Red Hat OpenShift Container Storage 4.5

Red Hat Virtualization プラットフォームを使用した OpenShift Container Storage のデプロイおよび管理

インストールおよび管理方法

Last Updated: 2023-03-28

Red Hat OpenShift Container Storage 4.5 Red Hat Virtualization プラットフォームを使用した OpenShift Container Storage のデプロイおよび管理

インストールおよび管理方法

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

Red Hat Virtualization プラットフォームで Red Hat OpenShift Container Storage をインストールし、管理する方法については、本書をお読みください。 Deploying and managing OpenShift Container Storage on Red Hat Virtualization platform is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

目次

前書き

Red Hat OpenShift Container Storage 4.5 では、既存の Red Hat OpenShift Container Platform(OCP)Red Hat Virtualization プラットフォームクラスターでのデプロイメントをサポートします。

内部モードで OpenShift Container Storage をデプロイするには、Red Hat Virtualization での OpenShift Container Storage のデプロイのデプロイメントプロセスを実行します。

第1章 RED HAT VIRTUALIZATION での OPENSHIFT CONTAINER STORAGE のデプロイ

Red Hat Virtualization のインストーラーでプロビジョニングされるインフラストラクチャー (IPI) によって提供される共有ストレージデバイスを使用して OpenShift Container Storage を OpenShift Container Platform にデプロイすると、内部クラスターリソースを作成できます。

注記

Red Hat Virtualization では、内部の Openshift Container Storage クラスターのみがサポートされます。デプロイメント要件の詳細は、『Planning your deployment』を参照してください。

このセクションを使用して、OpenShift Container Platform がすでにインストールされている Red Hat Virtualization インフラストラクチャーに OpenShift Container Storage をデプロイします。

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

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. Red Hat Virtualization で OpenShift Container Storage クラスターサービスを作成 します。

1.1. 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

- 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。
  - ノードの最小要件については、『プランニング』ガイドの「リソース要件」セクションを参照してください。
  - 使用するデバイスは空である必要があります。つまり、ディスクには PV、VG、または LV がない状態でなければなりません。
- 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 <NodeNames> cluster.ocs.openshift.io/openshift-storage=''
```

- Red Hat OpenShift Container Storage のローカルストレージ Operator の使用と競合する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ストレージプロバイダーは使用しないでください。

- 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 のアップグレード時にアップグレードされません。

1.2. RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR のインストール

Red Hat OpenShift Container Storage は、Red Hat OpenShift Container Platform Operator Hub を使用してインストールできます。ハードウェアおよびソフトウェアの要件に関する詳細は、『デプロイメントのプランニング』を参照してください。

前提条件

- OpenShift Container Platform クラスターにログインしている必要がある。

- 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。

注記

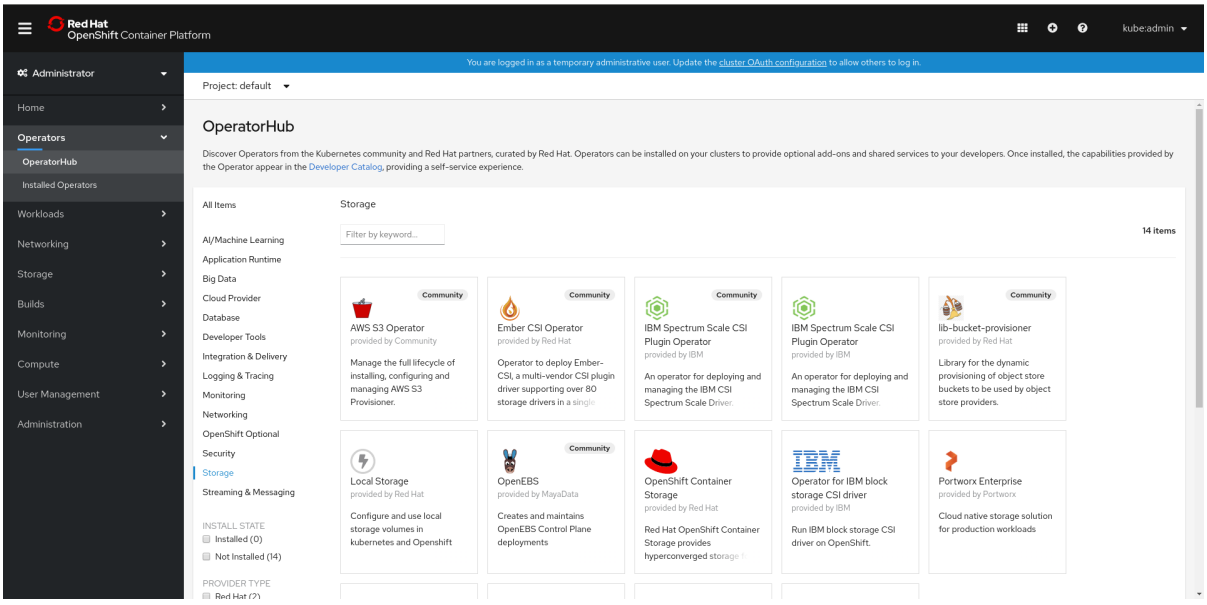
OpenShift Container Storage のクラスター全体でのデフォルトノードセレクターを上書きする必要がある場合は、コマンドラインインターフェースで以下のコマンドを使用し、**openshift-storage** namespace の空のノードセレクターを指定できます。

```
$ 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. **OpenShift Container Storage** をクリックします。
Filter by keyword テキストボックスまたはフィルター一覧を使用して、Operator の一覧から OpenShift Container Storage を検索できます。

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

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

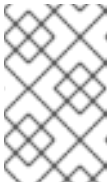
   a. Channel を **stable-4.5**として更新します。

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

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

   d. **承認ストラテジー** を **Automatic** または **Manual** として選択している。承認ストラテジーはデフォルトで **Automatic** に設定されます。

   - **Approval Strategy** に **Automatic** を選択します。

     注記

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

     i. **インストール** をクリックします。

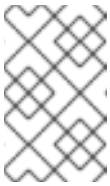
     ii. インストールが開始するまで待機します。これには、最長 20 分の時間がかかる可能性があります。

     iii. **Operators → Installed Operators**をクリックします。

     iv. **Project** が **openshift-storage** であることを確認します。デフォルトで、**プロジェクト** は **openshift-storage** です。

     v. **OpenShift Container Storage**の **Status** が **Succeeded** に変更するまで待機します。

   - **Approval Strategy** に **Manual** を選択します。

     注記

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

     i. **Install** をクリックします。

     ii. **Installed Operators** ページで、**ocs-operator** をクリックします。

     iii. **Subscription Details** ページで、**Install Plan** リンクをクリックします。

     iv. **InstallPlan Details** ページで、**Preview Install Plan** をクリックします。

v. インストール計画を確認し、**Approve** をクリックします。

vi. **Components** の **Status** が **Unknown** から **Created** または **Present** のいずれかに変更するまで待機します。

vii. **Operators → Installed Operators** をクリックします。

viii. **Project** が **openshift-storage** であることを確認します。デフォルトで、**プロジェクト** は **openshift-storage** です。

ix. **OpenShift Container Storage** の **Status** が **Succeeded** に変更するまで待機します。

検証手順

- OpenShift Container Storage Operator の Status が Installed Operators ダッシュボードで **Succeeded** と表示されることを確認します。

1.3. 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、OpenShift Container Storage クラスターをローカルストレージデバイスに作成する前に Operator Hub からローカルストレージ Operator をインストールします。

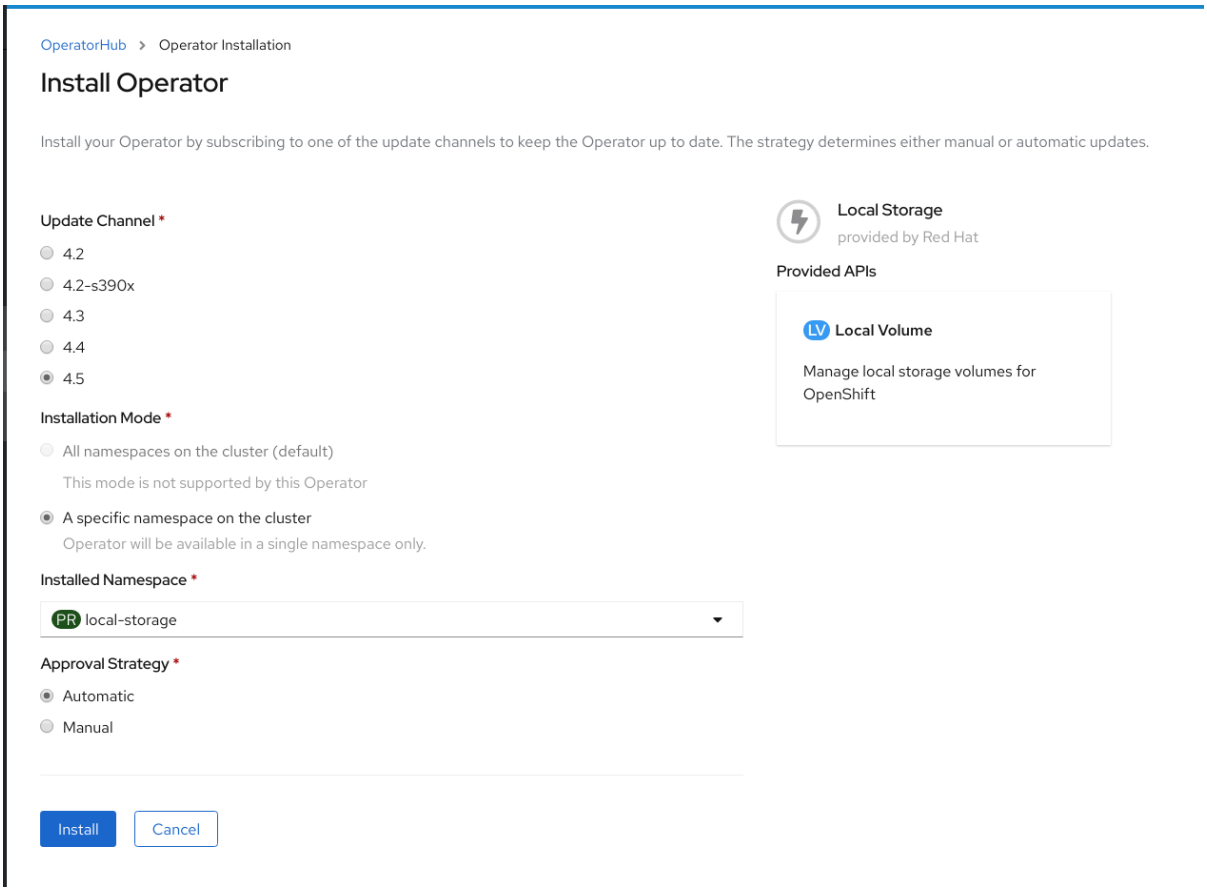
前提条件

- 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1.2 Install Operator ページ



4. **Install Operator** ページで、以下のオプションが選択されていることを確認します。
   a. Channel を **stable-4.5**として更新します。
   b. Installation Mode オプションに **A specific namespace on the cluster**を選択します。
   c. Installed Namespace を **local-storage** に選択します。
   d. Approval Strategy に **Automatic** を選択します。
5. **Install** をクリックします。
6. Local Storage Operator のステータスが **Succeeded** と表示されていることを確認します。

