```

16. 失敗した OSD をクラスターから削除します。

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

17. **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
```

18. OSD Pod デプロイメントおよび crashcollector Pod デプロイメントを削除します。

■

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=
<old_node_name> -n openshift-storage
```

19. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running 0      1d20h
```

- a. **rook-ceph-operator** を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running 0      66s
```

新規 OSD および **mon** の作成には、Operator が再起動するまでに数分かかる場合があります。

20. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

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。
新たな増分の **mon** が作成され、**Running** 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-c-64556f7659-c2ngc          1/1   Running   0        6h14m
rook-ceph-mon-d-7c8b74dc4d-tt6hd         1/1   Running   0        4h24m
rook-ceph-mon-e-57fb8c657-wg5f2         1/1   Running   0        162m
```

OSD と Mon が **Running** 状態になるまで数分かかる場合があります。

4. 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。

9.3.2. Amazon EC2 インフラストラクチャーでのストレージノードの置き換え

- ユーザーによってプロビジョニングされるインフラストラクチャーおよび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 Amazon EC2 ノードを置き換えるには、以下を参照してください。
 - [「ユーザーによって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 Amazon EC2 ノードの置き換え」](#)
 - [「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 Amazon EC2 ノードの置き換え」](#)
- ユーザーによってプロビジョニングされるインフラストラクチャーおよび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障害のある Amazon EC2 ノードを置き換えるには、以下を参照してください。
 - [「ユーザーによって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 Amazon EC2 ノードの置き換え」](#)
 - [「インストーラーでプロビジョニングされるインフラストラクチャーでの失敗した AWS ノードの置き換え」](#)

9.3.2.1. ユーザーによって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 Amazon EC2 ノードの置き換え

以下の手順に従って、Amazon EC2 I3 のユーザーによってプロビジョニングされるインフラストラクチャー (UPI) で動作するノードを置き換えます。



重要

Amazon EC2 I3 インフラストラクチャーのストレージノードの置き換えはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

前提条件

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

手順

1. ノードを特定し、置き換えるノードのラベルを取得します。

```
$ oc get nodes --show-labels | grep <node_name>
```

2. 置き換えるノードで実行されている mon (ある場合) および OSD を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

3. 先の手順で特定された Pod のデプロイメントをスケールダウンします。
以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage  
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage  
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>  
--replicas=0 -n openshift-storage
```

4. ノードにスケジューラ対象外 (unschedulable) のマークを付けます。

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

5. ノードをドレイン (解放) します。

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

6. ノードを削除します。

```
$ oc delete node <node_name>
```

7. 必要なインフラストラクチャーで新規 Amazon EC2 I3 マシンインスタンスを作成します。[サポートされるインフラストラクチャーおよびプラットフォーム](#)について参照してください。

8. 新規 Amazon EC2 I3 マシンインスタンスを使用して新規 OpenShift Container Platform ノードを作成します。

9. Pending 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

10. 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

11. OpenShift Web コンソールで **Compute** → **Nodes** をクリックします。新規ノードが **Ready** 状態にあるかどうかを確認します。

12. 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ースを使用する場合

- 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ースの使用

- 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=""
```

13. 新規ワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。

- a. 新規ディスクエントリを LocalVolume CR に追加します。

LocalVolume CR を編集します。障害のあるデバイス `/dev/disk/by-id/{id}` を削除またはコメントアウトし、新規の `/dev/disk/by-id/{id}` を追加します。

```
$ oc get -n local-storage localvolume
```

出力例:

```
NAME      AGE
local-block 25h
```

```
$ oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441494EC
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84FE3E9
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE4
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441464EP
  # - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84F43E7
  # - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE8
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4
  storageClassName: localblock
  volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

この CR に `by-id` を使用する 2 つの新規デバイスが追加されていることを確認できます。

- `nvme-Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9`
- `nvme-Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4`

b. `localblock` と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```
local-pv-3646185e 2328Gi RWO Delete Available
localblock 9s
local-pv-3933e86 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-1-v9jp4 localblock 5h1m
local-pv-8176b2bf 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-0-nvs68 localblock 5h1m
local-pv-ab7cabb3 2328Gi RWO Delete Available
localblock 9s
local-pv-ac52e8a 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-0-knrgr localblock 5h1m
local-pv-b7e6fd37 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-0-rdm7m localblock 5h1m
local-pv-cb454338 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-1-h9hfm localblock 5h1m
local-pv-da5e3175 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-1-g97lq localblock 5h
...
```

14. 以下の手順で、失敗したノードに関連付けられた各 PV および OSD を削除します。

a. 置き換える OSD に関連付けられた DeviceSet を特定します。

```
$ osd_id_to_remove=0
$ oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

ここで、`osd_id_to_remove` は `rook-ceph-osd` プレフィックスの直後にくる 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:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

b. PVC に関連付けられた PV を特定します。

```
$ oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、`x`、`y`、および `pvc-suffix` は、直前の手順で特定された DeviceSet の値です。

出力例:

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
ocs-deviceset-0-0-nvs68	Bound	local-pv-8176b2bf	2328Gi	RWO
4h49m				localblock

この例では、関連付けられた PV は **local-pv-8176b2bf** です。

- c. 先の手順で特定された PVC を削除します。この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

```
$ oc delete pvc ocs-deviceset-0-0-nvs68 -n openshift-storage
```

出力例:

```
persistentvolumeclaim "ocs-deviceset-0-0-nvs68" deleted
```

- d. 先のステップで特定された PV を削除します。この例では、PV 名は **local-pv-8176b2bf** です。

```
$ oc delete pv local-pv-8176b2bf
```

出力例:

```
persistentvolume "local-pv-8176b2bf" deleted
```

- e. 失敗した OSD をクラスターから削除します。

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

- f. **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
```

- g. OSD Pod デプロイメントを削除します。

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
```

15. 先の手順で特定された **crashcollector** Pod デプロイメントを削除します。

```
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=
<old_node_name> -n openshift-storage
```

16. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running 0      5h3m
```

- a. **rook-ceph-operator** を削除します。

```
$ oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running 0      66s
```

新規 OSD の作成には、Operator が起動するまでに数分かかる場合があります。

17. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

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。
また、増分の **mon** が新規に作成されており、**Running** 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-a-64556f7659-c2ngc 1/1 Running 0 5h1m
rook-ceph-mon-b-7c8b74dc4d-tt6hd 1/1 Running 0 5h1m
rook-ceph-mon-d-57fb8c657-wg5f2 1/1 Running 0 27m
```

OSD と **mon** が **Running** 状態になるまで数分かかる場合があります。

4. 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。

9.3.2.2. 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 Amazon EC2 ノードの置き換え

以下の手順を使用して、Amazon EC2 I3 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。



重要

Amazon EC2 I3 インフラストラクチャーのストレージノードの置き換えはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

前提条件

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

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. 置き換えるノードのラベルを取得します。

```
$ oc get nodes --show-labels | grep <node_name>
```

4. 置き換えるノードで実行されている **mon** (ある場合) および **OSD** を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

5. 先の手順で特定された Pod のデプロイメントをスケールダウンします。
以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

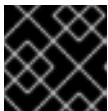
6. ノードにスケジューリング対象外 (unschedulable) のマークを付けます。

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

7. ノードをドレイン (解放) します。

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

8. **Compute** → **Machines** をクリックします。必要なマシンを検索します。
9. 必要なマシンの横にある **Action menu (!)** → **Delete Machine** をクリックします。
10. **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。
11. 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。

12. OpenShift Web コンソールで **Compute** → **Nodes** をクリックします。新規ノードが **Ready** 状態にあるかどうかを確認します。
13. 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ースを使用する場合

- 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ースの使用

- 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=""
```

14. 新規ワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。
 - a. 新規ディスクエントリーを LocalVolume CR に追加します。
LocalVolume CR を編集します。障害のあるデバイス **/dev/disk/by-id/{id}** を削除またはコメントアウトし、新規の **/dev/disk/by-id/{id}** を追加します。

```
$ oc get -n local-storage localvolume
```

出力例:


