Red Hat OpenShift Container Storage 4.4

Red Hat OpenShift Container Storage のデプロイ

環境のインストールおよび設定方法

Last Updated: 2020-07-30

Red Hat OpenShift Container Storage 4.4 Red Hat OpenShift Container Storage のデプロイ

環境のインストールおよび設定方法

法律上の通知

概要

Red Hat OpenShift Container Storage 4.4 のインストール手順については、本書を参照してください。

目次

はじめに

OpenShift Container Storage 4.4 のインストールは、既存の Red Hat OpenShift Container Platform (OCP) ワーカーノードでのみサポートされます。

第1章 OPENSHIFT CONTAINER STORAGE の OPENSHIFT CONTAINER PLATFORM へのデプロイ

デプロイメントプロセスは、以下の 2 つの主要な部分で構成されます。

1. 「Operator Hub を使用した Red Hat OpenShift Container Storage Operator のインストール」の手順に従って OpenShift Container Storage Operator をインストールします。
2. 「OpenShift Container Storage サービスの作成」 の手順に従って OpenShift Container Storage サービスを作成します。

注記

- OpenShift Container Storage をネットワークが制限された環境でインストールする場合、デフォルトでインターネット接続が OpenShift Container Platform で想定され、**chronyd** が ***.rhel.pool.ntp.org** サーバーを使用するように設定されるため、カスタム Network Time Protocol (NTP) 設定をノードに適用する必要があります。詳細は、Red Hat ナレッジベースの記事 およびchrony タイムサービスの設定について参照してください。
- OpenShift Container Platform アラートである **PodDisruptionBudget** アラートは、デプロイメント後にオブジェクトストレージデバイス (OSD) について表示されます。このアラートは無視できます。また、Openshift Container Platform ドキュメントの「クラスターアラートの管理」についての手順に従ってこのアラートをサイレンスにすることができます。これを実行する方法については、Red Hat Openshift Container Platform ドキュメントの「Managing cluster alerts」セクションを参照してください。
  詳細は、Red Hat ナレッジベースアーティクル 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 (UPI) の Red Hat Enterprise Linux ベースのホストについては、「Red Hat Enterprise Linux ベースのノード上のコンテナーでのファイルシステムアクセスの有効化」 の手順に従って、基礎となるファイルシステムへのコンテナーのアクセスを有効にする必要があります。

1.1. OPERATOR HUB を使用した RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR のインストール

Red Hat OpenShift Container Storage は、Amazon Web Services (AWS) および VMware vSphere プラットフォームで Red Hat OpenShift Container Platform Operator Hub を使用してインストールできます。ハードウェアおよびソフトウェア要件の詳細は、『デプロイメントのプランニング』ガイドを参照してください。

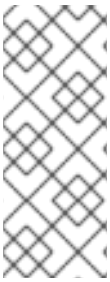
前提条件

- OpenShift Container Platform クラスターにログインします。
- OpenShift Container Platform クラスターにワーカーノードが少なくとも 3 つ必要です。
- 以下のように、**openshift-storage** という namespace を作成する必要があります。
  1. OpenShift Web コンソールの左側のペインで、**Administration → Namespaces** をクリックします。
  2. **Create Namespace** をクリックします。

3. Create Namespace ダイアログボックスで、Name には **openshift-storage** を、Labels には **openshift.io/cluster-monitoring=true** を入力します。このラベルは、ダッシュボードを取得するために必要です。

4. **Default Network Policy** に **No restrictions** オプションを選択します。

5. **Create** をクリックします。



注記

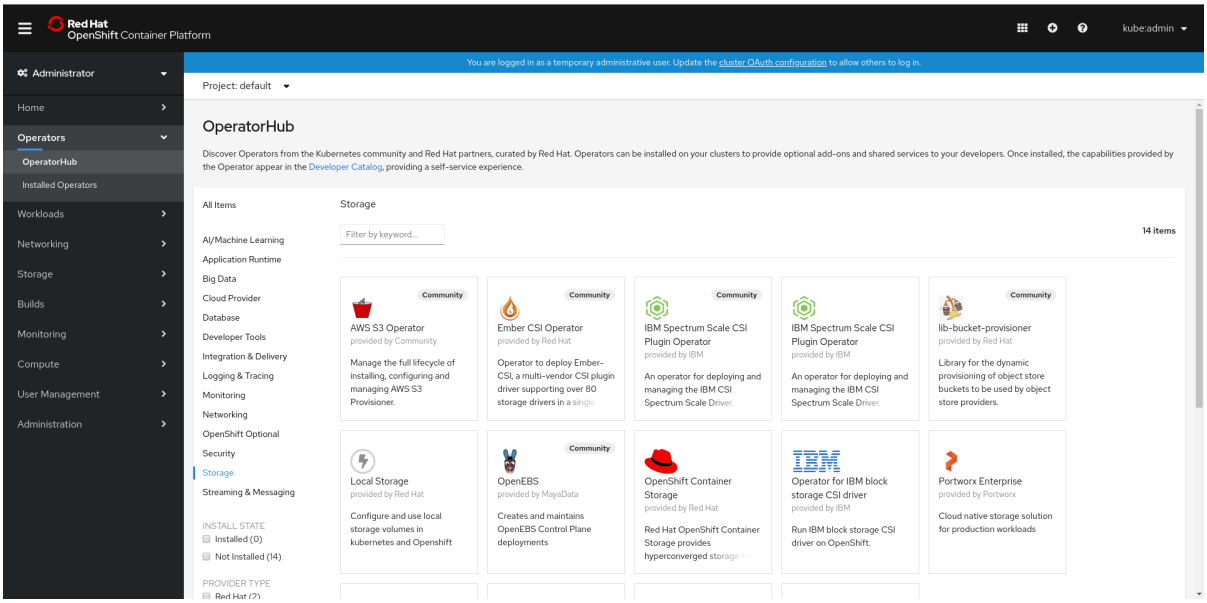
OpenShift Container Storage のクラスター全体でのデフォルトノードセレクターを上書きする必要がある場合は、コマンドラインインターフェースで以下のコマンドを使用し、**openshift-storage** namespace の空のノードセレクターを指定できます。