1.4. 利用可能なストレージデバイスの検索

以下の手順を使用して、Red Hat Virtualization の PV を作成する前に、OpenShift Container Storage ラベル **cluster.ocs.openshift.io/openshift-storage=''** でラベルを付けた 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
rhvworker01    Ready    worker   6h45m   v1.16.2
rhvworker02    Ready    worker   6h45m   v1.16.2
rhvworker03    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/rhvworker01
Starting pod/rhvworker01-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
xvda                         202:0    0   120G  0 disk
|-xvda1                       202:1    0   384M  0 part /boot
|-xvda2                       202:2    0   127M  0 part /boot/efi
|-xvda3                       202:3    0     1M  0 part
`-xvda4                       202:4    0 119.5G  0 part
  `-coreos-luks-root-nocrypt 253:0    0 119.5G  0 dm   /sysroot
nvme0n1                      259:0    0   931G  0 disk
```

この例では、**rhvworker01** の場合、利用可能なローカルデバイスは **nvme0n1** です。

3. 手順 2 で選択した各デバイスの一意の ID を特定します。

```
sh-4.4#  ls -l /dev/disk/by-id/ | grep nvme0n1
lrwxrwxrwx. 1 root root 13 Mar 17 16:24 nvme-
INTEL_SSDPE2KX010T7_PHLF733402LM1P0GGN -> ../../nvme0n1
```

上記の例では、ローカルデバイス **nvme0n1**の ID

```
nvme-INTEL_SSDPE2KX010T7_PHLF733402LM1P0GGN
```

4. 上記の手順を繰り返し、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のデバイスID を特定します。詳細は、ナレッジベースアーティクルを参照してください。

1.5. RED HAT VIRTUALIZATION プラットフォームでの 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ついてのセクションにあ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 OpenShift Contaner Storage ラベルを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"}'
```

各ノードの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
```

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。
出力例:

```
NAME                                READY   STATUS  RESTARTS AGE
local-block-local-diskmaker-cmfql     1/1     Running 0       31s
local-block-local-diskmaker-g6fzr     1/1     Running 0       31s
local-block-local-diskmaker-jkqxt     1/1     Running 0       31s
local-block-local-provisioner-jgqcc   1/1     Running 0       31s
```

```
local-block-local-provisioner-mx49d     1/1     Running 0       31s
local-block-local-provisioner-qbcvp     1/1     Running 0       31s
local-storage-operator-54bc7566c6-ddbrt 1/1     Running 0       12m
```

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   931Gi      RWO            Delete           Available           localblock
2m11s
local-pv-183bfc0a   931Gi      RWO            Delete           Available           localblock
2m15s
local-pv-b2f5cb25   931Gi      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                    RECLAIMPOLICY
VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION     AGE
localblock    kubernetes.io/no-provisioner   Delete
WaitForFirstConsumer  false                  2m10s
```

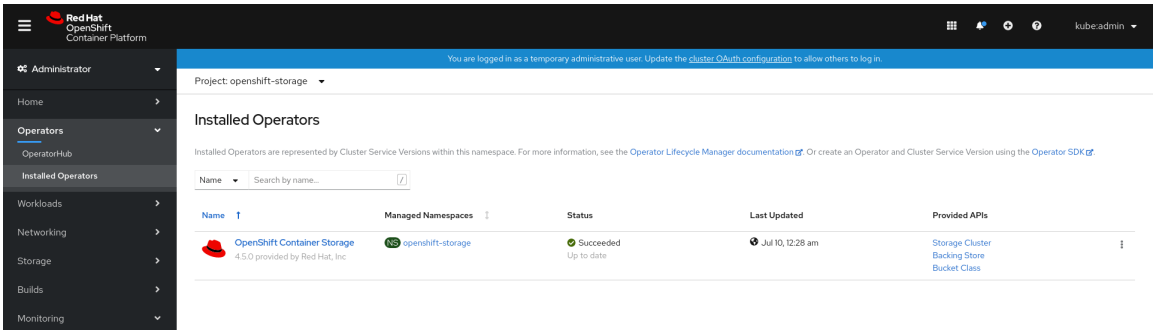
6. **localblock** Storage Class を使用する OpenShift Container Storage Cluster Service を作成します。

   a. OpenShift Web コンソールにログインします。

   b. OpenShift Web コンソールから **Operators → Installed Operators** をクリックし、インストールされた Operator を表示します。選択された **Project** が **openshift-storage** であることを確認します。

   c. **Installed Operators** ページで、**Openshift Container Storage** をクリックします。

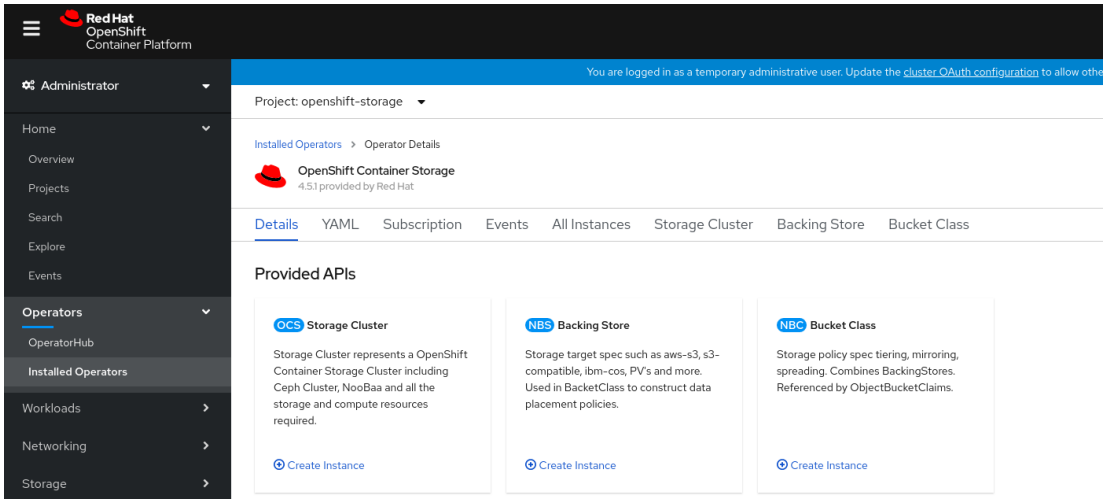
   図1.3 OpenShift Container Storage Operator ページ



   d. **Installed Operators → Operator Details** ページで、以下のいずれかを実行して Storage Cluster Service を作成します。

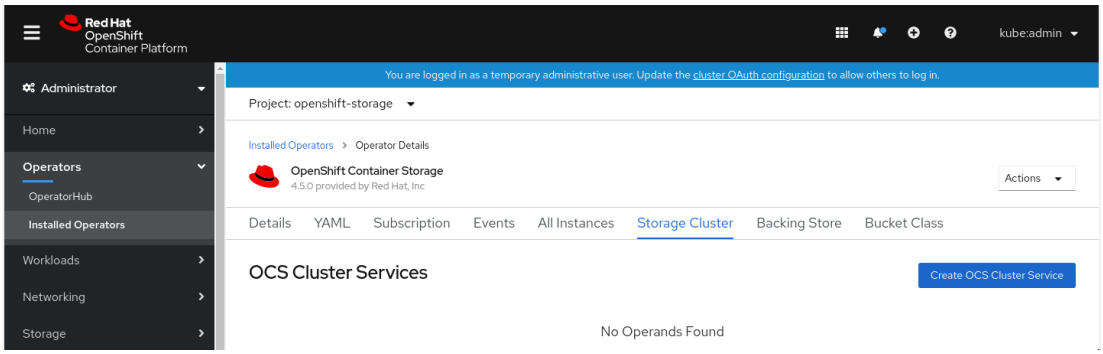
- Details タブで Provided APIs → OCS Storage Clusterで、 Create Instance をクリックします。

図1.4 Operator Details ページ

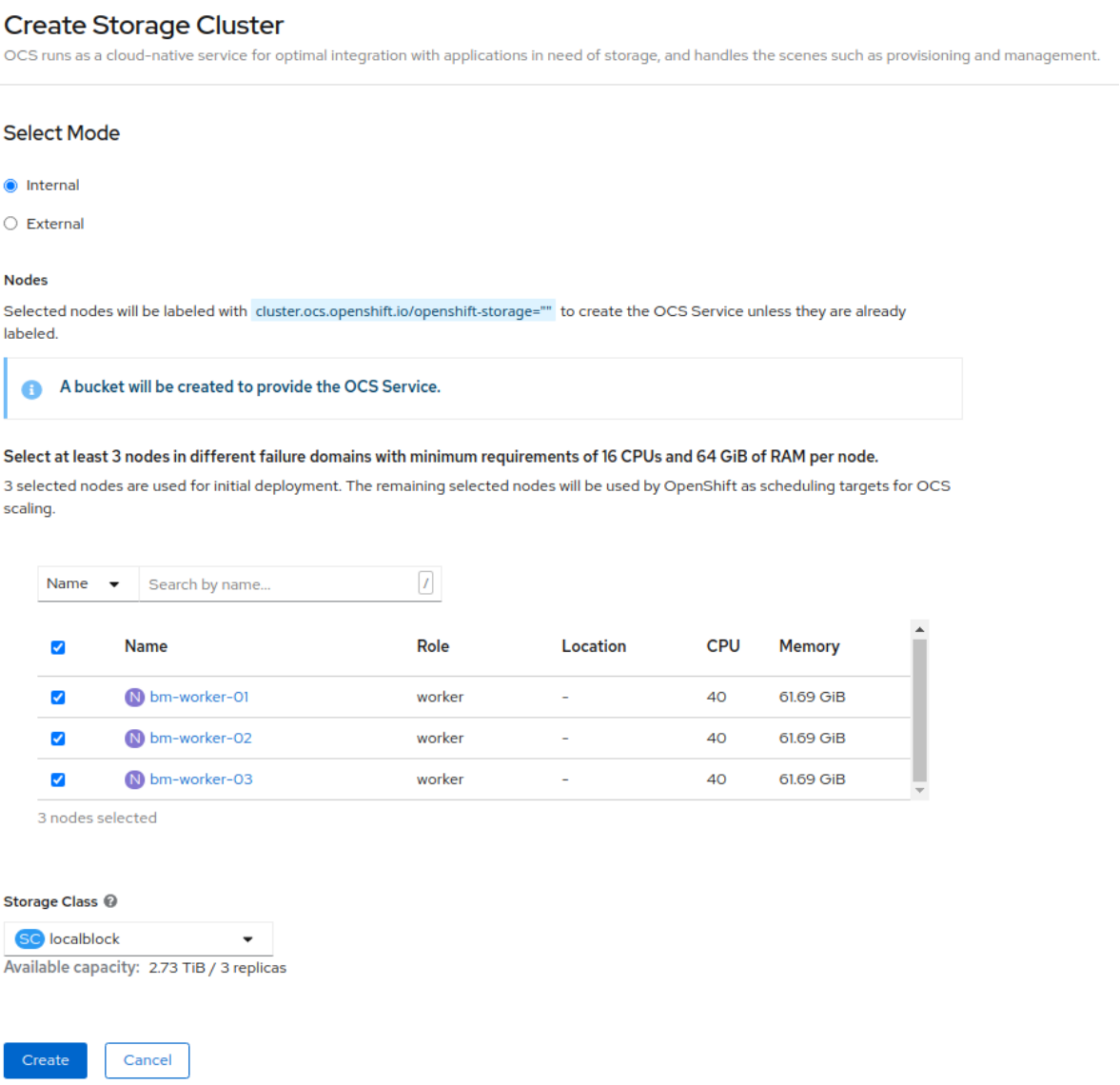


- または、Storage cluster タブを選択し、Create OCS Cluster Service をクリックします。

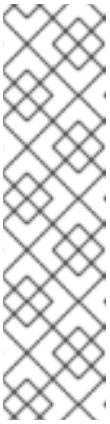
図1.5 Storage Cluster タブ



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



- **Select Mode** を **Internal** のままにします。
- **Nodes** セクションでは、OpenShift Container Storage サービスを使用するには、利用可能な一覧から 3 つ以上のワーカーノードを選択します。
  高可用性を確保するために、ワーカーノードは 3 つの異なる物理ノード、ラック、障害ドメインに分散することが推奨されます。