```
NAME      AGE
local-block 25h
```

```
$ oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
  storageClassDevices:
  - devicePaths:
    - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441494EC
    - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84FE3E9
    - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE4
    - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441464EP
    # - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84F43E7
    # - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE8
    - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9
    - /dev/disk/by-id/nvme-
      Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4
    storageClassName: localblock
    volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

この CR に by-id を使用する 2 つの新規デバイスが追加されていることを確認できます。

- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9**
- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4**

b. **localblock** と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```
local-pv-3646185e 2328Gi RWO Delete Available
localblock 9s
local-pv-3933e86 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-1-v9jp4 localblock 5h1m
local-pv-8176b2bf 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-0-nvs68 localblock 5h1m
local-pv-ab7cabb3 2328Gi RWO Delete Available
localblock 9s
local-pv-ac52e8a 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-0-knrgr localblock 5h1m
```

```

local-pv-b7e6fd37 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-0-rdm7m localblock 5h1m
local-pv-cb454338 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-1-h9hfm localblock 5h1m
local-pv-da5e3175 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-1-g97lq localblock 5h
...

```

15. 以下の手順で、失敗したノードに関連付けられた各 PV および OSD を削除します。

- a. 置き換える OSD に関連付けられた DeviceSet を特定します。

```

$ osd_id_to_remove=0
$ oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc

```

ここで、**osd_id_to_remove** は **rook-ceph-osd** プレフィックスの直後にくる 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:

```

ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68

```

- b. PVC に関連付けられた PV を特定します。

```

$ oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>

```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された DeviceSet の値です。

出力例:

```

NAME                STATUS    VOLUME          CAPACITY  ACCESS MODES
STORAGECLASS  AGE
ocs-deviceset-0-0-nvs68  Bound    local-pv-8176b2bf 2328Gi   RWO          localblock
4h49m

```

この例では、関連付けられた PV は **local-pv-8176b2bf** です。

- c. 先の手順で特定された PVC を削除します。この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

```

$ oc delete pvc ocs-deviceset-0-0-nvs68 -n openshift-storage

```

出力例:

```

persistentvolumeclaim "ocs-deviceset-0-0-nvs68" deleted

```

- d. 先のステップで特定された PV を削除します。この例では、PV 名は **local-pv-8176b2bf** です。

```

$ oc delete pv local-pv-8176b2bf

```

出力例:

```
persistentvolume "local-pv-8176b2bf" deleted
```

- e. 失敗した OSD をクラスターから削除します。

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

- f. **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
```

- g. OSD Pod デプロイメントを削除します。

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
```

16. 先の手順で特定された **crashcollector** Pod デプロイメントを削除します。

```
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=
<old_node_name> -n openshift-storage
```

17. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running 0      5h3m
```

- a. **rook-ceph-operator** を削除します。

```
$ oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running 0      66s
```

新規 OSD の作成には、Operator が起動するまでに数分かかる場合があります。

18. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

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。
また、増分の **mon** が新規に作成されており、**Running** 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-a-64556f7659-c2ngc 1/1   Running 0   5h1m
rook-ceph-mon-b-7c8b74dc4d-tt6hd 1/1   Running 0   5h1m
rook-ceph-mon-d-57fb8c657-wg5f2 1/1   Running 0   27m
```

OSD と mon が **Running** 状態になるまで数分かかる場合があります。

4. 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](#)。

9.3.2.3. ユーザーによって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 Amazon EC2 ノードの置き換え

OpenShift Container Storage の Amazon EC2 I3 の一時ストレージにより、インスタンスの電源がオフにされる場合にデータが失われる可能性があります。以下の手順を使用して、Amazon EC2 インフラストラクチャーでのインスタンスの電源オフからのリカバリーを行います。



重要

Amazon EC2 I3 インフラストラクチャーのストレージノードの置き換えはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

前提条件

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

手順

1. ノードを特定し、置き換えるノードのラベルを取得します。

```
$ oc get nodes --show-labels | grep <node_name>
```

2. 置き換えるノードで実行されている mon (ある場合) および OSD を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

3. 先の手順で特定された Pod のデプロイメントをスケールダウンします。以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

4. ノードにスケジューラ対象外 (unschedulable) のマークを付けます。

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

5. Terminating 状態の Pod を削除します。

```
$ oc get pods -A -o wide | grep -i <node_name> | awk '{if ($4 == "Terminating") system ("oc -n " $1 " delete pods " $2 " --grace-period=0 " " --force ")}'
```

6. ノードをドレイン (解放) します。

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

7. ノードを削除します。

```
$ oc delete node <node_name>
```

8. 必要なインフラストラクチャーで新規 Amazon EC2 I3 マシンインスタンスを作成します。サポートされるインフラストラクチャーおよびプラットフォームについて参照してください。

9. 新規 Amazon EC2 I3 マシンインスタンスを使用して新規 OpenShift Container Platform ノードを作成します。
10. Pending 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

11. 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

12. OpenShift Web コンソールで **Compute** → **Nodes** をクリックします。新規ノードが **Ready** 状態にあるかどうかを確認します。
13. 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ースを使用する場合

- 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ースの使用

- 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=""
```

14. 新規ワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。
 - a. 新規ディスクエントリを LocalVolume CR に追加します。
LocalVolume CR を編集します。障害のあるデバイス **/dev/disk/by-id/{id}** を削除またはコメントアウトし、新規の **/dev/disk/by-id/{id}** を追加します。

```
$ oc get -n local-storage localvolume
```

出力例:

```
NAME      AGE
local-block 25h
```

```
$ oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441494EC
```

```

- /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84FE3E9
- /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE4
- /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441464EP
# - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84F43E7
# - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE8
- /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9
- /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4
storageClassName: localblock
volumeMode: Block
[...]

```

CR の編集後に変更を保存するようにしてください。

この CR に `by-id` を使用する 2 つの新規デバイスが追加されていることを確認できます。

- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9**
- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4**

b. **localblock** と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```

local-pv-3646185e 2328Gi RWO Delete Available
localblock 9s
local-pv-3933e86 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-1-v9jp4 localblock 5h1m
local-pv-8176b2bf 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-0-nvs68 localblock 5h1m
local-pv-ab7cabb3 2328Gi RWO Delete Available
localblock 9s
local-pv-ac52e8a 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-0-knrgr localblock 5h1m
local-pv-b7e6fd37 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-0-rdm7m localblock 5h1m
local-pv-cb454338 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-1-h9hfm localblock 5h1m
local-pv-da5e3175 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-1-g97lq localblock 5h
...

```

15. 以下の手順で、失敗したノードに関連付けられた各 PV および OSD を削除します。

a. 置き換える OSD に関連付けられた DeviceSet を特定します。

```
$ osd_id_to_remove=0
$ oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

ここで、**osd_id_to_remove** は **rook-ceph-osd** プレフィックスの直後にくる 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:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

- b. PVC に関連付けられた PV を特定します。

```
$ oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された DeviceSet の値です。

出力例:

NAME	STATUS	VOLUME	CAPACITY	ACCESS	MODES
STORAGECLASS	AGE				
ocs-deviceset-0-0-nvs68	Bound	local-pv-8176b2bf	2328Gi	RWO	localblock
	4h49m				

この例では、関連付けられた PV は **local-pv-8176b2bf** です。

- c. 先の手順で特定された PVC を削除します。この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

```
$ oc delete pvc ocs-deviceset-0-0-nvs68 -n openshift-storage
```

出力例:

```
persistentvolumeclaim "ocs-deviceset-0-0-nvs68" deleted
```

- d. 先のステップで特定された PV を削除します。この例では、PV 名は **local-pv-8176b2bf** です。

```
$ oc delete pv local-pv-8176b2bf
```

出力例:

```
persistentvolume "local-pv-8176b2bf" deleted
```

- e. 失敗した OSD をクラスターから削除します。

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

- f. `oc get pod removal` Pod のフィニッシュ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

- f. **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
```

- g. OSD Pod デプロイメントを削除します。

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
```

16. 先の手順で特定された **crashcollector** Pod デプロイメントを削除します。

```
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=<old_node_name> -n openshift-storage
```

17. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running 0      5h3m
```

- a. **rook-ceph-operator** を削除します。

```
$ oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running 0      66s
```