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

手順

1. OpenShift Web コンソールの左側のペインで、**Operators → OperatorHub** をクリックします。

図1.1 Operator Hub の Operator 一覧

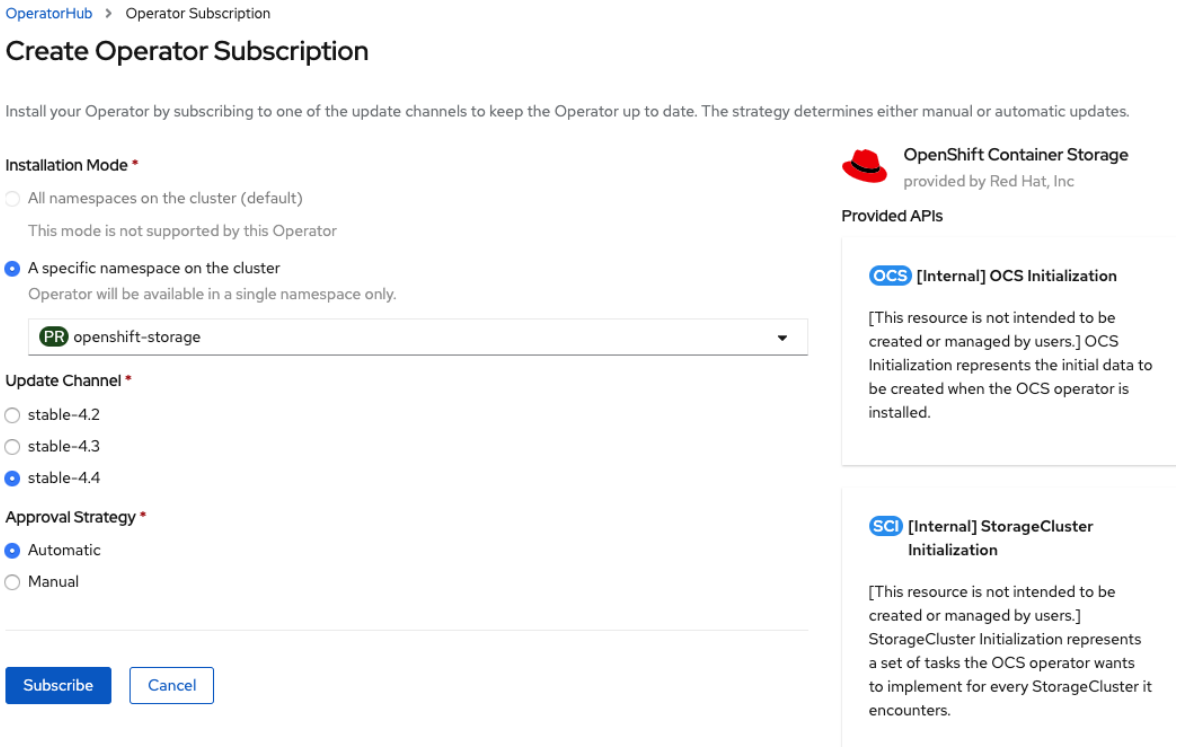


2. Operator の一覧から **OpenShift Container Storage** を検索し、これをクリックします。

3. OpenShift Container Storage Operator ページで **Install** をクリックします。

4. Create Operator Subscription ページで、Installation Mode、Update Channel、および Approval Strategy オプションを選択できます。

図1.2 Create Operator Subscription ページ



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

- ドロップダウンメニューから **openshift-storage** namespace を選択します。

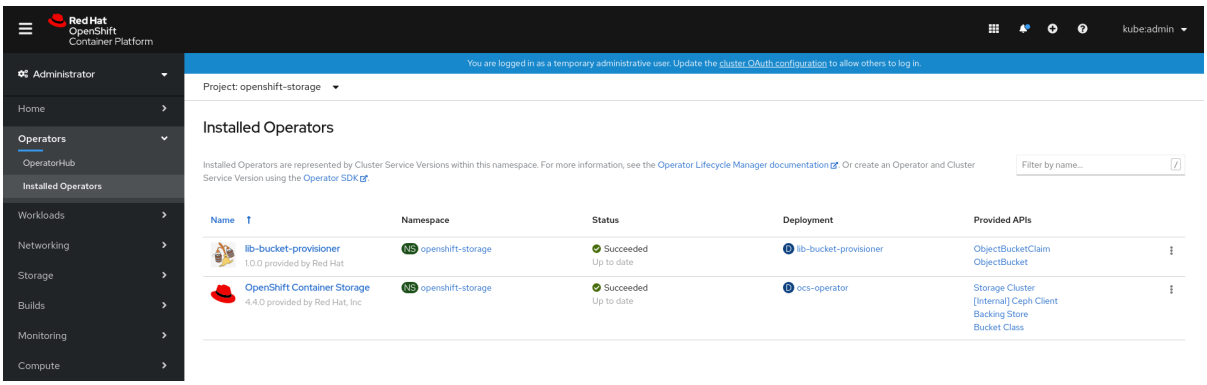
b. 更新チャネルとして **stable-4.4** を選択します。

c. 承認ストラテジーを選択します。

- **Automatic** は、OpenShift Container Platform が OpenShift Container Storage を自動的にアップグレードすることを指定します。
- **Manual** は、OpenShift Container Storage を手動でアップグレードする際に制御できることを指定します。

5. **Subscribe** をクリックします。

図1.3 インストールされた Operator



Installed Operators ページには、Operator のステータスが表示されます。

検証手順

- **OpenShift Container Storage** Operator のステータスが **Succeeded** であることを確認します。

1.2. OPENSHIFT CONTAINER STORAGE サービスの作成

OpenShift Container Storage Operator のインストール後に新規の OpenShift Container Storage サービスを作成する必要があります。

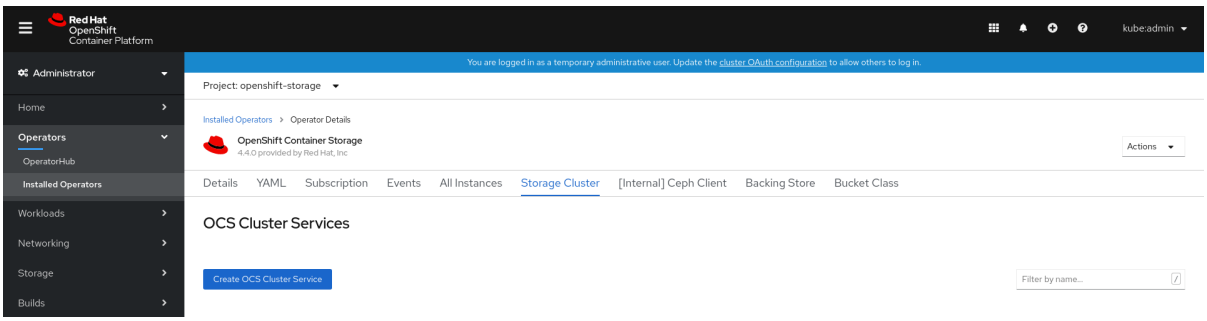
前提条件

- OpenShift Container Storage Operator は Operator Hub からインストールする必要があります。詳細は、Operator Hub を使用した OpenShift Container Storage Operator のインストールについて参照してください。

手順

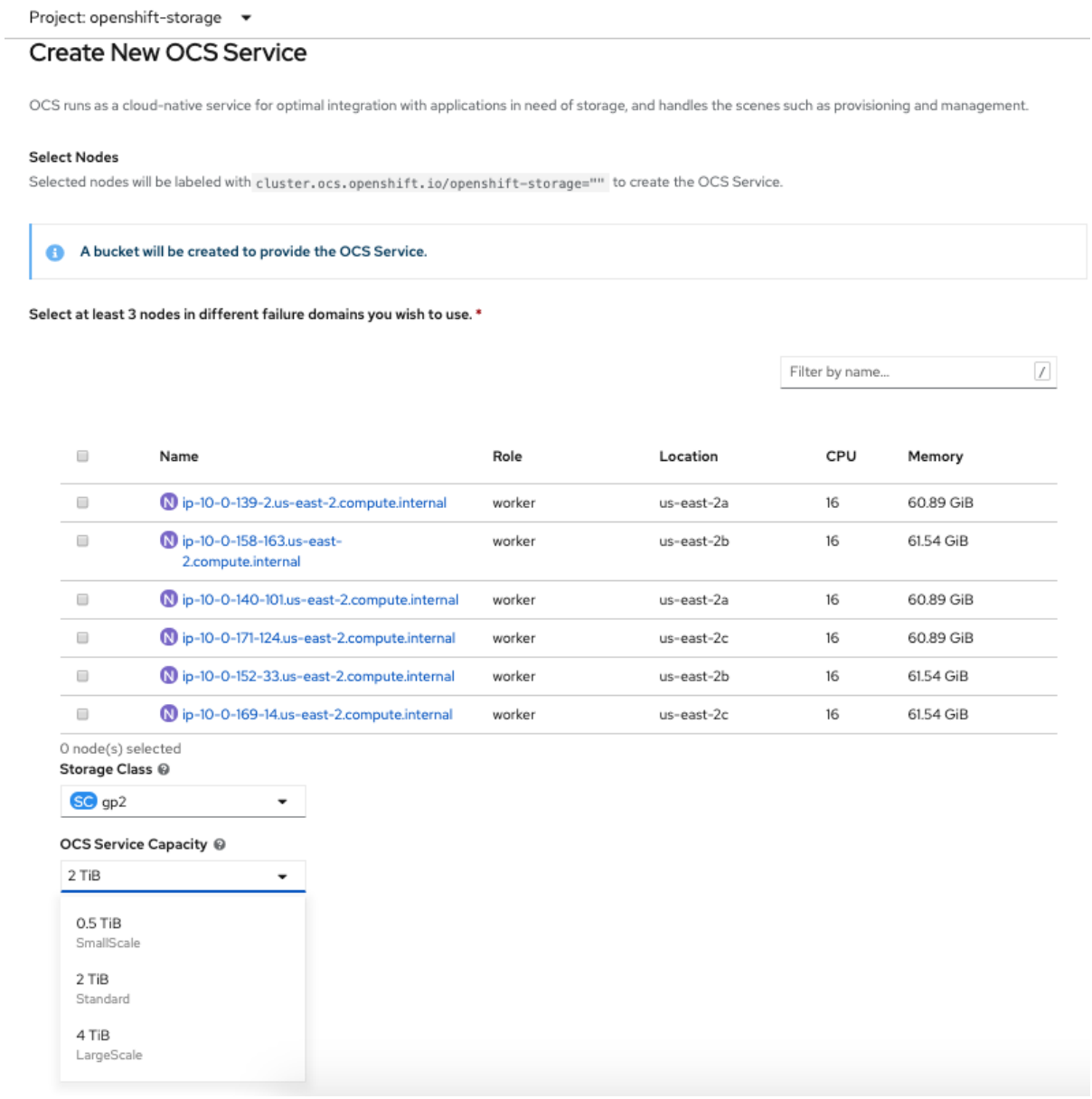
1. OpenShift Web コンソールの左側のペインで **Operators → Installed Operators** をクリックし、インストールされた Operator を表示します。
2. Installed Operator ページで、**Project** ドロップダウンリストから **openshift-storage** を選択し、**openshift-storage** プロジェクトに切り替えます。
3. **OpenShift Container Storage** Operator をクリックします。
   OpenShift Container Storage Operator は **OCSInitialization** リソースを自動的に作成します。
4. OpenShift Container Storage Operator ページで、右スクロールし、**Storage Cluster** タブをクリックします。

   図1.4 OpenShift Container Storage Operator ページ



5. **OCS Cluster Services** ページで、**Create OCS Cluster Service** をクリックします。

図1.5 新規 OCS サービスページの作成

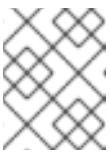


6. **Create New OCS Service**ページで、以下を実行します。

   a. OpenShift Container Storage サービスを使用するために、利用可能なノードの一覧から 3 つ以上の ワーカーノードを選択します。ノードが異なる **Location** にあることを確認します。

   b. **Storage Class** は、プラットフォームに応じてデフォルトで設定されます。
   gp2 は AWS のデフォルトストレージクラスであり、 **thin** は VMware のデフォルトストレージクラスです。

   c. ドロップダウンリストから **OCS Service Capacity** を選択します。



注記

ここで初期ストレージ容量を選択すると、この増分値でのみ容量を追加できます。

7. **Create** をクリックします。

Create ボタンは、3 つのノードを選択した後にのみ有効になります。3 つのボリュームからなる新規ストレージクラスターは、1 ワーカーノードごとに 1 つのボリュームを設定して作成されます。デフォルト設定では、レプリケーション係数 3 を使用します。

検証手順

- OpenShift Container Storage が正常にインストールされていることを確認するには、OpenShift Container Storage デプロイメントの検証 について参照してください。

1.3. RED HAT ENTERPRISE LINUX ベースのノード上のコンテナーでのファイルシステムアクセスの有効化

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) の Red Hat Enterprise Linux ベースに OpenShift Container Platform をデプロイしても、基礎となる Ceph ファイルシステムへのコンテナーアクセスは自動的に提供されません。

注記

このプロセスは、Red Hat Enterprise Linux CoreOS をベースとするホストには不要です。

手順

クラスター内の各ノードで以下の手順を実行します。

1. Red Hat Enterprise Linux ベースのノードにログインし、ターミナルを開きます。
2. ノードが rhel-7-server-extras-rpms リポジトリーにアクセスできることを確認します。

```
# subscription-manager repos --list-enabled | grep rhel-7-server
```