注記

- クラスターで特定のワーカーノードを見つけるには、Name または Label に基づいてノードをフィルターできます。
  - Name では、ノード名で検索できます。
  - Label では、事前に定義されたラベルを選択して検索できます。
- OpenShift Container Storage のラックラベルがデータセンターの物理ラックに合わせて調整されていることを確認し、障害ドメインのレベルで二重ノードに障害が発生しないようにします。

ノードの最小要件については、『プランニング』ガイドの「リソース要件」セクションを参照してください。

- **Storage Class** ドロップダウンリストから **localblock** を選択します。

8. **Create** をクリックします。

注記

Create ボタンは、最低でも 3 つのワーカーノードを選択した後にのみ有効になります。

デプロイメントに成功すると、3 つのストレージデバイスを持つストレージクラスターが作成されます。これらのデバイスは、選択したノードの 3 つに分散されます。この設定では、3 のレプリケーション係数が使用されます。初期クラスターをスケーリングするには、「ストレージノードのスケーリング」を参照してください。

検証手順

「Verifying your OpenShift Container Storage installation」を参照してください。

第2章 OPENSHIFT CONTAINER STORAGE デプロイメントの検証

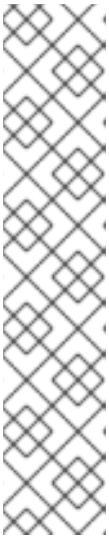
このセクションを使用して、OpenShift Container Storage が正常にデプロイされていることを確認します。

2.1. POD の状態の確認

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

手順

1. OpenShift Web コンソールの左側のペインから **Workloads → Pods** をクリックします。
2. **Project** ドロップダウンリストから **openshift-storage** を選択します。
   各コンポーネントについて予想される Pod 数や、これがノード数によってどのように異なるかについての詳細は、表2.1「OpenShift Container Storage クラスターに対応する Pod」 を参照してください。



注記

OpenShift Container Storage のクラスター全体でのデフォルトノードセ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
```

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

表2.1 OpenShift Container Storage クラスターに対応する Pod

コンポーネント	対応する Pod
OpenShift Container Storage Operator	**ocs-operator-*** （任意のワーカーノードに 1 Pod）
Rook-ceph Operator	**rook-ceph-operator-*** （任意のワーカーノードに 1 Pod）

コンポーネント	対応する Pod
Multicloud Object Gateway	• **noobaa-operator-*** (任意のワーカーノードに 1 Pod) • **noobaa-core-*** (任意のストレージノードに 1 Pod) • **nooba-db-*** (任意のストレージノードに 1 Pod) • **noobaa-endpoint-*** (任意のストレージノードに 1 Pod)
MON	**rook-ceph-mon-*** （ストレージノードに分散する 3 Pod）
MGR	**rook-ceph-mgr-*** （任意のストレージノードに 1 Pod）
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** （ストレージノードに分散する 2 Pod）
RGW	**rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*** (ストレージノードに分散する 2 Pod)

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

rook-ceph-drain-canary	**rook-ceph-drain-canary-*** （各ストレージノードに 1 Pod）

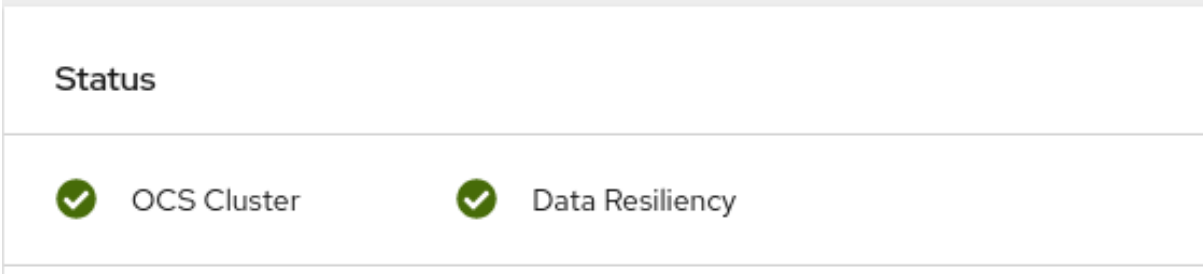
コンポーネント	対応する Pod
rook-ceph-crashcollector	**rook-ceph-crashcollector-*** （各ストレージノードに 1 Pod）
OSD	• **rook-ceph-osd-*** (各デバイス用に 1 Pod) • **rook-ceph-osd-prepare-ocs-deviceset-*** (各デバイス用に 1 Pod)

2.2. 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。

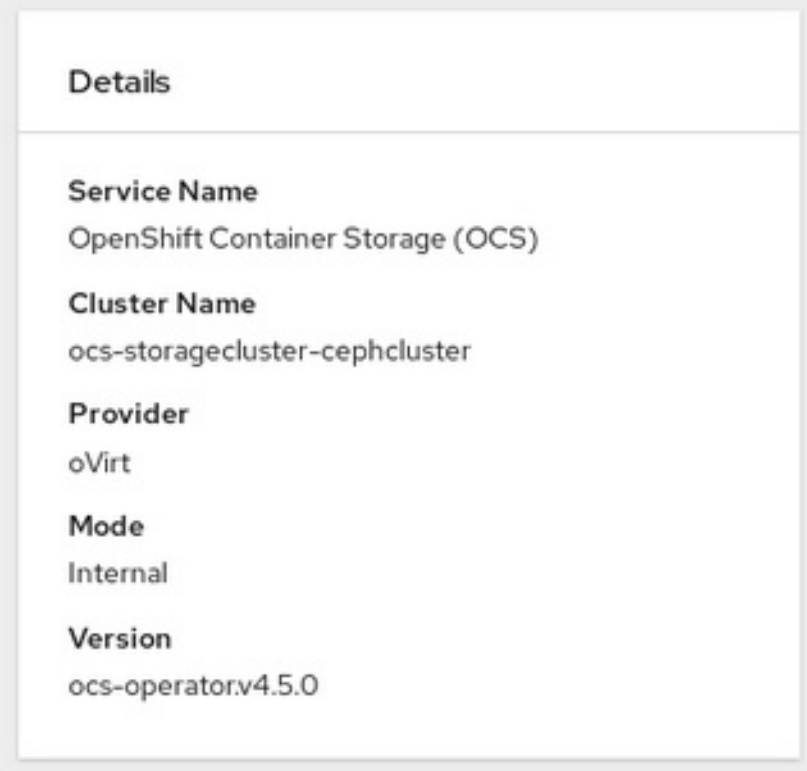
- OpenShift Web コンソールの左側のペインから **Home → Overview** をクリックし、**Persistent Storage** タブをクリックします。
- **Status カード** で、以下の画像のように **OCS Cluster** に緑色のチェックマークが表示されていることを確認します。

  図2.1 Persistent Storage Overview ダッシュボードの Health status カード



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

図2.2 Persistent Storage Overview ダッシュボードの Details カード

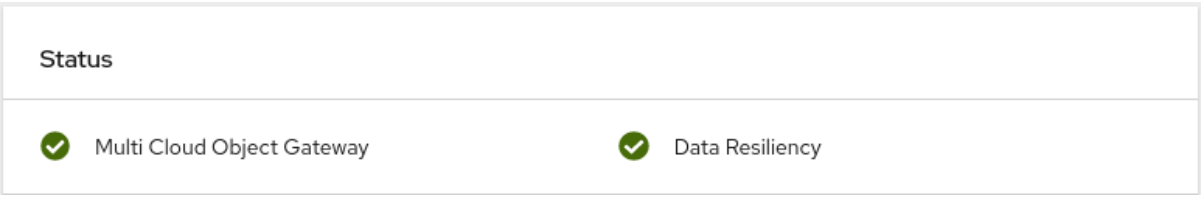


2.3. MULTICLOUD OBJECT GATEWAY が正常であることの確認

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。

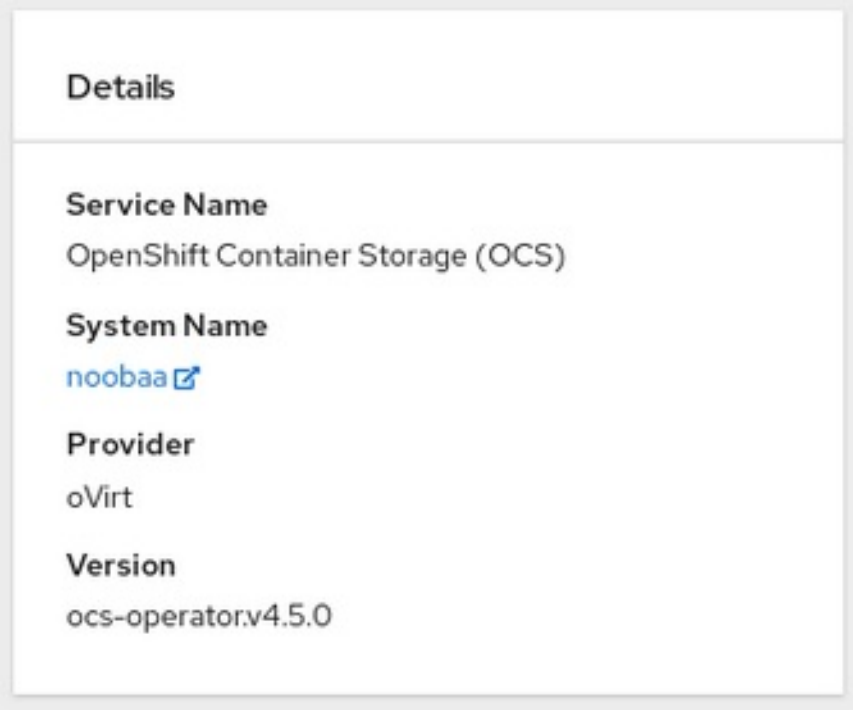
- OpenShift Web コンソールの左側のペインから **Home → Overview** をクリックし、**Object Service** タブをクリックします。
- **Status カード** で、以下のように Multicloud Object Gateway (MCG) ストレージに緑色のチェックマークが表示されていることを確認します。

  図2.3 Object Service Overview ダッシュボードの Health status カード

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

図2.4 Object Service Overview ダッシュボードの Details カード



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

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

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

第3章 OPENSHIFT CONTAINER STORAGE のアンインストール

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

このセクションの手順を使用して、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。
- アプリケーションが OpenShift Container Storage によって提供されるストレージクラスを使用して 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）を使用していないことを確認します。PVC および OBC はアンインストールプロセスで削除されます。

手順

1. OpenShift Container Storage ベースのストレージクラスプロビジョナーを使用する PVC および OBC をクエリーします。
   以下は例になります。

   ```
   $ 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/ )' | grep -v noobaa-default-backing-store-noobaa-pvc
   ```

   ```
   $ 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
   ```

2. 以下の手順に従って、直前の手順に記載されている PVC および OBC が削除されていることを確認します。
   モニタリングスタック、クラスターロギング Operator、またはイメージ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」
     残りの PVC または OBC のそれぞれに、以下の手順を実行します。

a. PVC または OBC を使用する Pod を判別します。

b. **Deployment**、**StatefulSet**、**DaemonSet**、**Job**、またはカスタムコントローラーなどの制御する側の API オブジェクトを特定します。
各 API 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** などの制御するオブジェクトを参照します。

c. API オブジェクトが OpenShift Container Storage によって提供される PVC または OBC を使用していないことを確認します。オブジェクトを削除するか、ストレージを置き換える必要があります。プロジェクトオーナーに、オブジェクトを安全に削除または変更できることを確認するよう依頼します。

注記

noobaa Pod は無視できます。

d. OBC を削除します。

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

e. 作成したカスタムバケットクラスを削除します。