新規 OSD の作成には、Operator が起動するまでに数分かかる場合があります。

18. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

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。
また、増分の **mon** が新規に作成されており、**Running** 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-a-64556f7659-c2ngc 1/1 Running 0 5h1m
rook-ceph-mon-b-7c8b74dc4d-tt6hd 1/1 Running 0 5h1m
rook-ceph-mon-d-57fb8c657-wg5f2 1/1 Running 0 27m
```

OSD と mon が **Running** 状態になるまで数分かかる場合があります。

4. 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](#)。

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

OpenShift Container Storage の Amazon EC2 I3 の一時ストレージにより、インスタンスの電源がオフにされる場合にデータが失われる可能性があります。以下の手順を使用して、Amazon EC2 インフラストラクチャーでのインスタンスの電源オフからのリカバリーを行います。

重要

Amazon EC2 I3 インフラストラクチャーのストレージノードの置き換えはテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。Red Hat は実稼働環境でこれらを使用することを推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

前提条件

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

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. 置き換えるノードのラベルを取得します。

```
$ oc get nodes --show-labels | grep <node_name>
```

4. 置き換えるノードで実行されている mon (ある場合) および OSD を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

5. 先の手順で特定された Pod のデプロイメントをスケールダウンします。以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

6. 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>
```

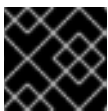
7. Terminating 状態の Pod を削除します。

```
$ oc get pods -A -o wide | grep -i <node_name> | awk '{if ($4 == "Terminating") system ("oc -
n " $1 " delete pods " $2 " --grace-period=0 " " --force ")}'
```

8. ノードをドレイン (解放) します。

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

9. **Compute** → **Machines** をクリックします。必要なマシンを検索します。
10. 必要なマシンの横にある **Action menu** (⋮) → **Delete Machine** をクリックします。
11. **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。
12. 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には少なくとも 5-10 分以上かかる場合があります。

13. OpenShift Web コンソールで **Compute** → **Nodes** をクリックします。新規ノードが **Ready** 状態にあるかどうかを確認します。

14. 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ースを使用する場合

- 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ースの使用

- 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=""
```

15. 新規ワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。

- a. 新規ディスクエントリを LocalVolume CR に追加します。

LocalVolume CR を編集します。障害のあるデバイス **/dev/disk/by-id/{id}** を削除またはコメントアウトし、新規の **/dev/disk/by-id/{id}** を追加します。

```
$ oc get -n local-storage localvolume
```

出力例:

```
NAME      AGE
local-block 25h
```

```
$ oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441494EC
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84FE3E9
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE4
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441464EP
  # - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84F43E7
  # - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE8
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9
  - /dev/disk/by-id/nvme-
    Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4
```

```
storageClassName: localblock
volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

この CR に `by-id` を使用する 2 つの新規デバイスが追加されていることを確認できます。

- `nvme-Amazon_EC2_NVMe_Instance_Storage_AWS6F45C01D7E84FE3E9`
- `nvme-Amazon_EC2_NVMe_Instance_Storage_AWS636BC945B4ECB9AE4`

b. `localblock` と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```
local-pv-3646185e 2328Gi RWO Delete Available
localblock 9s
local-pv-3933e86 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-1-v9jp4 localblock 5h1m
local-pv-8176b2bf 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-0-nvs68 localblock 5h1m
local-pv-ab7cabb3 2328Gi RWO Delete Available
localblock 9s
local-pv-ac52e8a 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-0-knrgr localblock 5h1m
local-pv-b7e6fd37 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-2-0-rdm7m localblock 5h1m
local-pv-cb454338 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-0-1-h9hfm localblock 5h1m
local-pv-da5e3175 2328Gi RWO Delete Bound openshift-storage/ocs-
deviceset-1-1-g97lq localblock 5h
...
```

16. 以下の手順で、失敗したノードに関連付けられた各 PV および OSD を削除します。

a. 置き換える OSD に関連付けられた DeviceSet を特定します。

```
$ osd_id_to_remove=0
$ oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

ここで、`osd_id_to_remove` は `rook-ceph-osd` プレフィックスの直後にくる 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:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

b. PVC に関連付けられた PV を特定します。

```
$ oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

-

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された DeviceSet の値です。

出力例:

```
NAME                STATUS    VOLUME          CAPACITY  ACCESS MODES
STORAGECLASS  AGE
ocs-deviceset-0-0-nvs68  Bound    local-pv-8176b2bf  2328Gi   RWO          localblock
4h49m
```

この例では、関連付けられた PV は **local-pv-8176b2bf** です。

- c. 先の手順で特定された PVC を削除します。この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

```
$ oc delete pvc ocs-deviceset-0-0-nvs68 -n openshift-storage
```

出力例:

```
persistentvolumeclaim "ocs-deviceset-0-0-nvs68" deleted
```

- d. 先のステップで特定された PV を削除します。この例では、PV 名は **local-pv-8176b2bf** です。

```
$ oc delete pv local-pv-8176b2bf
```

出力例:

```
persistentvolume "local-pv-8176b2bf" deleted
```

- e. 失敗した OSD をクラスターから削除します。

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

- f. **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
```

- g. OSD Pod デプロイメントを削除します。

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
```

17. 先の手順で特定された **crashcollector** Pod デプロイメントを削除します。

```
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=<old_node_name> -n openshift-storage
```

18. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS  RESTARTS  AGE
rook-ceph-operator-6f74fb5bff-2d982  1/1   Running  0         5h3m
```

- a. **rook-ceph-operator** を削除します。

```
$ oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
$ oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS  RESTARTS  AGE
rook-ceph-operator-6f74fb5bff-7mvrq  1/1   Running  0         66s
```

新規 OSD の作成には、Operator が起動するまでに数分かかる場合があります。

19. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

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。また、増分の **mon** が新規に作成されており、**Running** 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-a-64556f7659-c2ngc 1/1 Running 0 5h1m
rook-ceph-mon-b-7c8b74dc4d-tt6hd 1/1 Running 0 5h1m
rook-ceph-mon-d-57fb8c657-wg5f2 1/1 Running 0 27m
```

OSD と **mon** が **Running** 状態になるまで数分かかる場合があります。

4. 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。

9.3.3. VMWare インフラストラクチャーでのストレージノードの置き換え

- 稼働中のノードを置き換えるには、[を参照してください。](#) 「VMware のユーザーによって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」
- 障害のあるノードを置き換えるには、[を参照してください。](#) 「VMware ユーザーによって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」

9.3.3.1. VMware のユーザーによって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

前提条件

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

手順

1. ノードを特定し、置き換えるノードのラベルを取得します。

```
$ oc get nodes --show-labels | grep <node_name>
```

2. 置き換えるノードで実行されている **mon** (ある場合) および OSD を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

3. 先の手順で特定された Pod のデプロイメントをスケールダウンします。以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
```



```
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

4. 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>
```

5. ノードをドレイン (解放) します。

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

6. ノードを削除します。

```
$ oc delete node <node_name>
```

7. VSphere にログインし、特定された仮想マシンを終了します。

8. 必要なインフラストラクチャーで VMware に新規の仮想マシンを作成します。 [サポートされるインフラストラクチャーおよびプラットフォーム](#) について参照してください。

9. 新規の仮想マシンを使用して新規 OpenShift Container Platform ワーカーノードを作成します。

10. Pending 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

11. 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

12. OpenShift Web コンソールで **Compute → Nodes** をクリックし、新規ノードが **Ready** 状態にあるかどうかを確認します。

13. 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ースを使用する場合

- 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ースの使用

- 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=""
```

14. これらのワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。

- a. 新規ディスクエントリーを **LocalVolume** CR に追加します。

LocalVolume CR を編集し、障害のあるデバイス `/dev/disk/by-id/{id}` を削除またはコメントアウトし、新規の `/dev/disk/by-id/{id}` を追加します。この例では、新規デバイスは `/dev/disk/by-id/nvme-eui.01000000010000005cd2e490020e5251` です。

```
# oc get -n local-storage localvolume
```

出力例:

```
NAME      AGE
local-block 25h
```

```
# oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4895e0e5251
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4ea2f0f5251
# - /dev/disk/by-id/nvme-eui.01000000010000005cd2e4de2f0f5251
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e490020e5251
storageClassName: localblock
volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