出力に **rhel-7-server-rpms** と **rhel-7-server-extras-rpms** の両方が表示されない場合や出力がない場合は、以下のコマンドを実行して各リポジトリーを有効にします。

```
# subscription-manager repos --enable=rhel-7-server-rpms
# subscription-manager repos --enable=rhel-7-server-extras-rpms
```

3. 必要なパッケージをインストールします。

```
# yum install -y policycoreutils container-selinux
```

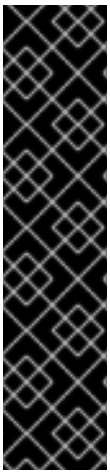
4. SELinux での Ceph ファイルシステムのコンテナーの使用を永続的に有効にします。

```
# setsebool -P container_use_cephfs on
```

5. コンテナーがこのノードでホストされる OpenShift Container Storage にアクセスできることを確認します。

第2章 ローカルストレージデバイスを使用した OPENSHIFT CONTAINER STORAGE のデプロイ

このセクションでは、OpenShift Container Platform がすでにインストールされているベアメタル、Amazon EC2 ストレージ最適化 I3、および VMware インフラストラクチャーに OpenShift Container Storage をデプロイします。



重要

ローカルストレージ Operator を使用した Amazon EC2 ストレージ最適化 I3 インスタンスへの OpenShift Container Storage のインストール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。Red Hat OpenShift Container Storage デプロイメントでは、3 つのワーカーノードでアプリケーションや他のワークロードが実行されていない状態の新規クラスターが想定されます。アプリケーションは追加のワーカーノードで実行する必要があります。

ローカルストレージデバイスを使用して OpenShift Container Storage をデプロイするには、以下の手順を実行します。

1. ローカルストレージデバイスを使用して OpenShift Container Storage をインストールするための要件を確認します。
2. Red Hat OpenShift Container Storage Operator をインストール します。
3. ローカルストレージ Operator をインストールします。
4. 利用可能なストレージデバイスを見つけます。
5. 要件に基づいて OpenShift Container Storage クラスターを作成します。
   - Amazon EC2 i3en.2xlarge インスタンスについては、Amazon EC2 ストレージ最適化 i3en.2xlarge インスタンスタイプでの OpenShift Container Storage クラスターの作成方法に従ってください。
   - VMware の場合は、VMware での OpenShift Container Storage クラスターの作成 方法に従ってください。
   - ベアメタルの場合は、ベアメタルでの OpenShift Container Storage クラスターの作成 方法に従ってください。

2.1. ローカルストレージデバイスを使用した OPENSHIFT CONTAINER STORAGE のインストール要件

- クラスターに、それぞれローカルで割り当てられたストレージデバイスを持つ OpenShift Container Platform ワーカーノードを 3 つ以上設定する必要があります。
  - 3 つのワーカーノードのそれぞれには、OpenShift Container Storage で使用できる raw ブロックデバイスが少なくとも 1 つ必要です。

- ノードの最小の初期要件は、『プランニング』の「ノード要件」のセクションを参照してください。
- 使用するデバイスは空である必要があります。つまり、ディスクに PV、VG、または LV は残さないでください。

- 3 つ以上のラベルが付けられたノードが必要です。
  - OpenShift Container Storage によって使用されるローカルストレージデバイスを持つ各ワーカーノードには、OpenShift Container Storage Pod をデプロイするための特定のラベルが必要です。ノードにラベルを付けるには、以下のコマンドを使用します。

```
$ oc label nodes <NodeName> cluster.ocs.openshift.io/openshift-storage=''
```

- ストレージプロバイダーはストレージノードでローカルにマウントされたストレージを管理することはできません。これが許可されると、Red Hat OpenShift Container Storage のローカルストレージ Operator の使用との競合が生じます。
- ローカルストレージ Operator のバージョンは、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ようにするために Red Hat OpenShift Container Platform のバージョンと一致している必要があります。ローカルストレージ Operator は、Red Hat OpenShift Container Platform のアップグレード時にアップグレードされません。

2.2. ローカルストレージ OPERATOR のインストール

以下の手順を使用して、Amazon EC2 I3、VMware、およびベアメタル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に OpenShift Container Storage クラスターを作成する前に、Operator Hub からローカルストレージ Operator をインストールします。

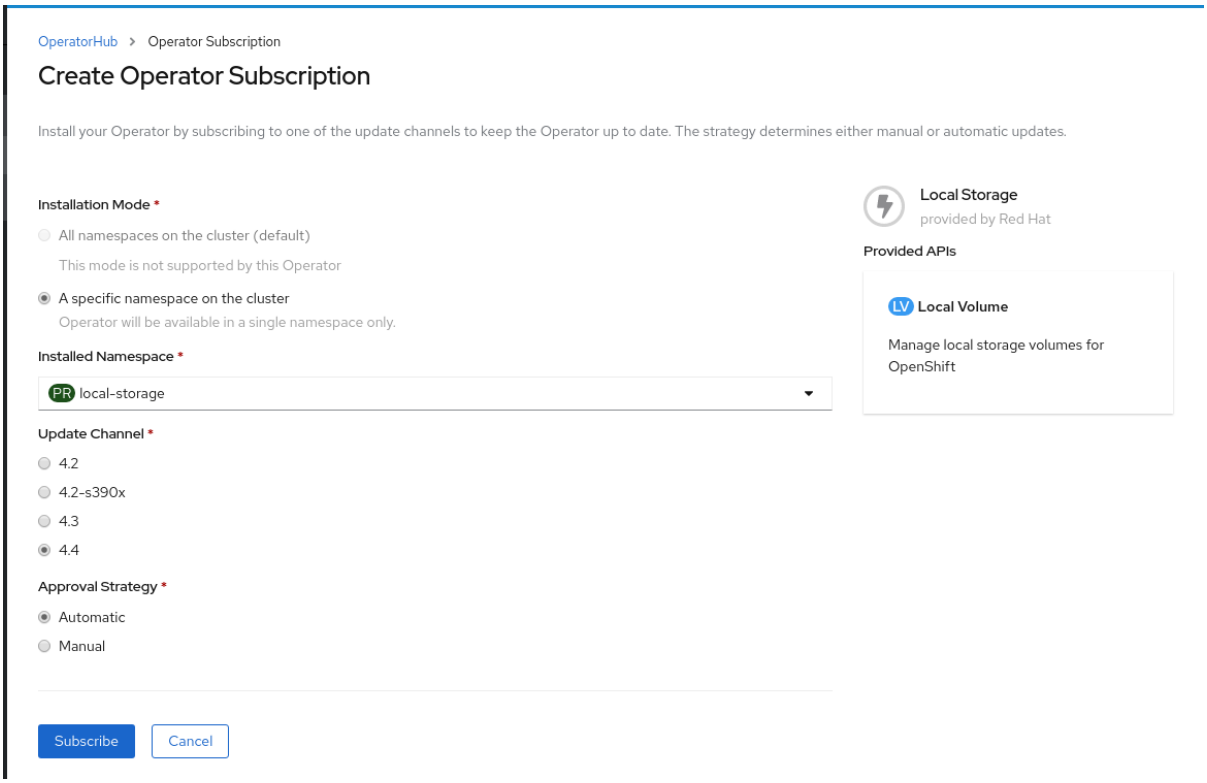
前提条件

- 以下のように、**openshift-storage** という namespace を作成します。
  a. OpenShift Web コンソールの左側のペインで、**Administration → Namespaces** をクリックします。
  b. **Create Namespace** をクリックします。
  c. Create Namespace ダイアログボックスで、Name に **local-storage** と入力します。
  d. **Default Network Policy** に **No restrictions** オプションを選択します。
  e. **Create** をクリックします。

手順

1. OpenShift Web コンソールの左側のペインで、**Operators → OperatorHub** をクリックします。
2. Operator の一覧から **Local Storage Operator** を検索し、これをクリックします。
3. **Install** をクリックします。

図2.1 Create Operator Subscription ページ



4. Installation Mode オプションに **A specific namespace on the cluster**を選択します。
   - ドロップダウンメニューから **local-storage** namespace を選択します。
5. **Update Channel** オプションの必要な値を選択します。
6. 必要な **Approval Strategy** を選択します。
7. **Subscribe** をクリックします。
8. ローカルストレージ Operator がステータス **Succeeded** を表示していることを確認します。

2.3. 利用可能なストレージデバイスの検索

以下の手順に従って、ベアメタル、Amazon EC2 I3、または VMware ストレージデバイス用に PV を作成する前に、OpenShift Container Storage ラベル **cluster.ocs.openshift.io/openshift-storage="** が付いた 3 つ以上のワーカーノードのそれぞれのデバイス名を特定します。

手順

1. OpenShift Container Storage ラベルの付いたワーカーノードの名前の一覧を表示し、確認します。

```
$ oc get nodes -l cluster.ocs.openshift.io/openshift-storage=
```

出力例:

```
NAME                                         STATUS   ROLES    AGE     VERSION
ip-10-0-135-71.us-east-2.compute.internal    Ready    worker   6h45m   v1.16.2
ip-10-0-145-125.us-east-2.compute.internal   Ready    worker   6h45m   v1.16.2
ip-10-0-160-91.us-east-2.compute.internal    Ready    worker   6h45m   v1.16.2
```

2. OpenShift Container Storage リソースに使用される各ワーカーノードにログインし、利用可能な各 raw ブロックデバイスの一意の **by-id** デバイス名を見つけます。

```
$ oc debug node/<Nodename>
```