```
$ oc get bucketclass -A  | grep -v noobaa-default-bucket-class
```

```
oc delete bucketclass <bucketclass name> -n <project-name>
```

f. カスタム Multi Cloud Gateway バッキングストアを作成している場合は、それらを削除します。

- バッキングストアの一覧を表示し、これらをメモします。

```
for bs in $(oc get backingstore -o name -n openshift-storage | grep -v noobaa-default-backing-store); do echo "Found backingstore $bs"; echo "Its has the following pods running :"; echo "$(oc get pods -o name -n openshift-storage | grep $(echo ${bs} | cut -f2 -d/))"; done
```

- 上記の各バッキングストアを削除し、依存するリソースも削除されていることを確認します。

```
for bs in $(oc get backingstore -o name -n openshift-storage | grep -v noobaa-default-backing-store); do echo "Deleting Backingstore $bs"; oc delete -n openshift-storage $bs; done
```

- 上上記のバッキングストアのいずれかが pv-pool をベースとする場合、対応する Pod および PVC も削除してください。

```
$ oc get pods -n openshift-storage | grep noobaa-pod | grep -v noobaa-default-backing-store-noobaa-pod
```

```
$ oc get pvc -n openshift-storage --no-headers | grep -v noobaa-db | grep noobaa-pvc | grep -v noobaa-default-backing-store-noobaa-pvc
```

-

g. 手順 1 に記載されている残りの PVC を削除します。

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

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

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

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

a. openshift-storage がアクティブな namespace の場合に別の namespace に切り替えます。以下は例になります。

```
$ oc project default
```

b. openshift-storage namespace を削除します。

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

c. 約 5 分間待機し、プロジェクトが正常に削除されたかどうかを確認します。

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

出力:

```
Error from server (NotFound): namespaces "openshift-storage" not found
```

注記

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

5. 各ノードでストレージ 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
```

削除されたディレクトリー **/var/lib/rook** が出力に表示されることを確認します。

ディレクトリーが存在しないことを確認します。

```
$ 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  ls -l /var/lib/rook; done
```

6. **openshift-storage.noobaa.io** ストレージクラスを削除します。

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

7. 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。

8. すべての PV が削除されていることを確認します。Released 状態のままの PV がある場合は、これを削除します。

```
# oc get pv | egrep 'ocs-storagecluster-ceph-rbd|ocs-storagecluster-cephfs'
```

```
# oc delete pv <pv name>
```

9. **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
cephclients.ceph.rook.io --wait=true --timeout=5m
```

10. OpenShift Container Platform Web コンソールで、OpenShift Container Storage が完全にアンインストールされていることを確認するには、以下を実行します。

    a. **Home → Overview** をクリックし、ダッシュボードにアクセスします。

    b. **Persistent Storage** および **Object Service** タブが **Cluster** タブの横に表示されないことを確認します。

3.2. 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。

編集前

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

編集後

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

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

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

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

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

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

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

前提条件

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

手順

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

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

編集前

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

編集後

```
.
.
.
storage:
  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
```

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

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

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

前提条件

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

手順

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

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

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

2. PVC を削除します。

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

第4章 OPENSHIFT 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レーションおよび保持期間を短く設定することを推奨します。詳細は、OpenShift Container Platform ドキュメントの「Curator スケジュールの設定」および「永続ストレージの設定」の「**Prometheus メトリクスデータの保持時間の変更**」サブセクションを参照してください。

これらのサービスのストレージ領域が不足する場合は、Red Hat カスタマーサポートにお問い合わせください。

4.1. OPENSHIFT 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ーイメージレジストリーを提供します。通常、レジストリーはクラスター上にビルドされたイメージの公開ターゲットとして、またクラスター上で実行されるワークロードのイメージのソースとして使用されます。

警告

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

前提条件

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

4.2. OPENSHIFT 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 **openshift-monitoring-config** 設定マップを定義します。
   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。

   サンプル openshift-monitoring-config 設定マップ

```
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>
```

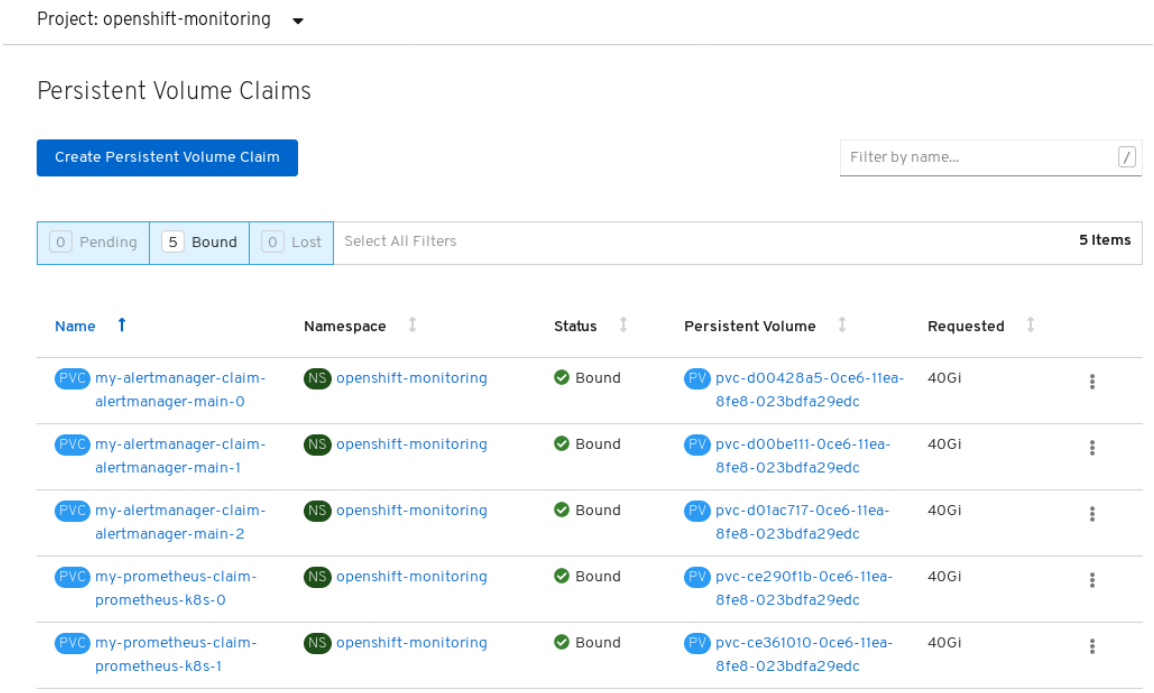
5. **Create** をクリックして、設定マップを保存し、作成します。

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

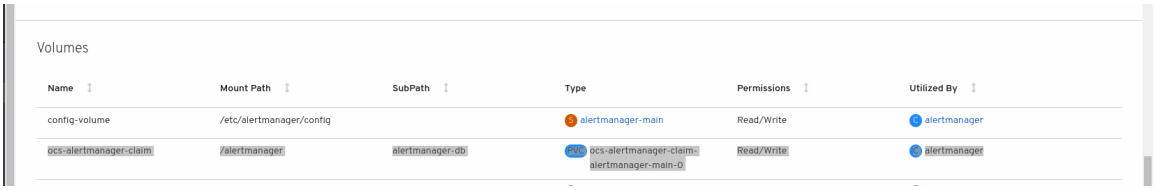


2. 新規の **alertmanager-main-*** Pod が **Running** 状態で表示されることを確認します。

a. 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)

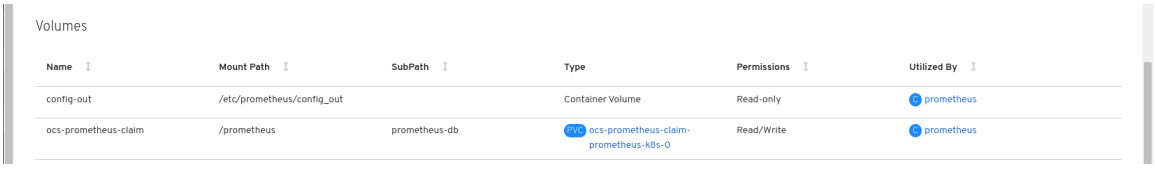


3. 新規 **prometheus-k8s-*** Pod が **Running** 状態で表示されることを確認します。

a. 新規 **prometheus-k8s-*** Pod をクリックし、Pod の詳細を表示します。

b. **Volumes** までスクロールダウンし、ボリュームに新規の Persistent Volume Claim (永続ボリューム要求、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）



4.3. OPENSHIFT 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レーションおよび保持期間を短く設定することを推奨します。詳細は、OpenShift Container Platform ドキュメントでクラスターロギング Curator について参照してください。

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

4.3.1. 永続ストレージの設定

ストレージクラス名およびサイズパラメーターを使用して、 Elasticsearch クラスターの永続ストレージクラスおよびサイズを設定できます。Cluster Logging Operator は、これらのパラメーターに基づいて、Elasticsearch クラスターの各データノードについて Persistent Volume Claim （永続ボリューム要求、PVC）を作成します。以下に例を示します。