- b. **localblock** と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```
local-pv-3e8964d3          1490Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-2-0-79j94 localblock 25h
local-pv-414755e0          1490Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-1-0-959rp localblock 25h
local-pv-b481410          1490Gi  RWO      Delete    Available
localblock                 3m24s
local-pv-d9c5cbd6          1490Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-0-0-nvs68 localblock
```

15. 障害のあるノードに関連付けられた PV を削除します。

- a. 置き換える OSD に関連付けられた **DeviceSet** を特定します。

```
# osd_id_to_remove=0
# oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

ここで、`osd_id_to_remove` は `rook-ceph-osd` プレフィックスの直後にくる 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:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

この例では、PVC 名は `OCS-deviceset-0-0-nvs68` です。

- b. PVC に関連付けられた PV を特定します。

```
# oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、`x`、`y`、および `pvc-suffix` は、直前の手順で特定された `DeviceSet` の値です。

出力例:

```
NAME                STATUS    VOLUME          CAPACITY  ACCESS MODES
STORAGECLASS  AGE
ocs-deviceset-0-0-nvs68  Bound    local-pv-d9c5cbd6  1490Gi   RWO          localblock
24h
```

この例では、関連付けられた PV は `local-pv-d9c5cbd6` です。

- c. PVC を削除します。

```
oc delete pvc <pvc-name> -n openshift-storage
```

- d. PV を削除します。

```
# oc delete pv local-pv-d9c5cbd6
```

出力例:

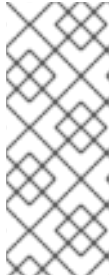
```
persistentvolume "local-pv-d9c5cbd6" deleted
```

16. 失敗した OSD をクラスターから削除します。

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

17. `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
```

18. OSD Pod デプロイメントおよび crashcollector Pod デプロイメントを削除します。

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=<old_node_name> -n openshift-storage
```

19. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running 0      1d20h
```

- a. **rook-ceph-operator** を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running 0      66s
```

新規 OSD および **mon** の作成には、Operator が再起動するまでに数分かかる場合があります。

20. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

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。

また、増分の **mon** が新規に作成されており、Running 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-c-64556f7659-c2ngc          1/1   Running   0      6h14m
rook-ceph-mon-d-7c8b74dc4d-tt6hd         1/1   Running   0      4h24m
rook-ceph-mon-e-57fb8c657-wg5f2         1/1   Running   0      162m
```

OSD と Mon が **Running** 状態になるまで数分かかる場合があります。

4. 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](#)。

9.3.3.2. VMware ユーザーによって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

前提条件

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

手順

1. ノードを特定し、置き換えるノードのラベルを取得します。

```
$ oc get nodes --show-labels | grep <node_name>
```

2. 置き換えるノードで実行されている **mon** (ある場合) および OSD を特定します。

```
$ oc get pods -n openshift-storage -o wide | grep -i <node_name>
```

3. 先の手順で特定された Pod のデプロイメントをスケールダウンします。
以下に例を示します。

```
$ oc scale deployment rook-ceph-mon-c --replicas=0 -n openshift-storage
$ oc scale deployment rook-ceph-osd-0 --replicas=0 -n openshift-storage
$ oc scale deployment --selector=app=rook-ceph-crashcollector,node_name=<node_name>
--replicas=0 -n openshift-storage
```

4. 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>
```

5. Terminating 状態の Pod を削除します。

```
$ oc get pods -A -o wide | grep -i <node_name> | awk '{if ($4 == "Terminating") system ("oc -n "$1 " delete pods "$2 " --grace-period=0 " " --force ")}'
```

6. ノードをドレイン (解放) します。

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

7. ノードを削除します。

```
$ oc delete node <node_name>
```

8. VSphere にログインし、特定された仮想マシンを終了します。

9. 必要なインフラストラクチャーで VMware に新規の仮想マシンを作成します。 [サポートされるインフラストラクチャーおよびプラットフォーム](#) について参照してください。

10. 新規の仮想マシンを使用して新規 OpenShift Container Platform ワーカーノードを作成します。

11. Pending 状態の OpenShift Container Platform に関連する証明書署名要求 (CSR) の有無を確認します。

```
$ oc get csr
```

12. 新規ノードに必要なすべての OpenShift Container Platform CSR を承認します。

```
$ oc adm certificate approve <Certificate_Name>
```

13. OpenShift Web コンソールで **Compute** → **Nodes** をクリックし、新規ノードが **Ready** 状態にあるかどうかを確認します。

14. 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ースを使用する場合

- 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ースの使用

- 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=""
```

15. これらのワーカーノードで利用可能なローカルストレージデバイスを OpenShift Container Storage StorageCluster に追加します。

- a. 新規ディスクエントリを **LocalVolume** CR に追加します。

LocalVolume CR を編集し、障害のあるデバイス `/dev/disk/by-id/{id}` を削除またはコメントアウトし、新規の `/dev/disk/by-id/{id}` を追加します。この例では、新規デバイスは `/dev/disk/by-id/nvme-eui.01000000010000005cd2e490020e5251` です。

```
# oc get -n local-storage localvolume
```

出力例:

```
NAME      AGE
local-block 25h
```

```
# oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4895e0e5251
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4ea2f0f5251
# - /dev/disk/by-id/nvme-eui.01000000010000005cd2e4de2f0f5251
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e490020e5251
storageClassName: localblock
volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

- b. **localblock** と共に PV を表示します。

```
$ oc get pv | grep localblock
```

出力例:

```
local-pv-3e8964d3          1490Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-2-0-79j94 localblock 25h
local-pv-414755e0          1490Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-1-0-959rp localblock 25h
local-pv-b481410          1490Gi  RWO      Delete    Available
localblock                 3m24s
local-pv-d9c5cbd6          1490Gi  RWO      Delete    Bound
openshift-storage/ocs-deviceset-0-0-nvs68 localblock
```

16. 障害のあるノードに関連付けられた PV を削除します。

- a. 置き換える OSD に関連付けられた **DeviceSet** を特定します。

```
# osd_id_to_remove=0
# oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

ここで、**osd_id_to_remove** は **rook-ceph-osd** プレフィックスの直後にくる 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:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

- b. PVC に関連付けられた PV を特定します。

```
# oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
NAME                STATUS      VOLUME          CAPACITY  ACCESS MODES
STORAGECLASS  AGE
ocs-deviceset-0-0-nvs68 Bound local-pv-d9c5cbd6  1490Gi  RWO          localblock
24h
```

この例では、関連付けられた PV は **local-pv-d9c5cbd6** です。

- c. PVC を削除します。

```
oc delete pvc <pvc-name> -n openshift-storage
```

- d. PV を削除します。

```
# oc delete pv local-pv-d9c5cbd6
```

出力例:

```
persistentvolume "local-pv-d9c5cbd6" deleted
```

17. 失敗した OSD をクラスターから削除します。

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

18. **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
```

19. OSD Pod デプロイメントおよび crashcollector Pod デプロイメントを削除します。

```
$ oc delete deployment rook-ceph-osd-${osd_id_to_remove} -n openshift-storage
$ oc delete deployment --selector=app=rook-ceph-crashcollector,node_name=<old_node_name> -n openshift-storage
```

20. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS  RESTARTS  AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running  0         1d20h
```

- a. **rook-ceph-operator** を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

- b. **rook-ceph-operator** Pod が再起動していることを確認します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS  RESTARTS  AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running  0         66s
```

新規 OSD および **mon** の作成には、Operator が再起動するまでに数分かかる場合があります。

21. `ocs-osd-removal` ジョブを削除します。

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

出力例:

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

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。

また、増分の **mon** が新規に作成されており、Running 状態にあることを確認します。

```
$ oc get pod -n openshift-storage | grep mon
```

出力例:

```
rook-ceph-mon-c-64556f7659-c2ngc          1/1   Running   0        6h14m
rook-ceph-mon-d-7c8b74dc4d-tt6hd         1/1   Running   0        4h24m
rook-ceph-mon-e-57fb8c657-wg5f2         1/1   Running   0        162m
```

OSD と Mon が **Running** 状態になるまで数分かかる場合があります。

4. 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10章 ストレージデバイスの置き換え

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ストレージノードを置き換えることができます。

- AWS にデプロイされた動的に作成されたストレージクラスターについては、以下を参照してください。
 - 「AWS のユーザーによってプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え」
 - 「AWS のインストーラーでプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え」
- VMware にデプロイされた動的に作成されたストレージクラスターについては、「VMware のユーザーによってプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え」を参照してください。
- ローカルストレージデバイスを使用してデプロイされたストレージクラスターについては、以下を参照してください。
 - 「Amazon EC2 インフラストラクチャーでの障害のあるストレージノードの置き換え」
 - 「VMware およびベアメタルインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え」

10.1. AWS への OPENSIFT CONTAINER STORAGE の動的プロビジョニング

10.1.1. AWS のユーザーによってプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え

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

- ユーザーによって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 AWS ノードの置き換え
- ユーザーによってプロビジョニングされるインフラストラクチャーでの失敗した AWS ノードの置き換え

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

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

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

10.2. VMWARE への OPENSIFT CONTAINER STORAGE の動的プロビジョニング

10.2.1. VMware のユーザーによってプロビジョニングされるインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え

この手順は、VMware インフラストラクチャーに動的にデプロイされる OpenShift Container Storage で仮想マシンディスク (VMDK) を置き換える必要がある場合に使用します。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. 置き換える必要のある OSC 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 10.131.2.32 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ます。



注記

置き換える OSD が正常であれば、Pod のステータスは Running になります。

2. 置き換えられる OSD の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** プレフィックスの直後にくる 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
```

- 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。

```
# oc process -n openshift-storage ocs-osd-removal -p
FAILED_OSD_ID=${osd_id_to_remove} | oc create -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。

- 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
```

- 置き換える OSD に関連付けられた PVC リソースを削除します。

- 置き換える OSD に関連付けられた **DeviceSet** を特定します。

```
# oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

出力例:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

- b. PVC に関連付けられた PV を特定します。

```
# oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
NAME                STATUS VOLUME                CAPACITY ACCESS
MODES STORAGECLASS AGE
ocs-deviceset-0-0-nvs68 Bound pvc-0e621d45-7d18-4d35-a282-9700c3cc8524
512Gi RWO thin 24h
```

この例では、PVC は直前の手順で特定される **ocs-deviceset-0-0-nvs68** になり、関連付けられた PV は **pvc-0e621d45-7d18-4d35-a282-9700c3cc8524** になります。

- c. 置き換える OSD に関連付けられた **prepare-pod** を特定します。直前の手順で取得した PVC 名を使用します。

```
# oc describe -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix> | grep
Mounted
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
Mounted By: rook-ceph-osd-prepare-ocs-deviceset-0-0-nvs68-zblp7
```

- d. 関連付けられた PVC を削除する前に **osd-prepare** Pod を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-osd-prepare-ocs-deviceset-<x>-<y>-
<pvc-suffix>-<pod-suffix>
```

ここで、**x**、**y**、**pvc-suffix**、および **pod-suffix** は、直前の手順で特定された **osd-prepare** Pod 名の値です。

出力例:

```
pod "rook-ceph-osd-prepare-ocs-deviceset-0-0-nvs68-zblp7" deleted
```

- e. デバイスに関連付けられた PVC を削除します。

```
# oc delete -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
persistentvolumeclaim "ocs-deviceset-0-0-nvs68" deleted
```

7. 新規デバイス用に新規 OSD を作成します。

a. 置き換えられる OSD のデプロイメントを削除します。

```
# oc delete -n openshift-storage deployment rook-ceph-osd-${osd_id_to_remove}
```

出力例:

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

b. 先の手順で特定されたデバイスの PV が削除されていることを確認します。

```
# oc get -n openshift-storage pv pvc-0e621d45-7d18-4d35-a282-9700c3cc8524
```

出力例:

```
Error from server (NotFound): persistentvolumes "pvc-0e621d45-7d18-4d35-a282-9700c3cc8524" not found
```

この例では、PV 名は **pvc-0e621d45-7d18-4d35-a282-9700c3cc8524** です。

- PV がまだ存在する場合は、そのデバイスに関連付けられた PV を削除します。

```
# oc delete pv pvc-0e621d45-7d18-4d35-a282-9700c3cc8524
```

出力例:

```
persistentvolume "pvc-0e621d45-7d18-4d35-a282-9700c3cc8524" deleted
```

この例では、PV 名は **pvc-0e621d45-7d18-4d35-a282-9700c3cc8524** です。

c. **rook-ceph-operator** を再起動して新規 OSD をデプロイし、Operator の調整を強制的に実行します。

i. **rook-ceph-operator** の名前を特定します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-2d982 1/1   Running 0      1d20h
```

ii. **rook-ceph-operator** を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```

出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

この例では、rook-ceph-operator Pod 名は **rook-ceph-operator-6f74fb5bff-2d982** です。

- iii. **rook-ceph-operator** Pod が再起動していることを確認します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running 0      66s
```

新規 OSD の作成には、Operator が再起動するまでに数分かかる場合があります。

8. **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

検証手順

- 新しい OSD が実行されており、新規 PVC が作成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
```

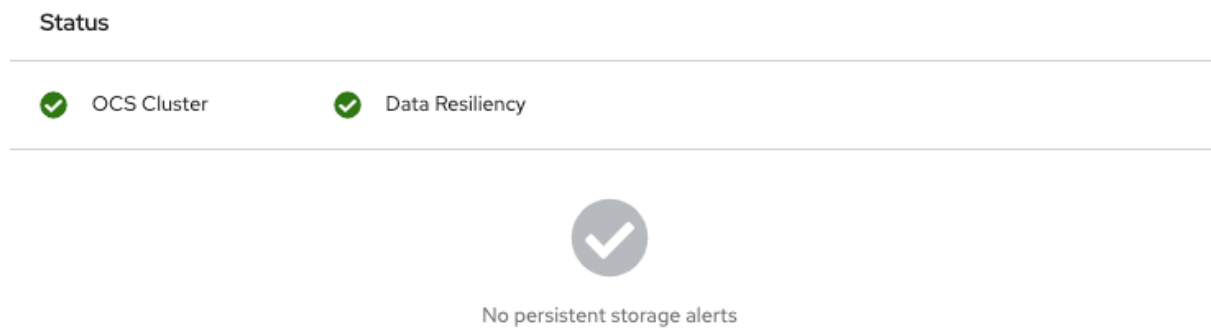
```
# oc get -n openshift-storage pvc
```

出力例:

```
NAME                                STATUS VOLUME                                     CAPACITY ACCESS
MODES STORAGECLASS AGE
ocs-deviceset-0-0-2s6w4 Bound  pvc-7c9bcaf7-de68-40e1-95f9-0b0d7c0ae2fc 512Gi
RWO    thin      5m
ocs-deviceset-1-0-q8fwh Bound  pvc-9e7e00cb-6b33-402e-9dc5-b8df4fd9010f 512Gi
RWO    thin      1d20h
ocs-deviceset-2-0-9v8lq Bound  pvc-38cdfcee-ea7e-42a5-a6e1-aaa6d4924291 512Gi
RWO    thin      1d20h
```

- 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10.1 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



10.3. ローカルストレージデバイスを使用した OPENSIFT CONTAINER STORAGE のデプロイ

10.3.1. Amazon EC2 インフラストラクチャーでの障害のあるストレージノードの置き換え

Amazon EC2（ストレージ最適化 I3）インフラストラクチャーのストレージデバイスを置き換える必要がある場合は、ストレージノードを置き換える必要があります。ノードを置き換える方法については、「[Amazon EC2 インフラストラクチャーでの障害のあるストレージノードの置き換え](#)」を参照してください。

10.3.2. VMware およびベアメタルインフラストラクチャーで動作するストレージデバイスまたは失敗したストレージデバイスの置き換え

ベアメタルおよび VMware インフラストラクチャーでローカルストレージデバイスを使用してデプロイされた OpenShift Container Storage のオブジェクトストレージデバイス (OSD) を置き換えることができます。基礎となるストレージデバイスを置き換える必要がある場合は、この手順を使用します。

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、置き換える必要があるのは **rook-ceph-osd-0-6d77d6c7c6-m8xj6** であり、OSD がスケジュールされている OCP ノードは **compute-2** です。



注記

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

- 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** プレフィックスの直後にくる 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
```