出力例:

```
$ oc debug node/ip-10-0-135-71.us-east-2.compute.internal
Starting pod/ip-10-0-135-71us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Pod IP: 10.0.135.71
If you don't see a command prompt, try pressing enter.
sh-4.2# chroot /host
sh-4.4# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
nvme1n1                   259:0    0   2.3T  0 disk
nvme2n1                   259:1    0   2.3T  0 disk
nvme0n1                   259:2    0   120G  0 disk
|-nvme0n1p1                259:3    0   384M  0 part /boot
|-nvme0n1p2                259:4    0   127M  0 part /boot/efi
|-nvme0n1p3                259:5    0     1M  0 part
`-nvme0n1p4                259:6    0 119.5G  0 part
  `-coreos-luks-root-nocrypt 253:0    0 119.5G  0 dm   /sysroot
```

この例では、利用可能なローカルデバイスは **nvme0n1** および **nvme1n1** です。

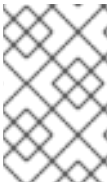
3. 各デバイスのハードウェアシリアル番号に応じて、一意の **by-id** デバイス名を見つけます。

```
sh-4.4#  ls -l /dev/disk/by-id/ | grep Storage
lrwxrwxrwx. 1 root root 13 Jun 26 07:29 nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS1924C57D4F1FC5236 -> ../../nvme2n1
lrwxrwxrwx. 1 root root 13 Jun 26 07:29 nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS22ABDB45A3BC9028F -> ../../nvme1n1
```

この例では、利用可能なローカルデバイスは **nvme0n1** および **nvme1n1** (サイズは 2.3 TiB) です。

OpenShift Container Storage ラベルが付けられた各ワーカーノード（最小 3 つ）については、固有の **by-id** デバイス名を見つける必要があります。この例では、**by-id** デバイス名は以下のようになります。

- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441494EC**
- **nvme-Amazon_EC2_NVMe_Instance_Storage_AWS60382E5D7441494EC**



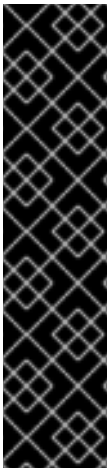
注記

OpenShift Container Storage で使用されるストレージデバイスを持つその他のすべてのノードについて、デバイス名 **by-id** の検索を繰り返し実行する必要があります。詳細は、https://access.redhat.com/solutions/4928841 を参照してください。

2.4. AMAZON EC2 ストレージ最適化 I3EN.2XLARGE インスタンスタイプでの OPENSHIFT CONTAINER STORAGE クラスターの作成

以下の手順を使用して、Amazon EC2（ストレージ最適化 i3en.2xlarge インスタンスタイプ）インフラストラクチャーに OpenShift Container Storage クラスターを作成します。これには、以下が含まれます。

1. **LocalVolume** CR の使用による PV の作成
2. 新規 **StorageClass** の作成



重要

ローカルストレージ Operator を使用した Amazon EC2 ストレージ最適化 i3en.2xlarge インスタンスへの OpenShift Container Storage のインストール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。Red Hat OpenShift Container Storage デプロイメントでは、3 つのワーカーノードでアプリケーションや他のワークロードが実行されていない状態の新規クラスターが想定されます。アプリケーションは追加のワーカーノードで実行する必要があります。

Amazon EC2 ストレージ最適化 i3en.2xlarge インスタンスタイプには、2 つの NVMe (non-volatile memory express) ディスクが含まれます。この手順の例では、このインスタンスタイプと共に提供される 2 つのディスクの使用方法について説明します。

警告

OpenShift Container Storage 永続データには、Amazon EC2 I3 の一時ストレージを使用することは推奨されません。3 つのノードすべてを停止するとデータが損失する可能性があるためです。一時ストレージは、以下のようなシナリオでのみ使用することが推奨されます。

- 特定のデータ処理 (data crunching) のためにデータがある場所からコピーされるクラウドバースト (時間に制限がある) が想定される場合。
- 開発環境またはテスト環境が想定される場合。

Amazon EC2 I3 の一時ストレージを使用している場合は、以下を行うことが推奨されます。

- 3 つのアベイラビリティーゾーンすべてを使用し、すべてのデータを失うリスクを軽減する。
- **ec2:StopInstances** パーミッションを持つユーザーの数を制限し、インスタンスを誤ってシャットダウンすることを回避する。

前提条件

- ローカルストレージデバイスを使用した OpenShift Container Storage のインストールの要件 に記載されるすべての要件を満たしていることを確認します。

- OpenShift Container Platform ワーカーノードに **nodeSelector** として使用される OpenShift Container Storage のラベルが付けられていることを確認します。

```
oc get nodes -l cluster.ocs.openshift.io/openshift-storage -o jsonpath='{range .items[*]}
{.metadata.name}{"\n"}'
```

出力例:

```
ip-10-0-135-71.us-east-2.compute.internal
ip-10-0-145-125.us-east-2.compute.internal
ip-10-0-160-91.us-east-2.compute.internal
```

手順

1. **LocalVolume** カスタムリソース (CR) を使用してストレージノードにローカル永続ボリューム (PV) を作成します。
OpenShift Storage Container ラベルをノードセレクターおよび **by-id** デバイス識別子として使用する **LocalVolume** CR **local-storage-block.yaml** の例

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: local-block
  namespace: local-storage
  labels:
    app: ocs-storagecluster
spec:
  nodeSelector:
    nodeSelectorTerms:
      - matchExpressions:
          - key: cluster.ocs.openshift.io/openshift-storage
            operator: In
            values:
              - ''
  storageClassDevices:
    - storageClassName: localblock
      volumeMode: Block
      devicePaths:
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441494EC   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84FE3E9   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE4   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS10382E5D7441464EP   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS1F45C01D7E84F43E7   # <-- modify this line
        - /dev/disk/by-id/nvme-
Amazon_EC2_NVMe_Instance_Storage_AWS136BC945B4ECB9AE8   # <-- modify this line
```

各 Amazon EC2 i3en.2xlarge インスタンスには 2 つのディスクがあり、この例では両方のディスクを使用します。

2. **LocalVolume** CR を作成します。

```
$ oc create -f local-storage-block.yaml
```

出力例:

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

3. Pod が作成されているかどうかを確認します。

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

出力例:

```
NAME                                  READY   STATUS    RESTARTS   AGE
local-block-local-diskmaker-59rmn     1/1     Running   0          15m
local-block-local-diskmaker-6n7ct     1/1     Running   0          15m
local-block-local-diskmaker-jwtsn     1/1     Running   0          15m
local-block-local-provisioner-6ssxc   1/1     Running   0          15m
local-block-local-provisioner-swwvx   1/1     Running   0          15m
local-block-local-provisioner-zmv5j   1/1     Running   0          15m
local-storage-operator-7848bbd595-686dg   1/1     Running   0          15m
```

4. PV が作成されているかどうかを確認します。
3 つのワーカーノード上の各ローカルストレージデバイスの新規 PV が表示される必要があります。利用可能なストレージデバイスの検索についてのセクションの例を参照してください。このセクションには、サイズが 2328Gi のワーカーノードごとに利用可能な 2 つのストレージデバイスが示されています。

```
$ oc get pv
```

出力例:

```
NAME                CAPACITY ACCESS MODES  RECLAIM POLICY STATUS      CLAIM
STORAGECLASS  REASON   AGE
local-pv-1a46bc79  2328Gi   RWO          Delete         Available           localblock           14m
local-pv-429d90ee  2328Gi   RWO          Delete         Available           localblock
14m
local-pv-4d0a62e3  2328Gi   RWO          Delete         Available           localblock
14m
local-pv-55c05d76  2328Gi   RWO          Delete         Available           localblock           14m
local-pv-5c7b0990  2328Gi   RWO          Delete         Available           localblock           14m
local-pv-a6b283b   2328Gi   RWO          Delete         Available           localblock           14m
```

5. **LocalVolume** CR の作成により新規の **StorageClass** が作成されているかどうかを確認します。この **StorageClass** は、PVC を作成するために **StorageCluster** を作成する間に使用されます。

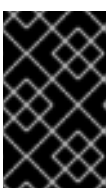
```
$ oc get sc | grep localblock
```

出力例:

```
NAME          PROVISIONER                    AGE
localblock    kubernetes.io/no-provisioner   7m46s
```

6. **localblock** StorageClass および作成される PV を使用する **StorageCluster** CR を作成します。**monDataDirHostPath** および**localblock** StorageClass を使用した **StorageCluster** CR **ocs-cluster-service.yaml** の例。

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
metadata:
  name: ocs-storagecluster
  namespace: openshift-storage
spec:
  manageNodes: false
  resources:
    mds:
      limits:
        cpu: 3
      requests:
        cpu: 1
    noobaa-core:
      limits:
        cpu: 2
        memory: 8Gi
      requests:
        cpu: 1
        memory: 8Gi
    noobaa-db:
      limits:
        cpu: 2
        memory: 8Gi
      requests:
        cpu: 1
        memory: 8Gi
  monDataDirHostPath: /var/lib/rook
  storageDeviceSets:
    - count: 2
      dataPVCTemplate:
        spec:
          accessModes:
            - ReadWriteOnce
          resources:
            requests:
              storage: 2328Gi
          storageClassName: localblock
          volumeMode: Block
      name: ocs-deviceset
      placement: {}
      portable: false
      replica: 3
      resources: {}
```



重要

OSD でノード間で保証されるサイズを確保するには、**storageDeviceSets** のストレージサイズを、ノード上に作成されている必要な PV のサイズ以下で指定する必要があります。

7. **StorageCluster** CR を作成します。

```
$ oc create -f ocs-cluster-service.yaml
```