```
spec:
   logStore:
     type: "elasticsearch"
     elasticsearch:
       nodeCount: 3
       storage:
         storageClassName: "ocs-storagecluster-ceph-rbd”
         size: "200G"
```

この例では、クラスター内の各データノードが **200GiB** の **ocs-storagecluster-ceph-rbd** ストレージを要求する Persistent Volume Claim（永続ボリューム要求、PVC）にバインドされるように指定します。それぞれのプライマリーシャードは単一のレプリカによってサポートされます。シャードのコピーはすべてのノードにレプリケートされ、常に利用可能となり、冗長性ポリシーにより 2 つ以上のノードが存在する場合にコピーを復元できます。Elasticsearch レプリケーションポリシーについての詳細は、「クラスターロギングのデプロイおよび設定について」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

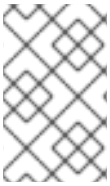
ストレージブロックを省略すると、デプロイメントはデフォルトのストレージでサポートされます。以下に例を示します。

```
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。

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

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



注記

OpenShift Container Storage でロギングを初めて設定する際にすべてのログを取得できます。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

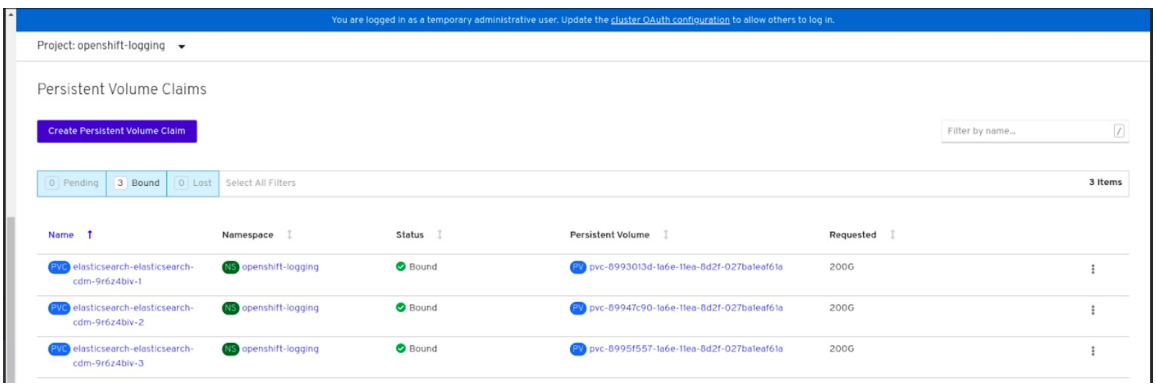
1. Persistent Volume Claim（永続ボリューム要求、PVC）が **elasticsearch** Pod にバインドされていることを確認します。

   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) の状態で表示されることを確認します。

   図4.1 作成済みのバインドされたクラスターロギング



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にしてください。

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

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

詳細は、「Elasticsearch データのキュレーション」を参照してください。

注記

Persistent Volume Claim（永続ボリューム要求、PVC）がサポートするクラスターロギングをアンインストールするには、それぞれのデプロイメントガイドのアンインストールについての章に記載されている、クラスターロギング Operator の OpenShift Container Storage からの削除についての手順を使用します。

第5章 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

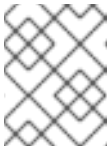
- OpenShift Container Platform がインストールされ、OpenShift Web コンソールへの管理者アクセスがある。
- OpenShift Container Storage が **openshift-storage** namespace にインストールされ、実行されている。

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。

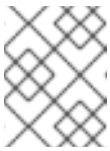
- **Create new claim** オプションをクリックします。
    a. **Storage Class** ドロップダウンリストから適切な **CephFS** または **RBD** ストレージクラスを選択します。
    b. Persistent Volume Claim (永続ボリューム要求、PVC) の名前を指定します。
    c. ReadWriteOnce (RWO) または ReadWriteMany (RWX) アクセスモードを選択します。



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. コンテナー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6章 ストレージノードのスケーリング

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

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

6.1. 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
  - ローカルストレージデバイス
  - 容量のプランニング

警告

常にストレージ容量が十分にあることを確認してください。

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

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

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

6.2. 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、Red Hat Virtualization 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。

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。

- 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。ローカルストレージ Operator のインストールについて参照してください。
- 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。

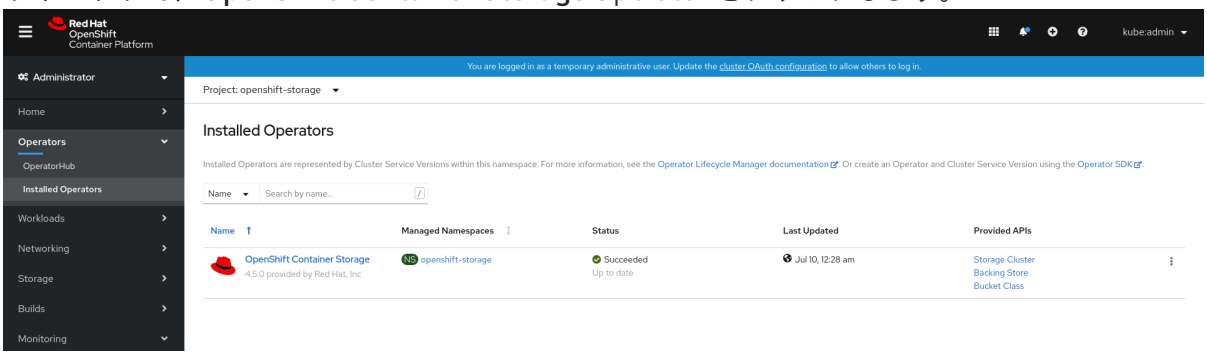
```
$ oc get pv | grep localblock | grep Available
```

出力例:

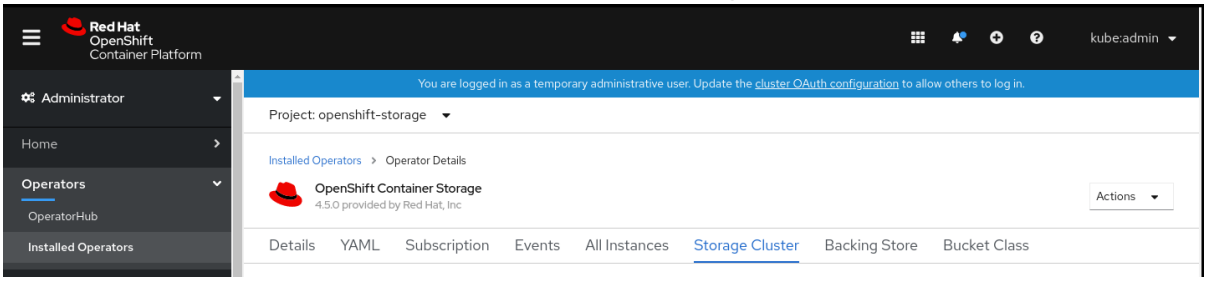
```
local-pv-5ee61dcc   931Gi   RWO    Delete  Available   localblock     2m35s
local-pv-b1fa607a   931Gi   RWO    Delete  Available   localblock     2m27s
local-pv-e971c51d   931Gi   RWO    Delete  Available   localblock     2m22s
...
```

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。

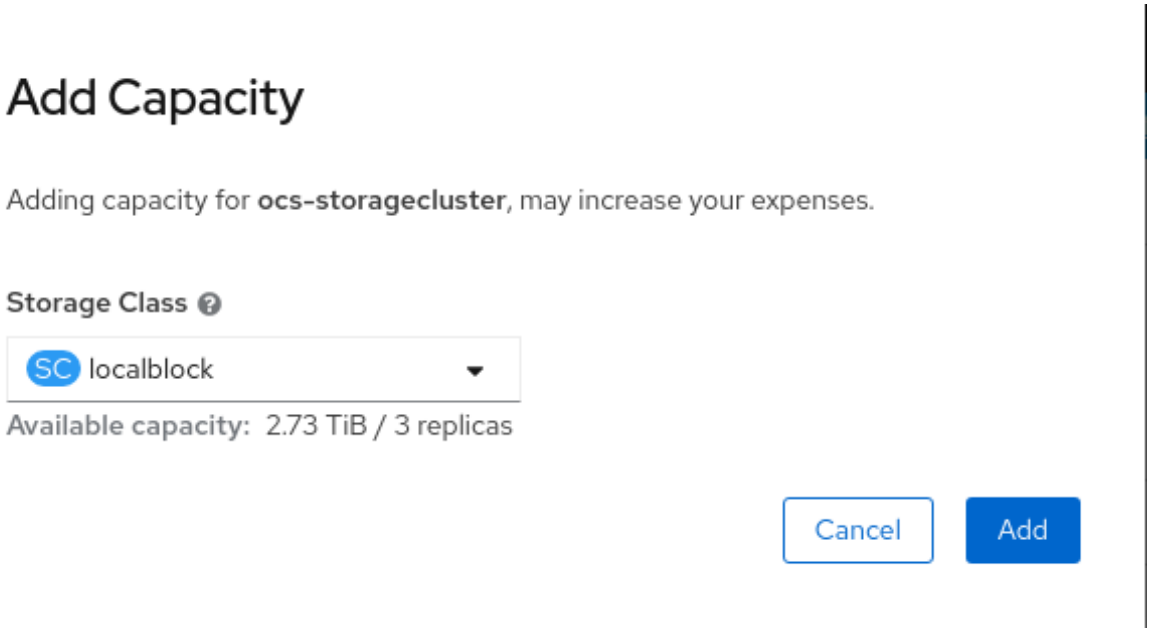
3. OpenShift Web コンソールに移動します。
4. 左側のナビゲーションバーの **Operators** をクリックします。
5. **Installed Operators** を選択します。
6. ウィンドウで、**OpenShift Container Storage** Operator をクリックします。



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。



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
```

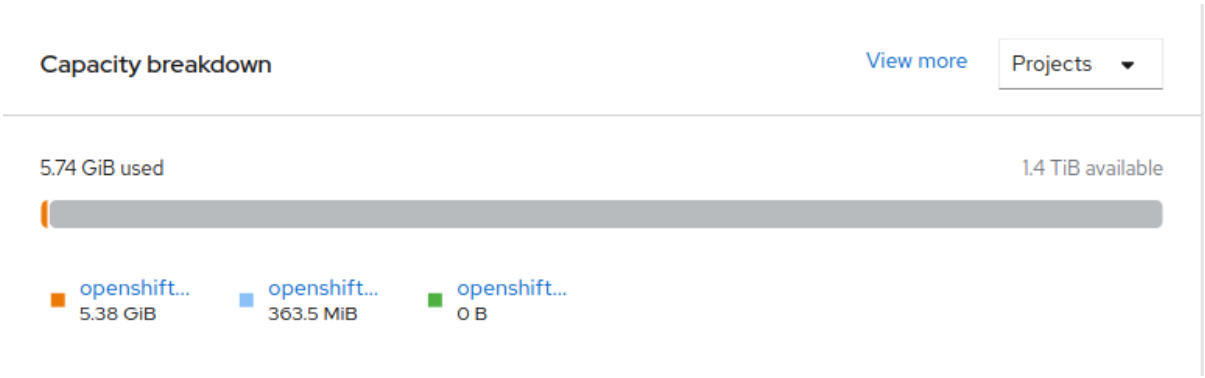
出力例:

```
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-rll47   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。

6.3. 新規ノードの追加によるストレージ容量のスケールアウト

ストレージ容量をスケールアウトするには、以下を実行する必要があります。

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

6.3.1. 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、Red Hat Virtualization インフラストラクチャーにノードを追加します。

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

1. 必要なインフラストラクチャーで Red Hat Virtualization 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。
2. 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。

3. **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
```

4. 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>
```

5. **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。

6. 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

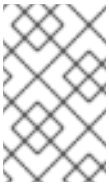
a. 新規ノードについて、**Action Menu (⋮)** → **Edit Labels** をクリックします。

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、OpenS+hift 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。

6.3.2. 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-***

6.3.3. ストレージ容量のスケールアップ

新規ノードを OpenShift Container Storage に追加した後に、「容量の 追加によるストレージのスケールアップ」で説明されているようにストレージ容量をスケールアップ する必要があります。

第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。

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:
```

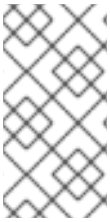
```
  Creation Timestamp:  2019-07-29T16:22:06Z
  Generation:          1
  Resource Version:    6718822
  Self Link:           /apis/noobaa.io/v1alpha1/namespaces/openshift-storage/noobaas/noobaa
  UID:                 019cfb4a-b21d-11e9-9a02-06c8de012f9e
Spec:
Status:
  Accounts:
    Admin:
      Secret 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 : HKLbH1rSuVU0I/souIkSiA==

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

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

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

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

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

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

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

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

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

No OBC's found.
```

(1) 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. 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

7.3.1. 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。

これを実行するには、MCG で使用できるバッキングストレージを追加します。

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
```

- 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。

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 <backing-store-type> <backingstore_name> --access-key=
<AWS ACCESS KEY> --secret-key=<AWS SECRET ACCESS KEY> --target-bucket
<bucket-name>
```

   a. **<backing-store-type>** を、関連するバッキングストアタイプの **aws-s3**、**google-cloud-store**、**azure-blob**、**s3-compatible**、または **ibm-cos** に置き換えます。

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

   c. **<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。

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

```
INFO[0001]   Exists: NooBaa "noobaa"
INFO[0002]   Created: BackingStore "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. 特定のバッキングストアについて以下の YAML を適用します。

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

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

   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。

   c. <backing-store-type> を、関連するバッキングストアタイプの **aws-s3**、**google-cloud-store**、 **azure-blob**、**s3-compatible**、または **ibm-cos** に置き換えます。

7.3.2. S3 と互換性のある NooBaa バッキングストアの作成

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 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、NooBaa に対して、バッキングストア、およびその後のデータストレージおよび管理のためのターゲットバケットとして使用するバケットについて指示します。
   出力は次のようになります。

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

YAML を使用してバッキングストアを作成することもできます。

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

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

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

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

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

```
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: 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
```

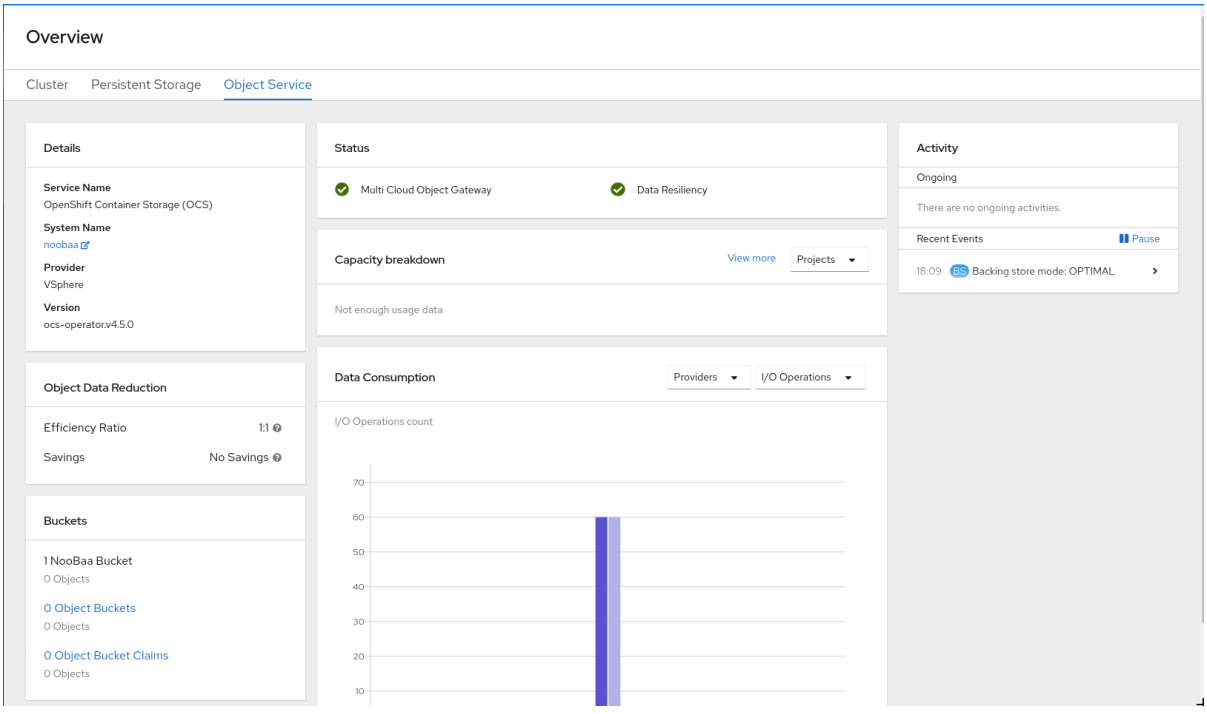
a. **<backingstore-secret-name>** を、直前の手順で **CephObjectStore** で作成したシークレットの名前に置き換えます。

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

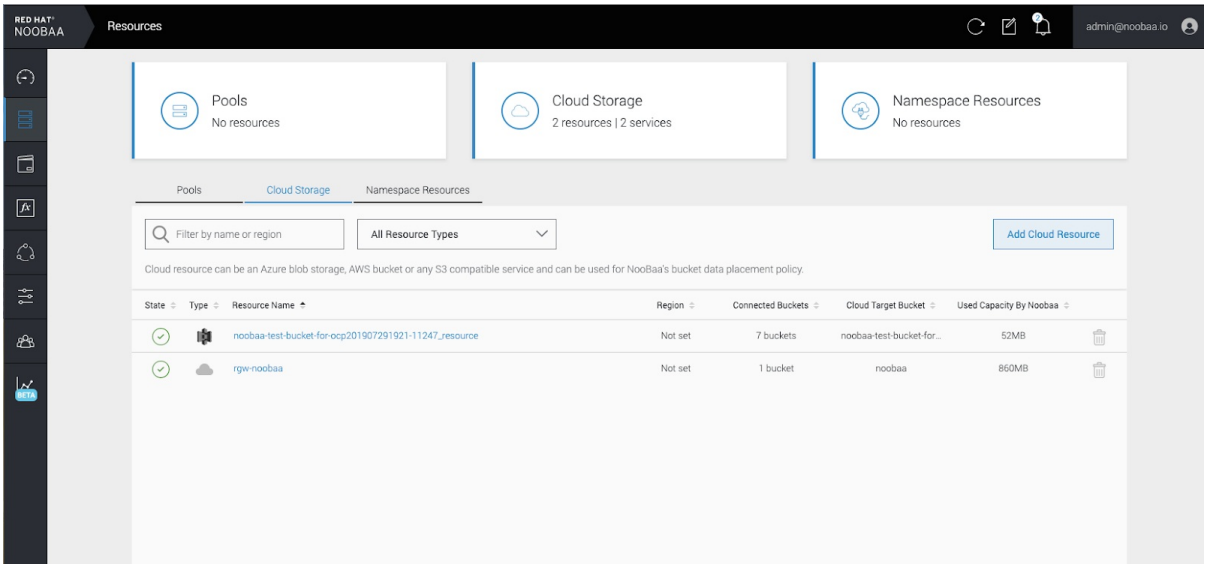
7.3.3. ユーザーインターフェースを使用したハイブリッドおよびマルチクラウドのストレージリソースの追加

手順

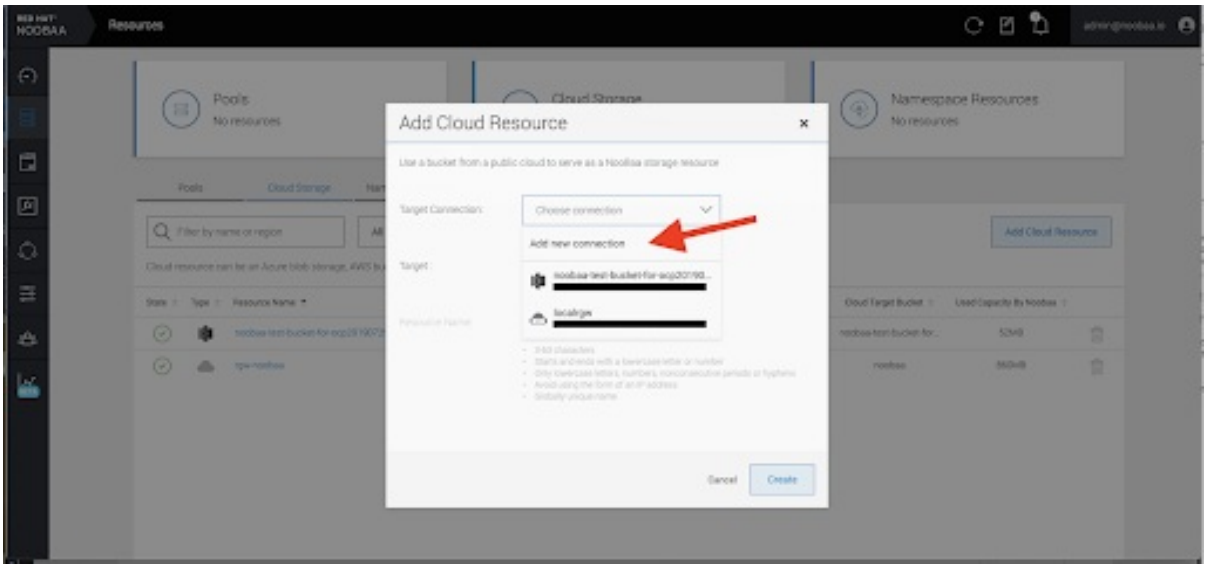
1. OpenShift Storage コンソールで、**Overview → Object Service →** に移動し、**noobaa** リンクを選択します。