- 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 in openshift-storage namespace.
```



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 --grace-period=0 --force
```

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
```

- 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。

- 古い **ocs-osd-removal** ジョブを削除します。

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

出力例:

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

- 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_ID=${osd_id_to_remove} | oc create -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. 置き換える OSD に関連付けられた Persistent Volume Claim (永続ボリューム要求、PVC) リソースを削除します。
 - a. 置き換える OSD に関連付けられた **DeviceSet** を特定します。

```
# oc get -n openshift-storage -o yaml deployment rook-ceph-osd-${osd_id_to_remove} |
grep ceph.rook.io/pvc
```

出力例:

```
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
ceph.rook.io/pvc: ocs-deviceset-0-0-nvs68
```

この例では、PVC 名は **OCS-deviceset-0-0-nvs68** です。

- b. PVC に関連付けられた PV を特定します。

```
# oc get -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
NAME                STATUS    VOLUME    CAPACITY  ACCESS MODES
STORAGECLASS  AGE
```

```
ocs-deviceset-0-0-nvs68 Bound local-pv-d9c5cbd6 1490Gi RWO localblock
24h
```

この例では、関連付けられた PV は **local-pv-d9c5cbd6** です。

- c. 置き換えるデバイスの名前を特定します。

```
# oc get pv local-pv-<pv-suffix> -o yaml | grep path
```

ここで、**pv-suffix** は、前のステップで特定された PV 名の値です。

出力例:

```
path: /mnt/local-storage/localblock/nvme0n1
```

この例では、デバイス名は **nvme0n1** です。

- d. 置き換える OSD に関連付けられた **prepare-pod** を特定します。

```
# oc describe -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix> | grep
Mounted
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
Mounted By: rook-ceph-osd-prepare-ocs-deviceset-0-0-nvs68-zblp7
```

この例では、**prepare-pod** 名は **rook-ceph-osd-prepare-ocs-deviceset-0-0-nvs68-zblp7** です。

- e. 関連付けられた PVC を削除する前に **osd-prepare** Pod を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-osd-prepare-ocs-deviceset-<x>-<y>-
<pvc-suffix>-<pod-suffix>
```

ここで、**x**、**y**、**pvc-suffix**、および **pod-suffix** は、直前の手順で特定された **osd-prepare** Pod 名の値です。

出力例:

```
pod "rook-ceph-osd-prepare-ocs-deviceset-0-0-nvs68-zblp7" deleted
```

- f. 置き換える OSD に関連付けられた PVC を削除します。

```
# oc delete -n openshift-storage pvc ocs-deviceset-<x>-<y>-<pvc-suffix>
```

ここで、**x**、**y**、および **pvc-suffix** は、直前の手順で特定された **DeviceSet** の値です。

出力例:

```
persistentvolumeclaim "ocs-deviceset-0-0-nvs68" deleted
```

7. 古いデバイスを置き換え、新規デバイスを使用して新規の OpenShift Container Platform PV を作成します。
- 置き換えるデバイスのある OpenShift Container Platform ノードにログインします。この例では、OpenShift Container Platform ノードは **compute-2** です。

```
# oc debug node/compute-2
```

出力例:

```
Starting pod/compute-2-debug ...
To use host binaries, run `chroot /host`
Pod IP: 10.70.56.66
If you don't see a command prompt, try pressing enter.
# chroot /host
```

- 先に特定したデバイス名 **nvme0n1** を使用して置き換える **/dev/disk/by-id/{id}** を記録します。

```
# ls -alh /mnt/local-storage/localblock
```

出力例:

```
total 0
drwxr-xr-x. 2 root root 51 Aug 18 19:05 .
drwxr-xr-x. 3 root root 24 Aug 18 19:05 ..
lrwxrwxrwx. 1 root root 57 Aug 18 19:05 nvme0n1 -> /dev/disk/by-id/nvme-
eui.01000000010000005cd2e4de2f0f5251
```

- LocalVolume** CR の名前を見つけ、置き換えるデバイス **/dev/disk/by-id/{id}** を削除またはコメントアウトします。

```
# oc get -n local-storage localvolume
NAME      AGE
local-block 25h
```

```
# oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4895e0e5251
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4ea2f0f5251
# - /dev/disk/by-id/nvme-eui.01000000010000005cd2e4de2f0f5251
storageClassName: localblock
volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

8. 置き換えるデバイスで OpenShift Container Platform ノードにログインし、古い **symlink** を削除します。

```
# oc debug node/compute-2
```

出力例:

```
Starting pod/compute-2-debug ...
To use host binaries, run `chroot /host`
Pod IP: 10.70.56.66
If you don't see a command prompt, try pressing enter.
# chroot /host
```

- a. 置き換えるデバイス名の古い **symlink** を特定します。この例では、デバイス名は **nvme0n1** です。

```
# ls -alh /mnt/local-storage/localblock
```

出力例:

```
total 0
drwxr-xr-x. 2 root root 51 Aug 18 19:05 .
drwxr-xr-x. 3 root root 24 Aug 18 19:05 ..
lrwxrwxrwx. 1 root root 57 Aug 18 19:05 nvme0n1 -> /dev/disk/by-id/nvme-eui.01000000010000005cd2e4de2f0f5251
```

- b. **symlink** を削除します。

```
# rm /mnt/local-storage/localblock/nvme0n1
```

- c. **symlink** が削除されていることを確認します。

```
# ls -alh /mnt/local-storage/localblock
```

出力例:

```
total 0
drwxr-xr-x. 2 root root 17 Apr 10 00:56 .
drwxr-xr-x. 3 root root 24 Apr 8 23:03 ..
```

重要

OpenShift Container Storage 4.5 以降の新規デプロイメントでは、LVM が使用されていないため、**ceph-volume** raw モードが動作します。そのため、追加の検証は不要であり、次のステップに進むことができます。

OpenShift Container Storage 4.4 の場合、または OpenShift Container Storage が以前のバージョンからバージョン 4.5 にアップグレードされた場合、**/dev/mapper** と **/dev/** の両方をチェックし、移行前に **ceph** に関連する孤立したオブジェクトがあるかどうかを確認します。**vgdisplay** の結果を使用して、これらの孤立を見つけます。**/dev/mapper** または **/dev/ceph-*** に、VG 名の一覧にない名前に **ceph** が設定されている場合は、**dmsetup** を使用してこれを削除します。

9. 先の手順で特定された、置き換えるデバイスに関連付けられた PV を削除します。この例では、PV 名は **local-pv-d9c5cbd6** です。

```
# oc delete pv local-pv-d9c5cbd6
```

出力例:

```
persistentvolume "local-pv-d9c5cbd6" deleted
```

10. デバイスを新しいデバイスに置き換えます。
11. 正しい OpenShift Container Platform ノードにログインし、新規ドライブのデバイス名を特定します。デバイス名は古いデバイスと同じにすることができますが、同じデバイスを使用しない限り **by-id** は変更する必要があります。

```
# lsblk
```

出力例:

```
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                                  8:0  0  120G  0 disk
|-sda1                               8:1  0   384M  0 part /boot
|-sda2                               8:2  0   127M  0 part /boot/efi
|-sda3                               8:3  0     1M  0 part
`-sda4                               8:4  0  119.5G  0 part
   `--coreos-luks-root-nocrypt 253:0  0  119.5G  0 dm  /sysroot
nvme0n1                             259:0  0   1.5T  0 disk
```

この例では、新しいデバイス名は **nvme0n1** です。

- a. 新規デバイスの **/dev/disk/by-id/{id}** を特定し、これを記録します。

```
# ls -alh /dev/disk/by-id | grep nvme0n1
```

出力例:

```
lrwxrwxrwx. 1 root root 57 Aug 18 19:05 nvme0n1 -> /dev/disk/by-id/nvme-eui.01000000010000005cd2e4ce090e5251
```

12. 新規の **/dev/disk/by-id/{id}** が利用可能になると、新規ディスクエントリーを **LocalVolume** CR に追加できます。

- a. **LocalVolume** CR の名前を検索します。