出力例

```
storagecluster.ocs.openshift.io/ocs-cluster-service created
```

検証手順

OpenShift Container Storage デプロイメントの検証 について参照してください。

2.5. VMWARE での OPENSHIFT CONTAINER STORAGE クラスターの作成

以下の手順を使用して、VMware インフラストラクチャーにストレージクラスターを作成します。

VMware は、以下の 3 つのタイプのローカルストレージをサポートします。

- 仮想マシンディスク (VMDK)
- raw デバイスマッピング (RDM)
- VMDirectPath I/O

前提条件

- ローカルストレージデバイスを使用した OpenShift Container Storage のインストールの要件 に記載されるすべての要件を満たしていることを確認します。
- VMware でローカルストレージデバイスを使用するには、同じストレージタイプと同じサイズが割り当てられた 3 つのワーカーノードが必要です。
- OpenShift Container Platform ワーカーノードに OpenShift Container Storage のラベルが付けられていることを確認します。

```
oc get nodes -l cluster.ocs.openshift.io/openshift-storage -o jsonpath='{range .items[*]}
{.metadata.name}{"\n"}'
```

各ノードでストレージデバイスを特定するには、利用可能なストレージデバイスの検索について参照してください。

手順

1. ブロック PV の LocalVolume CR を作成します。
OpenShift Container Storage ラベルをノードセレクターとして使用する **LocalVolume** CR **local-storage-block.yaml** の例:

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: local-block
  namespace: local-storage
  labels:
```

```
    app: ocs-storagecluster
spec:
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
        - key: cluster.ocs.openshift.io/openshift-storage
          operator: In
          values:
          - ""
  storageClassDevices:
    - storageClassName: localblock
      volumeMode: Block
      devicePaths:
        - /dev/disk/by-id/scsi-36000c2991c27c2e5ba7c47d1e4352de2   # <-- modify this line
        - /dev/disk/by-id/scsi-36000c29682ca9e347926406711f3dc4e   # <-- modify this line
        - /dev/disk/by-id/scsi-36000c296aaf03a9b1e4b01d086bc6348   # <-- modify this line
```

2. ブロック PV の **LocalVolume** CR を作成します。

```
$ oc create -f local-storage-block.yaml
```

出力例:

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

3. Pod が作成されているかどうかを確認します。

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

出力例:

```
NAME                                  READY   STATUS    RESTARTS   AGE
local-block-local-diskmaker-5brzv         1/1     Running   0          31s
local-block-local-diskmaker-8sxcs         1/1     Running   0          31s
local-block-local-diskmaker-s7s9p         1/1     Running   0          31s
local-block-local-provisioner-9cbw8       1/1     Running   0          31s
local-block-local-provisioner-cpddv       1/1     Running   0          31s
local-block-local-provisioner-f6h7h       1/1     Running   0          31s
local-storage-operator-75b9776b75-vwdzh   1/1     Running   0          2m47s
```

4. 新規の **localblock** StorageClass を確認します。

```
$ oc get sc | grep localblock
```

出力例:

```
NAME          PROVISIONER                    AGE
localblock    kubernetes.io/no-provisioner   8m38s
```

5. **Available** のステータスで作成されている PV を確認します。

```
$ oc get pv
```

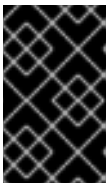
出力例:

```
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
local-pv-150fdc87   2TiB       RWO            Delete           Available   localblock
2m11s
local-pv-183bfc0a   2TiB       RWO            Delete           Available   localblock
2m11s
local-pv-b2f5cb25   2TiB       RWO            Delete           Available   localblock
2m21s
```

この例では、3 つの PV が OSD ストレージに使用されています。

6. **monDataDirHostPath** および **localblock** StorageClass を使用する **StorageCluster** CR **ocs-cluster-service-VMware.yaml** を作成します。

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
metadata:
  name: ocs-storagecluster
  namespace: openshift-storage
spec:
  manageNodes: false
  monDataDirHostPath: /var/lib/rook
  storageDeviceSets:
  - count: 1
    dataPVCTemplate:
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 2Ti
        storageClassName: localblock
        volumeMode: Block
    name: ocs-deviceset
    placement: {}
    portable: false
    replica: 3
    resources: {}
```



重要

OSD でノード間で保証されるサイズを確保するには、**storageDeviceSets** のストレージサイズを、ノード上に作成されている必要な PV のサイズ以下で指定する必要があります。

7. **StorageCluster** CR を作成します。

```
$ oc create -f ocs-cluster-service-VMware.yaml
```

出力例:

```
storagecluster.ocs.openshift.io/ocs-storagecluster created
```

検証手順

OpenShift Container Storage デプロイメントの検証 について参照してください。

2.6. ベアメタルでの OPENSHIFT CONTAINER STORAGE クラスターの作成

前提条件

- ローカルストレージデバイスを使用した OpenShift Container Storage のインストールの要件 に記載されるすべての要件を満たしていることを確認します。
- ベアメタルでローカルストレージデバイスを使用するには、各ノードに同じストレージタイプとサイズが割り当てられている 3 つのワーカーノードが必要です (例: 2TB NVMe ハードドライブ)。
- OpenShift Container Platform ワーカーノードに OpenShift Container Storage のラベルが付けられていることを確認します。

```
oc get nodes -l cluster.ocs.openshift.io/openshift-storage -o jsonpath='{range .items[*]}
{.metadata.name}{"\n"}'
```

各ノードでストレージデバイスを特定するには、利用可能なストレージデバイスの検索について参照してください。

手順

1. ブロック PV の **LocalVolume** CR を作成します。
OCS ラベルをノードセレクターとして使用する **LocalVolume** CR **local-storage-block.yaml** の例。

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: local-block
  namespace: local-storage
  labels:
    app: ocs-storagecluster
spec:
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
        - key: cluster.ocs.openshift.io/openshift-storage
          operator: In
          values:
          - ""
  storageClassDevices:
    - storageClassName: localblock
      volumeMode: Block
      devicePaths:
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY81260978128A   # <-- modify
this line
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY80440W5U128A   # <-- modify
this line
```

```
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYB85AABDE128A   # <-- modify
this line
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY0A60CB81128A   # <-- modify
this line
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPY0093D45E128A   # <-- modify
this line
        - /dev/disk/by-id/nvme-INTEL_SSDPEKKA128G7_BTPYE46F6060128A   # <-- modify
this line
```

2. ブロック PV の **LocalVolume** CR を作成します。

```
$ oc create -f local-storage-block.yaml
```

3. Pod が作成されているかどうかを確認します。

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

4. PV が作成されているかどうかを確認します。

```
$ oc get pv
```

出力例:

```
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGECLASS   REASON   AGE
local-pv-150fdc87   2Ti      RWO            Delete           Available           localblock
2m11s
local-pv-183bfc0a   2Ti      RWO            Delete           Available           localblock
2m11s
local-pv-b2f5cb25   2Ti      RWO            Delete           Available           localblock
2m21s
```

5. 新規 **localblock StorageClass** を確認します。

```
oc get sc | grep localblock
```

出力例:

```
NAME          PROVISIONER                    AGE
localblock    kubernetes.io/no-provisioner   10m20s
```

6. **localblock** StorageClass と作成された 3 つの PV を使用する **StorageCluster** CR を作成します。

7. **monDataDirHostPath** および **localblock** StorageClass を使用した **StorageCluster** CR **cluster-service-metal.yaml** の例。

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
metadata:
  name: ocs-storagecluster
  namespace: openshift-storage
spec:
```

```
manageNodes: false
monDataDirHostPath: /var/lib/rook
storageDeviceSets:
- count: 1
  dataPVCTemplate:
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 2Ti
      storageClassName: localblock
      volumeMode: Block
  name: ocs-deviceset
  placement: {}
  portable: false
  replica: 3
  resources: {}
```

重要

OSD にノード間で保証されたサイズを確保するには、**storageDeviceSets** のストレージサイズを、ノード上に作成された必要な PV のサイズ以下に指定する必要があります。

8. **StorageCluster** CR を作成します。

```
$ oc create -f cluster-service-metal.yaml
```

出力例:

```
storagecluster.ocs.openshift.io/ocs-storagecluster created
```

検証手順

OpenShift Container Storage デプロイメントの検証 について参照してください。

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

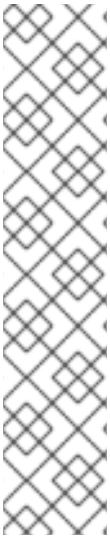
このセクションを使用して、OpenShift Container Storage が正常にデプロイされていることを確認します。

3.1. POD の状態の確認

OpenShift Container ストレージが正常にデプロイされているかどうかを確認するために、Pod が **running** 状態にあることを確認できます。

手順

1. OpenShift Web コンソールの左側のペインから **Workloads → Pods** をクリックします。
2. **Project** ドロップダウンリストから **openshift-storage** を選択します。
   Pod 数は、OpenShift Container Platform にデプロイされる OSD およびワーカーノードの数によって異なります。OSD の数は、**StorageCluster** の各 **StorageDeviceSet** に対して定義される **Count** と **Replica** によって異なります。コンポーネントごとの Pod 数は、ワーカーノード数だけでなく OSD にも直接関連付けられます。



注記

OpenShift Container Storage のクラスター全体でのデフォルトノードセレクターを上書きする必要がある場合は、コマンドラインインターフェースで以下の手順を実行できます。

1. **openshift-storage** namespace の空のノードセレクターを指定します。

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

2. **DaemonSets** によって生成される元の Pod を削除します。

```
oc delete pod -l app=csi-cephfsplugin -n openshift-storage
oc delete pod -l app=csi-rbdplugin -n openshift-storage
```

1. **Running** および **Completed** タブをクリックして、以下の Pod が実行中および完了状態にあることを確認します。

表3.1 3 つのワーカーノードからなるクラスターのストレージコンポーネントに対応する Pod

コンポーネント	Pod の数	Pod の名前
以下のコンポーネントについて確認する必要のある Pod 数。		
OpenShift Container Storage Operator	1	**ocs-operator-***
Rook-ceph Operator	1	**rook-ceph-operator-***