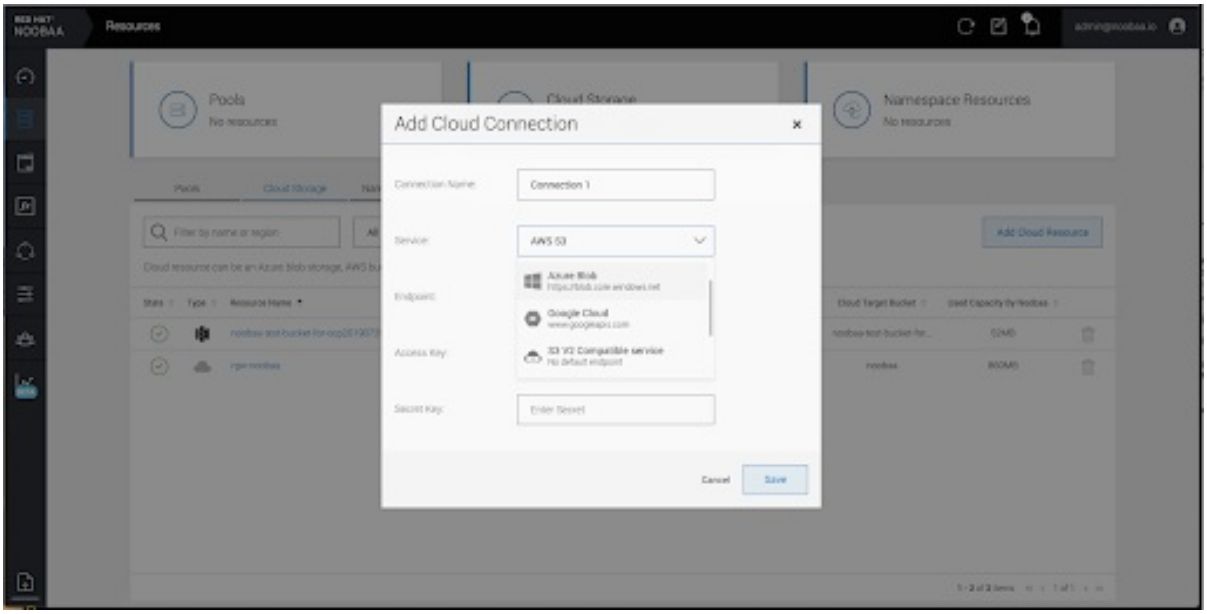
2. 以下に強調表示されているように左側にある **Resources** タブを選択します。設定する一覧から、**Add Cloud Resource** を選択します。



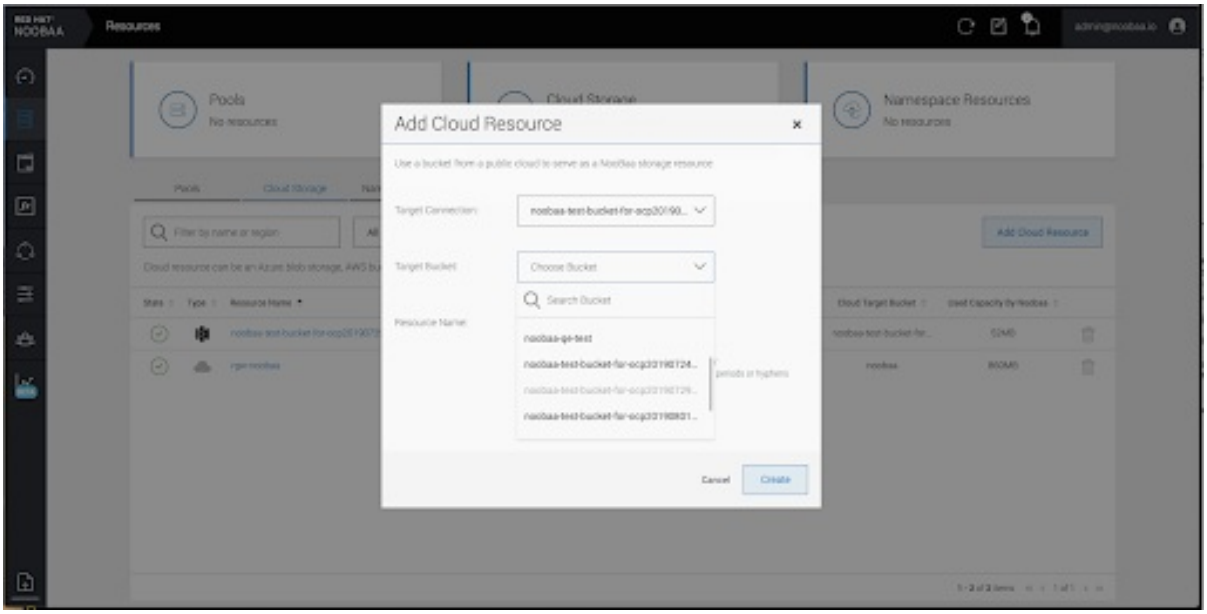
3. **Add new connection** を選択します。



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、必要な数のバッキングストアを作成します。

注記

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

7.3.4. 新規バケットクラスの作成

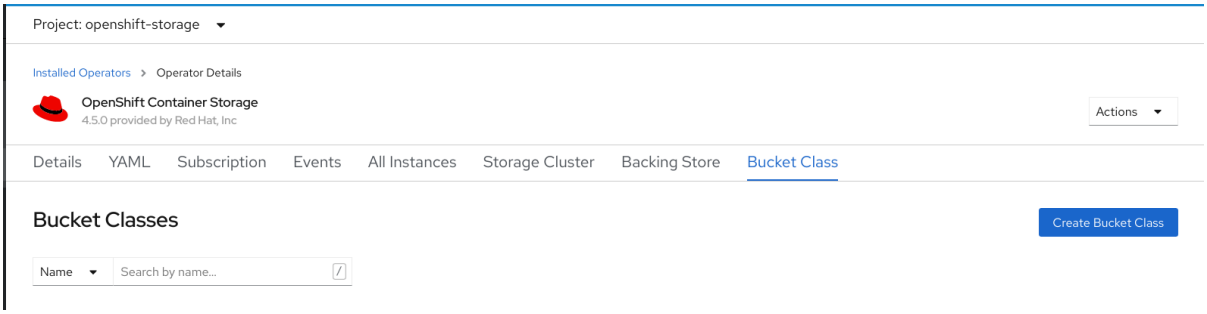
バケットクラスは、OBC (Object Bucket Class) の階層ポリシーおよびデータ配置を定義するバケットのクラスを表す CRD です。

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。

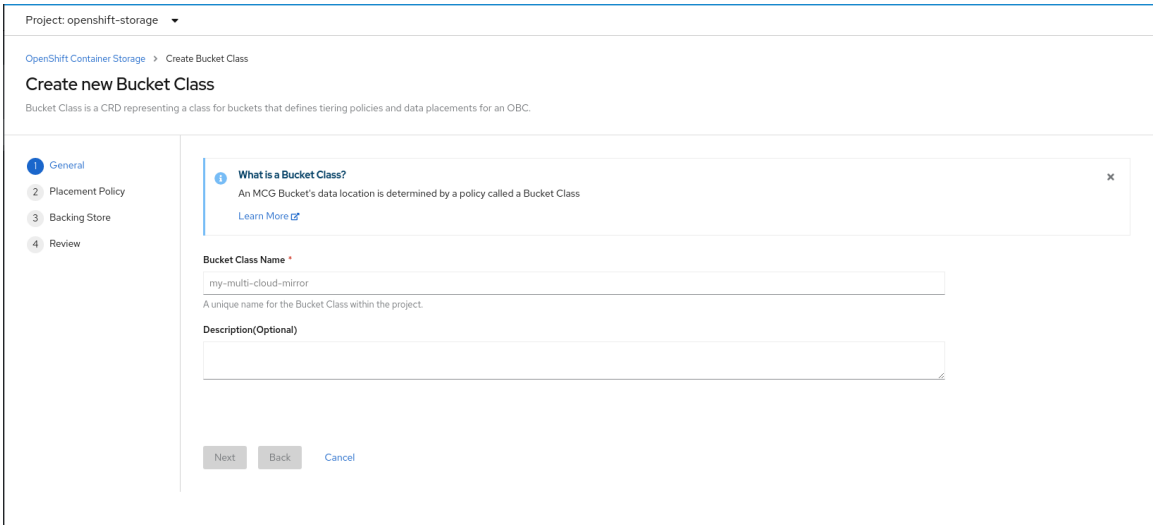
図7.1 Bucket Class タブのある OpenShift Container Storage Operator ページ



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。

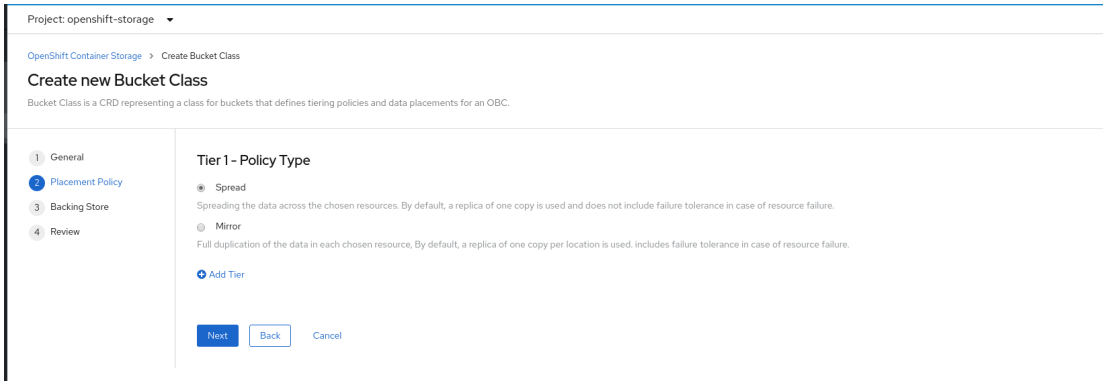
図7.2 Create Bucket Class ページ



b. Placement Policy で **Tier 1 - Policy Type**を選択し、**Next** をクリックします。要件に応じて、いずれかのオプションを選択できます。