```
# oc get -n local-storage localvolume
NAME      AGE
local-block 25h
```

- b. **LocalVolume** CR を編集し、新規の **/dev/disk/by-id/{id}** を追加します。この例では、新しいデバイスは **/dev/disk/by-id/nvme-eui.01000000010000005cd2e4ce090e5251** です。

```
# oc edit -n local-storage localvolume local-block
```

出力例:

```
[...]
storageClassDevices:
- devicePaths:
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4895e0e5251
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4ea2f0f5251
# - /dev/disk/by-id/nvme-eui.01000000010000005cd2e4de2f0f5251
- /dev/disk/by-id/nvme-eui.01000000010000005cd2e4ce090e5251
storageClassName: localblock
volumeMode: Block
[...]
```

CR の編集後に変更を保存するようにしてください。

13. 新規 PV が **Available** 状態にあり、正しいサイズであることを確認します。

```
# oc get pv | grep 1490Gi
```

出力例:

local-pv-3e8964d3	1490Gi	RWO	Delete	Bound	openshift-
storage/ocs-deviceset-2-0-79j94	localblock		25h		
local-pv-414755e0	1490Gi	RWO	Delete	Bound	openshift-
storage/ocs-deviceset-1-0-959rp	localblock		25h		
local-pv-b481410	1490Gi	RWO	Delete	Available	

14. 新規デバイス用に新規 OSD を作成します。

- a. 置き換えられる OSD のデプロイメントを削除します。

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

出力例:

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

- b. **rook-ceph-operator** を再起動して Operator の調整を強制的に実行して新規 OSD をデプロイします。

- i. **rook-ceph-operator** の名前を特定します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

NAME	READY	STATUS	RESTARTS	AGE
rook-ceph-operator-6f74fb5bff-2d982	1/1	Running	0	1d20h

- ii. **rook-ceph-operator** を削除します。

```
# oc delete -n openshift-storage pod rook-ceph-operator-6f74fb5bff-2d982
```


出力例:

```
pod "rook-ceph-operator-6f74fb5bff-2d982" deleted
```

この例では、rook-ceph-operator Pod 名は **rook-ceph-operator-6f74fb5bff-2d982** です。

- iii. **rook-ceph-operator** Pod が再起動していることを確認します。

```
# oc get -n openshift-storage pod -l app=rook-ceph-operator
```

出力例:

```
NAME                                READY STATUS RESTARTS AGE
rook-ceph-operator-6f74fb5bff-7mvrq 1/1   Running 0       66s
```

新規 OSD の作成には、Operator が再起動するまでに数分かかる場合があります。

検証手順

- 新しい OSD が実行されており、新規 PVC が作成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
```

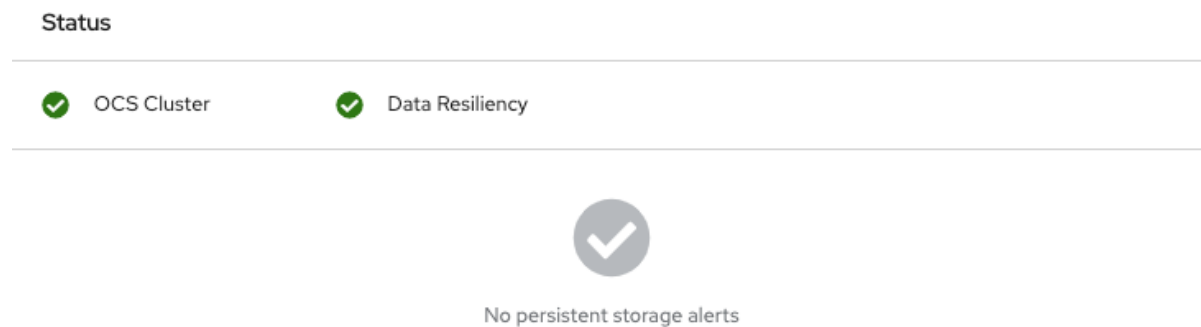
```
# oc get -n openshift-storage pvc | grep localblock
```

出力例:

```
ocs-deviceset-0-0-c2mqb Bound local-pv-b481410      1490Gi RWO
localblock                    5m
ocs-deviceset-1-0-959rp Bound local-pv-414755e0      1490Gi RWO
localblock                    1d20h
ocs-deviceset-2-0-79j94 Bound local-pv-3e8964d3      1490Gi RWO
localblock                    1d20h
```

- 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10.2 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11章 OPENSIFT CONTAINER STORAGE の更新

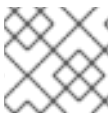
クラスターを更新するには、まず Red Hat OpenShift Container Platform を更新してから、Red Hat OpenShift Container Storage を更新します。Red Hat OpenShift Container Storage で同じバージョンの Red Hat OpenShift Container Platform を使用することを推奨します。完全な OpenShift Container Platform および OpenShift Container Storage のサポートの容易性や互換性のマトリックスについては、[Red Hat ナレッジベースアトキクル](#) を参照してください。

ローカルストレージ Operator の更新:

- ローカルストレージ Operator が Red Hat OpenShift Container Storage で完全にサポートされるために、ローカルストレージ Operator のバージョンは Red Hat OpenShift Container Platform バージョンと一致する必要があります。
- ローカルストレージ Operator は、Red Hat OpenShift Container Platform の更新時に更新されません。OpenShift Container Storage クラスターがローカルストレージ Operator を使用するかどうかを確認するには、『トラブルシューティングガイド』の [ローカルストレージ Operator デプロイメントの確認](#) についてのセクションを参照してください。

Red Hat OpenShift Container Storage を更新できます。

- [内部モード](#)
- [外部モード](#)
- [非接続環境](#)



注記

更新手順は、プロキシ環境の場合と同じです。

11.1. 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。

11.1.1. 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** の下で、**OCS クラスター** が正常であり、データに回復性があることを確認します。
- OpenShift Container Platform クラスターをバージョン 4.4.X または 4.5.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-4.4 から **stable-4.5** に切り替え **ま**す。チャンネルについての詳細は、「[OpenShift Container Platform アップグレードチャンネルおよびリリース](#)」を参照してください。



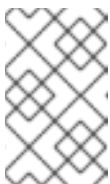
注記

マイナーバージョンを更新する場合（例：4.4 から 4.5 に更新）にのみチャンネルを切り換える必要があり、4.5 のバッチ更新間で更新する場合（例：4.5.0 から 4.5.1 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 (OCS) 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

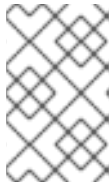
Upgrade status には、新規 OpenShift Container Storage バージョンがチャンネルですでに検知され、承認ストラテジーが更新時に **Manual** から **Automatic** に変更されている場合に **requires approval** が表示されます。

- 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. **Overview** → **Persistent Storage** タブをクリックし、**Status** カードで、**OCS クラスター** に正常であることを示す緑色のチェックマークが表示されていることを確認します。
2. **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 4.4 から 4.5 に更新された後も、**Version** フィールドには依然として 4.4 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。

3. 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。
4. 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に一般に必要なログ](#)」セクションを参照してください。

11.1.2. 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** の下で、**OCS クラスター** が正常であり、データに回復性があることを確認します。
- OpenShift Container Platform クラスターをバージョン 4.4.X または 4.5.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-4.4** から **stable-4.5** に切り替え **ま**す。チャンネルについての詳細は、「[OpenShift Container Platform アップグレードチャンネルおよびリリース](#)」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：4.4 から 4.5 に更新）にのみチャンネルを切り換える必要があり、4.5 のバッチ更新間で更新する場合（例：4.5.0 から 4.5.1 に更新）はチャンネルを切り換える必要はありません。

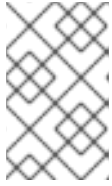
- Operator Pod を含むすべての OpenShift Container Storage Pod が **openshift-storage namespace** で **Running** 状態にあることを確認します。
Pod の状態を確認するには、OpenShift Web コンソールの左側のペインから **Workloads** → **Pods** をクリックします。Project ドロップダウンリストから **openshift-storage** を選択します。
- 更新時間はクラスターで実行される OSD の数によって異なるため、Openshift Container Storage (OCS) 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. **Overview** → **Persistent Storage** タブをクリックし、**Status** カードで、**OCS** クラスターに正常であることを示す緑色のチェックマークが表示されていることを確認します。
2. **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 4.4 から 4.5 に更新された後も、**Version** フィールドには依然として 4.4 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。