コンポーネント	Pod の数	Pod の名前
Multicloud Object Gateway	4	• **noobaa-operator-*** • **noobaa-core-*** • **nooba-db-*** • **noobaa-endpoint-***
Mon	3	• **rook-ceph-mon-*** • **rook-ceph-mon-*** • **rook-ceph-mon-*** (異なるノード上)
rook-ceph-mgr	1	**rook-ceph-mgr-*** (ストレージノード上)
MDS	2	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** (異なるストレージノード上に 2 つの Pod)
lib-bucket-provisioner	1	**lib-bucket-provisioner--*** (任意のノード上)
CSI の Pod 数は、ストレージノードとして選択されるノード数 (最小 3 ノード) によって異なります。		

CSI	10	• **cephfs** (5 つ以上の Pod)
		  ◦ **csi-cephfsplugin-*** (ストレージが使用される各ノード上に 1 つの Pod 、つまり異なるノード上に 3 つの Pod)
		  ◦ **csi-cephfsplugin-provisioner-*** (利用可能な場合、異なるストレージノード上に 2 つの Pod)
		• **rbd** (合計 5 つ以上の Pod)
		  ◦ **csi-rbdplugin-*** (ストレージが使用されている各ノード上に 1 つの Pod、つまり異なるノード上に 3 つの Pod)
		  ◦ **csi-rbdplugin-provisioner-*** (利用可能な場合、異なるストレージノード上に 2 つの Pod)

rook-ceph-drain-canary	3	**rook-ceph-drain-canary-*** (3 つの Pod、つまり各ストレージノード上に 1 つの Pod)
rook-ceph-crashcollector	3	**rook-ceph-crashcollector-*** (3 つの Pod)

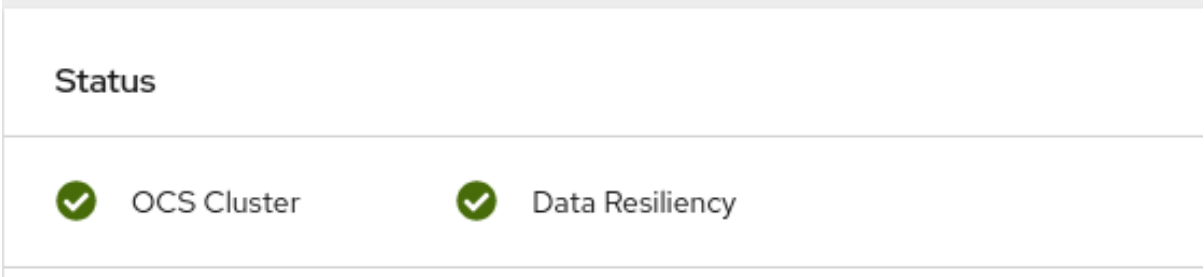
コンポーネント	Pod の数	Pod の名前
OSD の数は、StorageCluster の各 StorageDeviceSet に対して定義される **Count** と **Replica** によって異なります。		
OSD	6	• **rook-ceph-osd-*** (異なるノードに 3 つの Pod) • **rook-ceph-osd-prepare-ocs-deviceset-*** (3 つの Pod)

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

永続ストレージダッシュボードを使用して OpenShift Container Storage クラスターの正常性を確認できます。詳細は、『OpenShift Container Storage のモニタリング』を参照してください。

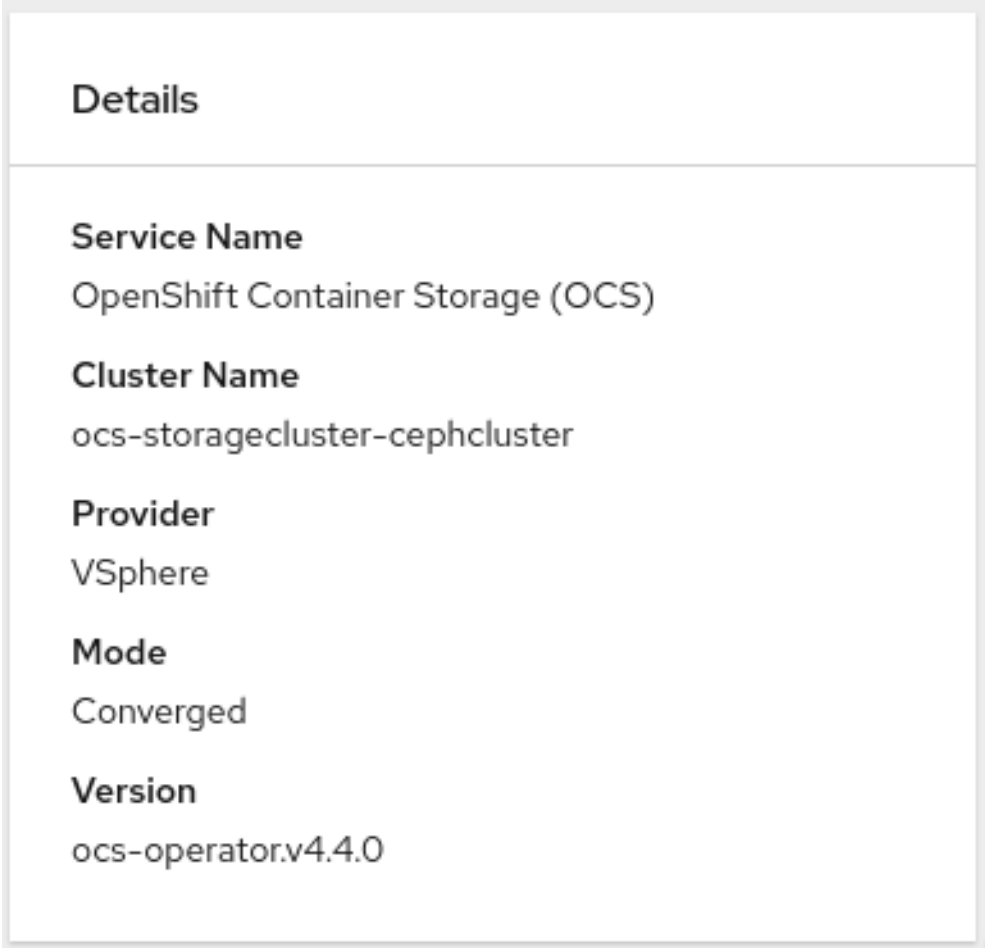
- OpenShift Web コンソールの左側のペインから **Home → Overview** をクリックし、**Persistent Storage** タブをクリックします。
  Status カード で、以下の画像のように **OCS Cluster** に緑色のチェックマークが表示されていることを確認します。

  図3.1 Persistent Storage Overview Dashboard の Health status カード



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

図3.2 Persistent Storage Overview Dashboard の Details カード

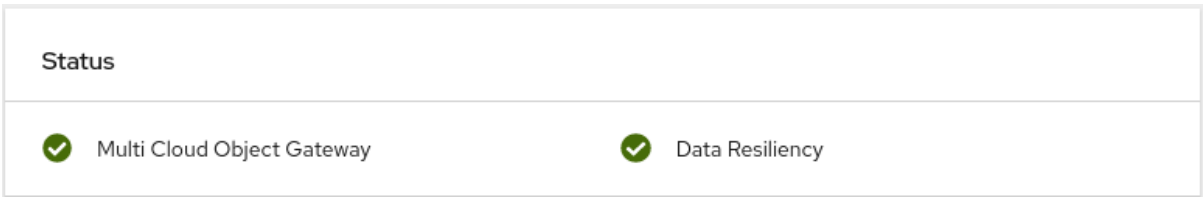


3.2.1. Multicloud Object Gateway が正常であることの確認

オブジェクトサービスダッシュボードを使用して、OpenShift Container Storage クラスターの正常性を確認できます。詳細は、『OpenShift Container Storage のモニタリング』を参照してください。

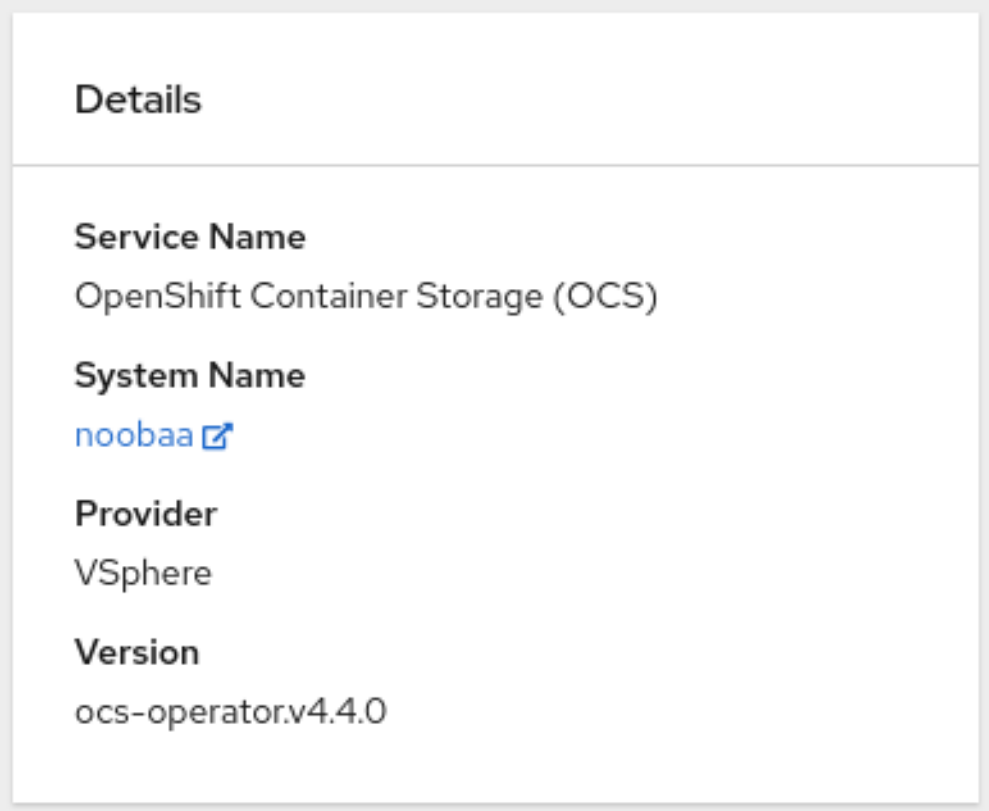
- OpenShift Web コンソールの左側のペインから **Home → Overview** をクリックし、**Object Service** タブをクリックします。
  Status カード で、以下のように Multicloud Object Gateway (MCG) ストレージに緑色のチェックマークが表示されていることを確認します。

  図3.3 Object Service Overview Dashboard の Health status カード

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

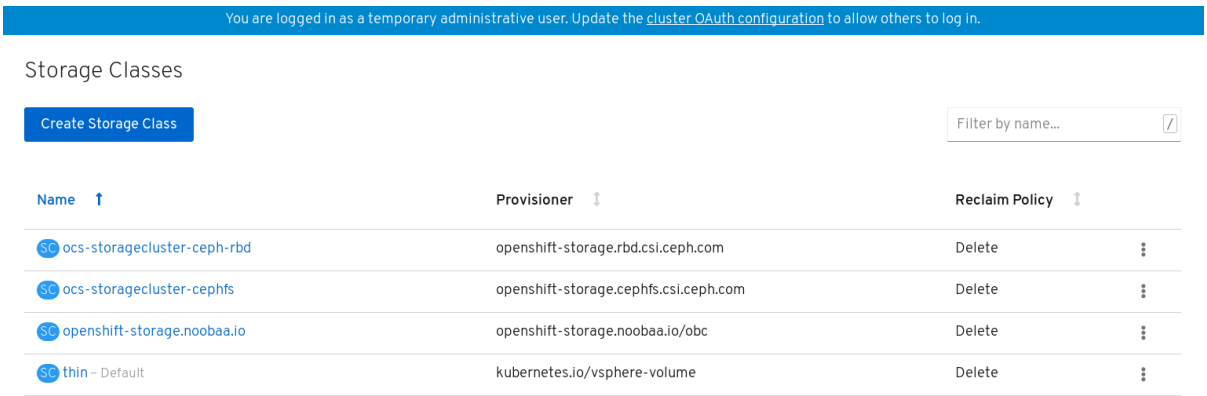
図3.4 Object Service Overview Dashboard の Details カード



3.2.2. ストレージクラスが作成され、一覧表示されることの確認

ストレージクラスが作成され、以下のように一覧表示されていることを確認できます。

- OpenShift Web コンソールの左側のペインから **Storage → Storage Classes**をクリックします。
  以下の 3 つのストレージクラスが OpenShift Container Storage クラスターの作成時に作成されることを確認します。
  - **ocs-storagecluster-ceph-rbd**
  - **ocs-storagecluster-cephfs**
  - **openshift-storage.noobaa.io**



第4章 OPENSHIFT CONTAINER PLATFORM のアンインストール

このセクションの手順を使用して、ユーザーインターフェースから **Uninstall** オプションを使用せずに OpenShift Container Storage をアンインストールします。

前提条件

- OpenShift Container Storage クラスターの状態が正常であることを確認します。一部の Pod がリソースまたはノードの不足により正常に終了しないと、削除に失敗する可能性があります。クラスターが状態が正常でない場合は、OpenShift Container Storage をアンインストールする前に Red Hat カスタマーサポートにお問い合わせください。
- OpenShift Container Storage ストレージクラスに基づいて Persistent Volume Claim（永続ボリューム要求、PVC）または Object Bucket Claim (オブジェクトバケット要求、OBC) を使用するアプリケーションを削除してから、OpenShift Container Storage ストレージクラスを使用している PVC および OBC を削除します。

手順

1. ストレージクラスを一覧表示し、以下のストレージクラスプロビジョナーのストレージクラスをメモします。
   - **openshift-storage.rbd.csi.ceph.com**
   - **openshift-storage.cephfs.csi.ceph.com**
   - **openshift-storage.noobaa.io/obc**

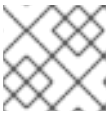
     以下は例になります。

     ```
     $ oc get storageclasses
     NAME                        PROVISIONER                             AGE
     gp2 (default)               kubernetes.io/aws-ebs                   23h
     ocs-storagecluster-ceph-rbd  openshift-storage.rbd.csi.ceph.com      22h
     ocs-storagecluster-cephfs    openshift-storage.cephfs.csi.ceph.com   22h
     openshift-storage.noobaa.io  openshift-storage.noobaa.io/obc         22h
     ```
2. 直前の手順に記載されているストレージクラスプロビジョナーを使用している PVC および OBC をクエリーします。

   ```
   $ oc get pvc -o=jsonpath='{range .items[?(@.spec.storageClassName=="ocs-storagecluster-ceph-rbd")]}{"Name: "}{@.metadata.name}{" Namespace: "}{@.metadata.namespace}{" Labels: "}{@.metadata.labels}{"\n"}{end}' --all-namespaces|awk '! ( /Namespace: openshift-storage/ && /app:noobaa/ )'
   ```

   ```
   $ oc get pvc -o=jsonpath='{range .items[?(@.spec.storageClassName=="ocs-storagecluster-cephfs")]}{"Name: "}{@.metadata.name}{" Namespace: "}{@.metadata.namespace}{"\n"}{end}' --all-namespaces
   ```

   ```
   $ oc get obc -o=jsonpath='{range .items[?(@.spec.storageClassName=="openshift-storage.noobaa.io")]}{"Name: "}{@.metadata.name}{" Namespace: "}{@.metadata.namespace}{"\n"}{end}' --all-namespaces
   ```



注記

openshift-storage namespace の NooBaa PVC を無視します。

3. 以下の手順に従って、直前の手順に記載されている PVC が削除されていることを確認します。

   a. PVC を使用する Pod を判別します。

   b. **Deployment**、**StatefulSet**、 **DeamonSet**、**Job**、またはカスタムコントローラーなどの制御する側のオブジェクトを特定します。
   各オブジェクトには、**OwnerReference** として知られるメタデータフィールドがあります。これは、関連付けられたオブジェクトの一覧です。**controller** フィールドが **true** に設定された **OwnerReference** は、 **ReplicaSet**、**StatefulSet**、 **DaemonSet** などの制御する側のオブジェクトをポイントします。

   c. プロジェクトの所有者を確認してからこれを削除し、オブジェクトを安全に削除できるようにします。

   d. PVC および OBC を削除します。

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

   モニタリングスタック、クラスターロギング Operator、または prometheus レジストリーの設定の一部として PVC を作成した場合は、必要に応じて以下のセクションで説明されているクリーンアップ手順を実行する必要があります。

   - 「OpenShift Container Storage からのモニタリングスタックの削除」
   - 「OpenShift Container Storage からの OpenShift Container Platform レジストリーの削除」
   - 「OpenShift Container Storage からのクラスターロギング Operator の削除」

4. バッキングローカルボリュームオブジェクトを一覧表示し、メモします。結果が見つからない場合は、手順 8 および 9 を省略します。

   ```
   $ for sc in $(oc get storageclass|grep 'kubernetes.io/no-provisioner' |grep -E $(oc get
   storagecluster -n openshift-storage -o jsonpath='{
   .items[*].spec.storageDeviceSets[*].dataPVCTemplate.spec.storageClassName}' | sed 's/
   /|/g')| awk '{ print $1 }');
   do
      echo -n "StorageClass: $sc ";
      oc get storageclass $sc -o jsonpath=" { 'LocalVolume: ' }{
   .metadata.labels['local\.storage\.openshift\.io/owner-name'] } { '\n' }";
   done
   ```

   ```
   StorageClass: localblock  LocalVolume: local-block
   ```

5. **StorageCluster** オブジェクトを削除します。

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

6. namespace を削除し、削除が完了するまで待機します。

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

注記

openshift-storage がアクティブなプロジェクトである場合、別のプロジェクトに切り換える必要があります。

以下は例になります。

```
$ oc project default
```

7. 各ノードのストレージ Operator アーティファクトをクリーンアップします。

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

出力に **removed directory /var/lib/rook** があることを確認します。

```
Starting pod/ip-10-0-134-65us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
removed '/var/lib/rook/openshift-storage/log/ocs-deviceset-2-0-gk22s/ceph-volume.log'
removed directory '/var/lib/rook/openshift-storage/log/ocs-deviceset-2-0-gk22s'
removed '/var/lib/rook/openshift-storage/log/ceph-osd.2.log'
removed '/var/lib/rook/openshift-storage/log/ceph-volume.log'
removed directory '/var/lib/rook/openshift-storage/log'
removed directory '/var/lib/rook/openshift-storage/crash/posted'
removed directory '/var/lib/rook/openshift-storage/crash'
removed '/var/lib/rook/openshift-storage/client.admin.keyring'
removed '/var/lib/rook/openshift-storage/openshift-storage.config'
removed directory '/var/lib/rook/openshift-storage'
removed '/var/lib/rook/osd2/openshift-storage.config'
removed directory '/var/lib/rook/osd2'
removed directory '/var/lib/rook'

Removing debug pod ...
Starting pod/ip-10-0-155-149us-east-2computeinternal-debug ...
.
.
removed directory '/var/lib/rook'

Removing debug pod ...
Starting pod/ip-10-0-162-89us-east-2computeinternal-debug ...
.
.
removed directory '/var/lib/rook'

Removing debug pod ...
```