- 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。
- 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に一般に必要なログ](#)」セクションを参照してください。

11.2. 外部モードでの OPENSIFT CONTAINER STORAGE の更新

以下の手順に従って、外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。



注記

Red Hat OpenShift Container Storage Operator を更新しても、外部の Red Hat Ceph Storage クラスターは更新されません。これは、OpenShift Container Platform で実行されている Red Hat OpenShift Container Storage サービスのみを更新します。外部の Red Hat Ceph Storage クラスターを更新するには、Red Hat Ceph Storage の管理者にお問い合わせください。

11.2.1. 外部モードでの 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。外部モードでの OpenShift Container Storage の自動更新はバージョン 4.5 以降でサポートされます。



注記

OpenShift Container Storage を更新しても、外部の Red Hat Ceph Storage クラスターは更新されません。

前提条件

- OpenShift Container Platform クラスターをバージョン 4.5.x の最新の安定したリリースに更新する場合は、『[クラスターの更新](#)』を参照してください。
- Red Hat OpenShift Container Storage チャンネルが **stable-4.5** に設定されていることを確認します。「[OpenShift Container Platform アップグレードチャンネルおよびリリース](#)」を参照してください。



注記

4.5 のバッチ更新間で更新する際にチャンネルを切り替える必要はありません（例：4.5.0 から 4.5.1 に更新）。

- 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。
- Status カードの **Persistent Storage** の下で、**OCS クラスタ** が正常であることを確認します。
- Openshift Container Storage (OCS) の更新プロセスを完了するための十分な時間が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. **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、承認ストラテジーの更新時に **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. **Overview** → **Persistent Storage** タブをクリックし、**Status** カードで、**OCS** クラスタに正常であることを示す緑色のチェックマークが表示されていることを確認します。
2. **Operators** → **Installed Operators** → **OpenShift Container Storage Operator** をクリックします。**Storage Cluster** で、クラスタサービスのステータスが **Ready** であることを確認します。
3. 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。
4. 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に一般に必要なログ](#)」セクションを参照してください。

11.2.2. 外部モードでの 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。外部モードでの OpenShift Container Storage の手動更新は、バージョン 4.5 以降でサポートされます。



注記

OpenShift Container Storage を更新しても、外部の Red Hat Ceph Storage クラスタは更新されません。

前提条件

- OpenShift Container Platform クラスタを 4.5.x の最新の安定したリリースに更新する場合は、『[クラスタの更新](#)』を参照してください。
- Red Hat OpenShift Container Storage チャンネルが **stable-4.5** に設定されていることを確認します。「[OpenShift Container Platform アップグレードチャンネルおよびリリース](#)」を参照してください。



注記

4.5 のバッチ更新間で更新する際にチャンネルを切り替える必要はありません（例：4.5.0 から 4.5.1 に更新）。

- 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。
- **Status** カードの **Persistent Storage** の下で、**OCS クラスタ** が正常であることを確認します。
- OpenShift Container Storage (OCS) の更新プロセスを完了するための十分な時間が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. **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. **Overview** → **Persistent Storage** タブをクリックし、**Status** カードで、**OCS クラスタ** に正常であることを示す緑色のチェックマークが表示されていることを確認します。
2. **Operators** → **Installed Operators** → **OpenShift Container Storage Operator** をクリックします。**Storage Cluster** で、クラスタサービスのステータスが **Ready** であることを確認します。
3. 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。

4. 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に一般に必要なログ](#)」セクションを参照してください。

11.3. 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。

11.3.1. 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`

- 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。以下のようになります。

```
{
  "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"
    }
  }
}
```

- 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>"
```

- 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": "*****",
  }
}
}

```

11.3.2. 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:v4.5 \
  --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

```

11.3.3. 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。

11.3.4. redhat-operator CatalogSource の更新

手順

1. Red Hat Operator のカタログイメージを参照する **CatalogSource** オブジェクトを再作成します。



注記

正しいバージョン (**v2**) で正しいカタログソースをミラーリングしていることを確認します。

+ 以下を **redhat-operator-catalogsource.yaml** ファイルに保存し、**< your_registry >** をミラーレジストリー URL に必ず置き換えます。

+

```

apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: redhat-operators
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  icon:
    base64data:
      PHN2ZyBpZD0iTGZF5ZXJfMSlgZGF0YS1uYW11PSJMYXllciAxliB4bWxucz0iaHR0cDovL3d3dy53My5vc
      mcvMjAwMC9zdmcilHZpZXdCb3g9IjAgMCAxOTIlgMTQ1Ij48ZGVmcmcz48c3R5bGU+LmNscy0xe2ZpbG
      w6I2UwMDt9PC9zdHlsZT48L2RlZnM+PHRpdGxIPiJIZehhdC1Mb2dvLUhhdC1Db2xvcjwvdGl0bGU+P
      HBhdGggZD0iTTE1Ny43Nyw2Mi42MWEwNCwxcwNCwwLDAwMSwzMy40MmMwLDE0Ljg4LTE4Lj
      EsMTcuNDYtMzAuNjEsMTcuNDZDNzguODMsODMuNDksNDluNTMsNTMuMjYsNDluNTMsNDRhNi4

```

```
0Myw2LjQzLDAsMCwxLC4yMi0xLjk0bC0zLjY2LDkuMDZhMTguNDUsMTguNDUsMCwwLDAAtMS41M
Sw3LjMzYzAsMTguMTEsNDEsNDUuNDgsODcuNzQsNDUuNDgsMjAuNjksMCwzNi40My03Ljc2LDM2
LjQzLTlxLjc3LDAAtMS4wOCwwLTEuOTQtMS43My0xMC4xM1oiLz48cGF0aCBjbGFzc20iY2xzLTEiIGQ
9Ik0xMjcuNDcsODMuNDIjMTIuNTEsMCwzMC42MS0yLjU4LDMwLjYxLjE3LjQ2YTE0LDE0LDAAsMCw
wLS4zMS0zLjQybC03LjQ1LTMyljM2Yy0xLjcyLTcuMTItMy4yMy0xMC4zNS0xNS43My0xNi42QzEyNC
4OSw4LjY5LDEwMy43Ni41LDk3LjUxLjUsOTEuNjkuNSw5MCw4LDgzLjA2LDhjLTYuNjgsMC0xMS42N
C01LjYtMTcuODktNS42LTYsMC05LjkxLDQuMDktMTIuOTMsMTIuNSwwLDAAtOC40MSwyMy43Mi05Lj
Q5LDI3LjE2QTYuNDMsNi40MywwLDAAsMCw0Mi41Myw0NGMwLDkuMjlsMzYuMywzOS40NSw4NC4
5NCwzOS40NU0xNjAsNzluMDdjMS43Myw4LjE5LDEuNzMsOS4wNSwxLjczLDEwLjEzLDAAsMTQtMTU
uNzQsMjEuNzctMzYuNDMsMjEuNzdDNzguNTQsMTA0LDM3LjU4LDC2LjYsMzcuNTgsNTguNDIhMTg
uNDUsMTguNDUsMCwwLDEsMS41MS03LjMzQzlyLjI3LDUyLC41LDU1LC41LDC0LjlyYzAsMzEuNDg
sNzQuNTksNzAuMjgsMTMzLjY1LDcwLjI4LDQ1LjI4LDAAsNTYuNy0yMC40OCw1Ni43LTM2LjY1LDAAtM
TluNzltMTEtMjcuMTYtMzAuODMtMzUuNzgiLz48L3N2Zz4=
```

```
mediatype: image/svg+xml
```

```
image: <your_registry>/olm/redhat-operators:v2
```

```
displayName: Redhat Operators Catalog
```

```
publisher: Red Hat
```

1. redhat-operator-catalogsource.yaml ファイルを使用してカタログソースを作成します。

```
$ oc apply -f redhat-operator-catalogsource.yaml
```

2. 新規の **redhat-operator** Pod が実行していることを確認します。

```
$ oc get pod -n openshift-marketplace | grep redhat-operators
```

11.3.5. 更新の継続

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](#)