8. デプロイメント時に作成したローカルボリュームを作成し、手順 4 で一覧表示されているそれぞれのローカルボリュームについてこれを実行します。
ローカルボリュームごとに、以下を実行します。

   a. 変数 **LV** を LocalVolume の名前に設定し、変数 **SC** を StorageClass の名前に設定します。

以下は例になります。

```
$ LV=local-block
$ SC=localblock
```

b. 後でクリーンアップするデバイスを一覧表示して書き留めておきます。

```
$ oc get localvolume -n local-storage $LV -o jsonpath='{
.spec.storageClassDevices[*].devicePaths[*] }'
```

```
/dev/disk/by-id/nvme-Amazon_Elastic_Block_Store_vol078f5cdde09efc165 /dev/disk/by-
id/nvme-Amazon_Elastic_Block_Store_vol0defc1d5e2dd07f9e /dev/disk/by-id/nvme-
Amazon_Elastic_Block_Store_vol0c8e82a3beeb7b7e5
```

c. ローカルボリュームリソースを削除します。

```
$ oc delete localvolume -n local-storage --wait=true $LV
```

d. 残りの PV および StorageClass が存在する場合は削除します。

```
$ oc delete pv -l storage.openshift.com/local-volume-owner-name=${LV} --wait --
timeout=5m
$ oc delete storageclass $SC --wait --timeout=5m
```

e. そのリソースのストレージノードからアーティファクトをクリーンアップします。

```
$ [[ ! -z $SC ]] && for i in $(oc get node -l cluster.ocs.openshift.io/openshift-storage= -o
jsonpath='{ .items[*].metadata.name }'); do oc debug node/${i} -- chroot /host rm -rfv
/mnt/local-storage/${SC}/; done
```

```
Starting pod/ip-10-0-141-2us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
removed '/mnt/local-storage/localblock/nvme2n1'
removed directory '/mnt/local-storage/localblock'

Removing debug pod ...
Starting pod/ip-10-0-144-55us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
removed '/mnt/local-storage/localblock/nvme2n1'
removed directory '/mnt/local-storage/localblock'

Removing debug pod ...
Starting pod/ip-10-0-175-34us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
removed '/mnt/local-storage/localblock/nvme2n1'
removed directory '/mnt/local-storage/localblock'

Removing debug pod ...
```

9. 手順 4 で一覧表示されている各ローカルボリュームのディスクを消去し、それらを再利用できるようにします。

a. ストレージノードを一覧表示します。

```
oc get nodes -l cluster.ocs.openshift.io/openshift-storage=
```

```
NAME                                         STATUS   ROLES    AGE     VERSION
ip-10-0-134-65.us-east-2.compute.internal    Ready    worker   4h45m   v1.17.1
ip-10-0-155-149.us-east-2.compute.internal   Ready    worker   4h46m   v1.17.1
ip-10-0-162-89.us-east-2.compute.internal    Ready    worker   4h45m   v1.17.1
```

b. プロンプトが表示されたら、ノードコンソールを取得して、**chroot /host** コマンドを実行します。

```
$ oc debug node/ip-10-0-134-65.us-east-2.compute.internal
Starting pod/ip-10-0-134-65us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Pod IP: 10.0.134.65
If you don't see a command prompt, try pressing enter.
sh-4.2# chroot /host
```

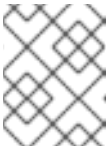
c. 手順 8 (ii) で収集したディスクパスを引用符内の**DISKS** 変数に保存します。

```
sh-4.2# DISKS="/dev/disk/by-id/nvme-
Amazon_Elastic_Block_Store_vol078f5cdde09efc165 /dev/disk/by-id/nvme-
Amazon_Elasti_Block_Store_vol0defc1d5e2dd07f9e /dev/disk/by-id/nvme-
Amazon_Elastic_Block_Store_vol0c8e82a3beeb7b7e5"
```

d. すべてのディスクで **sgdisk --zap-all** を実行します。

```
sh-4.4# for disk in $DISKS; do sgdisk --zap-all $disk;done
```

```
Problem opening /dev/disk/by-id/nvme-
Amazon_Elastic_Block_Store_vol078f5cdde09efc165 for reading! Error is 2.
The specified file does not exist!
Problem opening '' for writing! Program will now terminate.
Warning! MBR not overwritten! Error is 2!
Problem opening /dev/disk/by-id/nvme-
Amazon_Elasti_Block_Store_vol0defc1d5e2dd07f9e for reading! Error is 2.
The specified file does not exist!
Problem opening '' for writing! Program will now terminate.
Warning! MBR not overwritten! Error is 2!
Creating new GPT entries.
GPT data structures destroyed! You may now partition the disk using fdisk or
other utilities.
```



注記

file-not-found の警告は他のマシンにあるディスクを参照するため、この警告は無視してください。

e. シェルを終了して、他のノードで繰り返します。

```
sh-4.4# exit
exit
```

```
sh-4.2# exit
exit

Removing debug pod ...
```

10. 手順 1 で一覧表示されている **openshift-storage** プロビジョナーの設定されているストレージクラスを削除します。

```
$ oc delete storageclass <storageclass-name> --wait=true --timeout=5m
```

以下は例になります。

```
$ oc delete storageclass ocs-storagecluster-ceph-rbd ocs-storagecluster-cephfs openshift-storage.noobaa.io --wait=true --timeout=5m
```

11. ストレージノードのラベルを解除します。

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

注記

label <label> not found のようなラベルが解除されているノードについて表示される警告は無視できます。

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

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

13. OpenShift Container Storage がアンインストールされていることを確認するには、openshift-storage namespace がすでに存在しておらず、ストレージダッシュボードが UI に表示されないことを確認します。

注記

OpenShift Container Storage のアンインストール時に、namespace が完全に削除されず、**Terminating** 状態のままである場合は、https://access.redhat.com/solutions/3881901 の記事を参照して namespace の終了をブロックしているオブジェクトを特定します。**Cephcluster**、**StorageCluster**、**NooBaa**、および **PVC** などの ファイナライザーがある OpenShift オブジェクトにより、namespace が **Terminating** 状態になる可能性があります。PVC にファイナライザーがある場合、関連付けられた Pod を削除を強制的に実行してファイナライザーを削除します。

4.1. OPENSHIFT CONTAINER STORAGE からのモニタリングスタックの削除

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

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

前提条件

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

手順

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

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

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

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

2. モニタリング **configmap** を編集します。

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

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

Before editing

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



.
.
.
```

After editing

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

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

4. PVC を使用する Pod を一覧表示します。
この例では、PVC を消費していた **alertmanagerMain** および **prometheusK8s** Pod は **Terminating** 状態にあります。これらの Pod が OpenShift Container Storage PVC を使用しなくなったら PVC を削除できます。

```
$ oc get pod,pvc -n openshift-monitoring
NAME                                          READY   STATUS      RESTARTS AGE
pod/alertmanager-main-0                       3/3     Terminating   0      10h
pod/alertmanager-main-1                       3/3     Terminating   0      10h
pod/alertmanager-main-2                       3/3     Terminating   0      10h
pod/cluster-monitoring-operator-84cd9df668-zhjfn   1/1     Running       0      18h
pod/grafana-5db6fd97f8-pmtbf                  2/2     Running       0      10h
pod/kube-state-metrics-895899678-z2r9q        3/3     Running       0      10h
pod/node-exporter-4njxv                       2/2     Running       0      18h
pod/node-exporter-b8ckz                       2/2     Running       0      11h
pod/node-exporter-c2vp5                       2/2     Running       0      18h
pod/node-exporter-cq65n                       2/2     Running       0      18h
pod/node-exporter-f5sm7                       2/2     Running       0      11h
pod/node-exporter-f852c                       2/2     Running       0      18h
pod/node-exporter-l9zn7                       2/2     Running       0      11h
pod/node-exporter-ngbs8                       2/2     Running       0      18h
pod/node-exporter-rv4v9                       2/2     Running       0      18h
pod/openshift-state-metrics-77d5f699d8-69q5x      3/3     Running       0      10h
pod/prometheus-adapter-765465b56-4tbxx        1/1     Running       0      10h
pod/prometheus-adapter-765465b56-s2qg2        1/1     Running       0      10h
pod/prometheus-k8s-0                          6/6     Terminating   1      9m47s
pod/prometheus-k8s-1                          6/6     Terminating   1      9m47s
pod/prometheus-operator-cbfd89f9-ldnwc        1/1     Running       0      43m
pod/telemeter-client-7b5ddb4489-2xfpz         3/3     Running       0      10h

NAME                                        STATUS  VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS               AGE
```

```
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-0   Bound    pvc-2eb79797-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO            ocs-storagecluster-ceph-rbd   19h
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-1   Bound    pvc-2ebeee54-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO            ocs-storagecluster-ceph-rbd   19h
persistentvolumeclaim/ocs-alertmanager-claim-alertmanager-main-2   Bound    pvc-2ec6a9cf-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO            ocs-storagecluster-ceph-rbd   19h
persistentvolumeclaim/ocs-prometheus-claim-prometheus-k8s-0        Bound    pvc-3162a80c-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO            ocs-storagecluster-ceph-rbd   19h
persistentvolumeclaim/ocs-prometheus-claim-prometheus-k8s-1        Bound    pvc-316e99e2-1fed-11ea-93e1-0a88476a6a64   40Gi       RWO            ocs-storagecluster-ceph-rbd   19h
```

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

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

4.2. OPENSHIFT CONTAINER STORAGE からの OPENSHIFT CONTAINER PLATFORM レジストリーの削除

このセクションでは、OpenShift Container Storage から OpenShift Container Platform レジストリーをクリーンアップします。代替ストレージを設定する必要がある場合は、https://access.redhat.com/documentation/en-us/openshift_container_platform/4.4/html-single/registry/architecture-component-imageregistry を参照してください。

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

前提条件

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

手順

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

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

- AWS の場合:

編集前

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

編集後

```
.
.
storage:
.
.
```

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

- VMware およびベアメタルの場合:

編集前

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

編集後

```
.
.
storage:
  emptyDir: {}
.
.
```

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

2. PVC を削除します。

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

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

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

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

前提条件

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

手順

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

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

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

2. PVC を削除します。

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