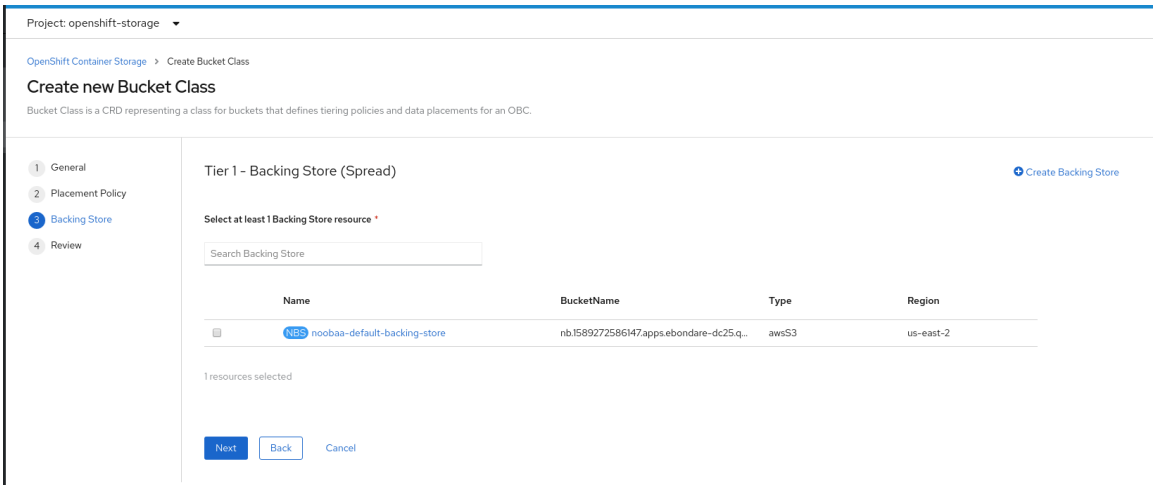
- **Spread** により、選択したリソース全体にデータを分散できます。
- **Mirror** により、選択したリソース全体でデータを完全に複製できます。
- **Add Tier** をクリックし、別のポリシー階層を追加します。

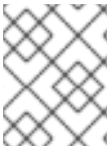
図7.3 階層 1 - Policy Type 選択ページ



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、新規バッキングストアを作成することもできます。

図7.4 階層 1 - Baking Store 選択ページ



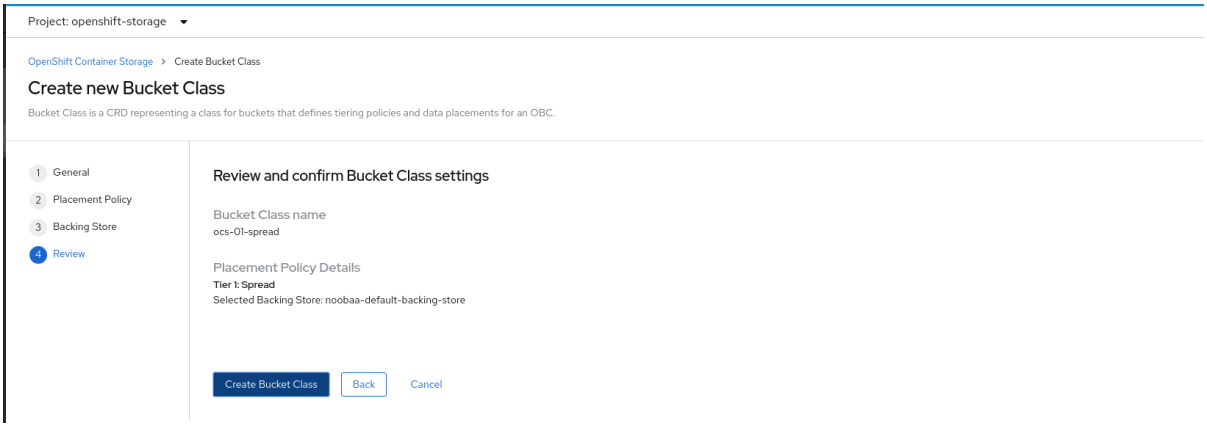


注記

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

a. Bucket Class 設定を確認し、確認します。

図7.5 バケットクラス設定の確認ページ



b. **Create Bucket Class** をクリックします。

検証手順

1. **Operators → Installed Operators** をクリックします。
2. **OpenShift Container Storage** Operator をクリックします。
3. 新しい Bucket Class を検索するか、または **Bucket Class** タブをクリックし、すべての Bucket Class を表示します。

7.3.5. 新規バッキングストアの作成

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

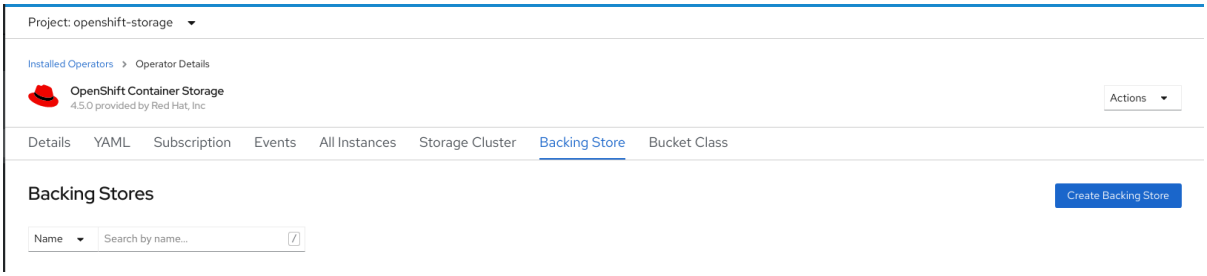
前提条件

- OpenShift への管理者アクセス。

手順

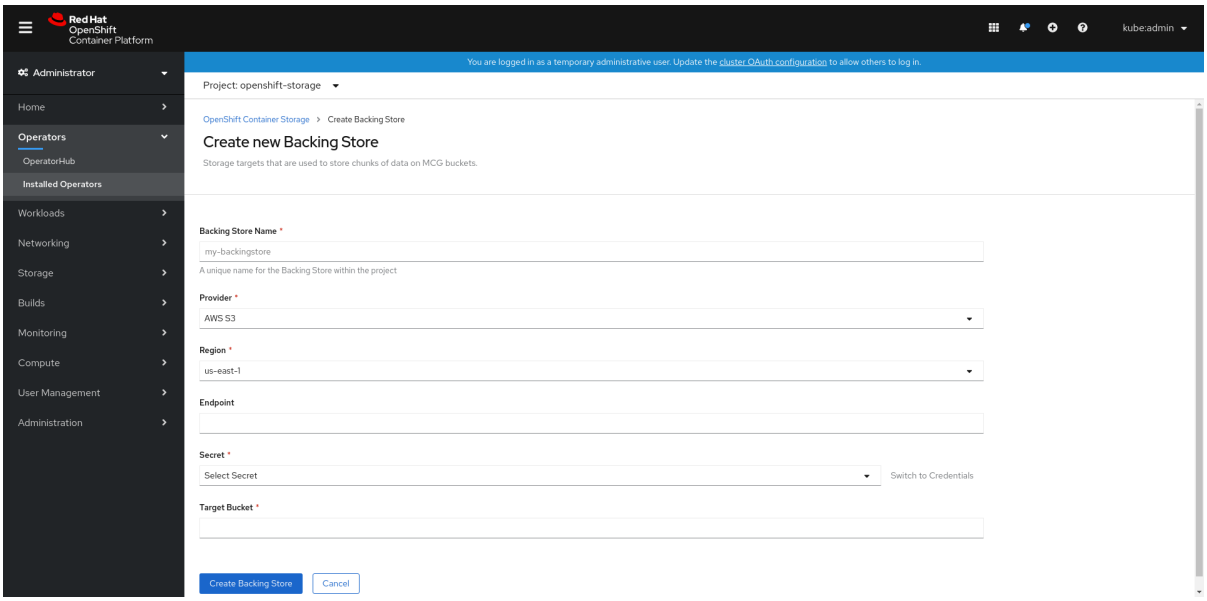
1. OpenShift Web コンソールの左側のペインで **Operators → Installed Operators** をクリックし、インストールされた Operator を表示します。
2. **OpenShift Container Storage** Operator をクリックします。
3. OpenShift Container Storage Operator ページで右側にスクロールし、**Backing Store** タブをクリックします。

図7.6 バッキングストアタブのある OpenShift Container Storage Operator ページ



4. **Create Backing Store** をクリックします。

図7.7 Create Backing Store ページ



5. Create New Backing Store ページで、以下を実行します。
   a. **Backing Store Name** を入力します。
   b. **Provider** を選択します。
   c. **Region** を選択します。
   d. **Endpoint** を入力します。これはオプションです。
   e. ドロップダウンリストから **Secret** を選択するか、または独自のシークレットを作成します。オプションで、**Switch to Credentials** ビューを選択すると、必要なシークレットを入力できます。

注記

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

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

6. **Create Backing Store** をクリックします。

検証手順

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

7.4. 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 で使用できるバッキ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」

7.4.1. 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
```

7.4.2. 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
```

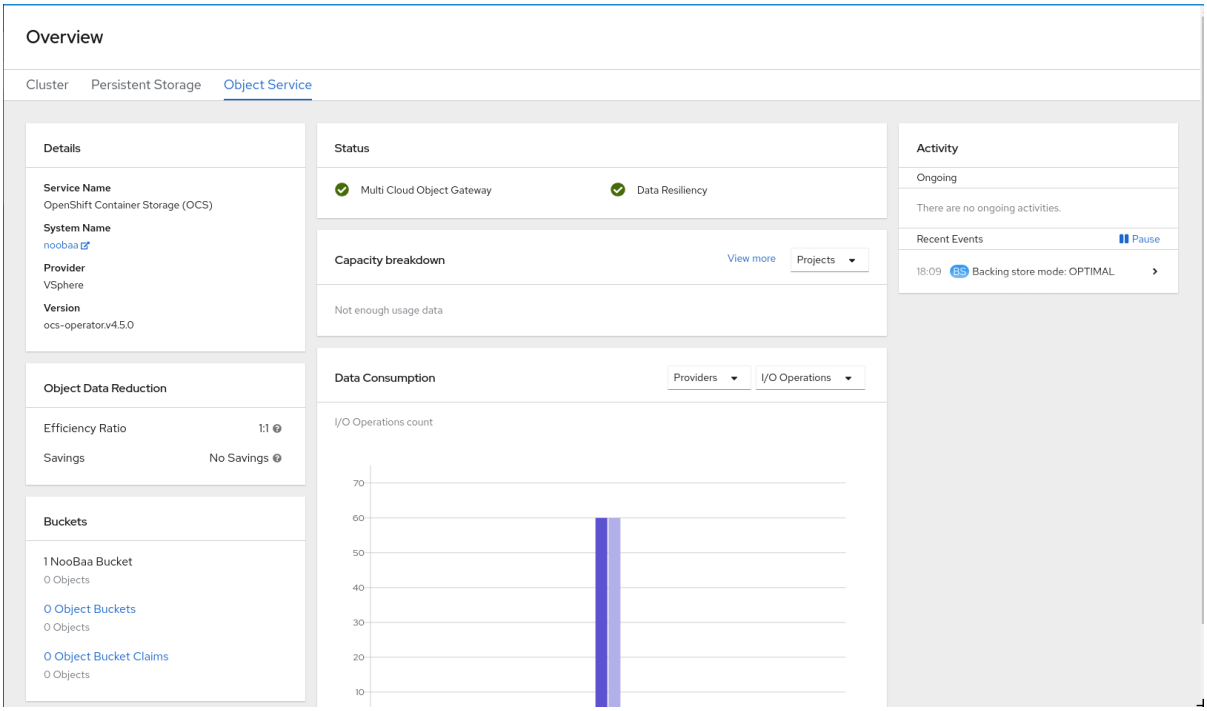
2. 以下の行を標準の Object Bucket Claim (オブジェクトバケット要求、OBC) に追加します。

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

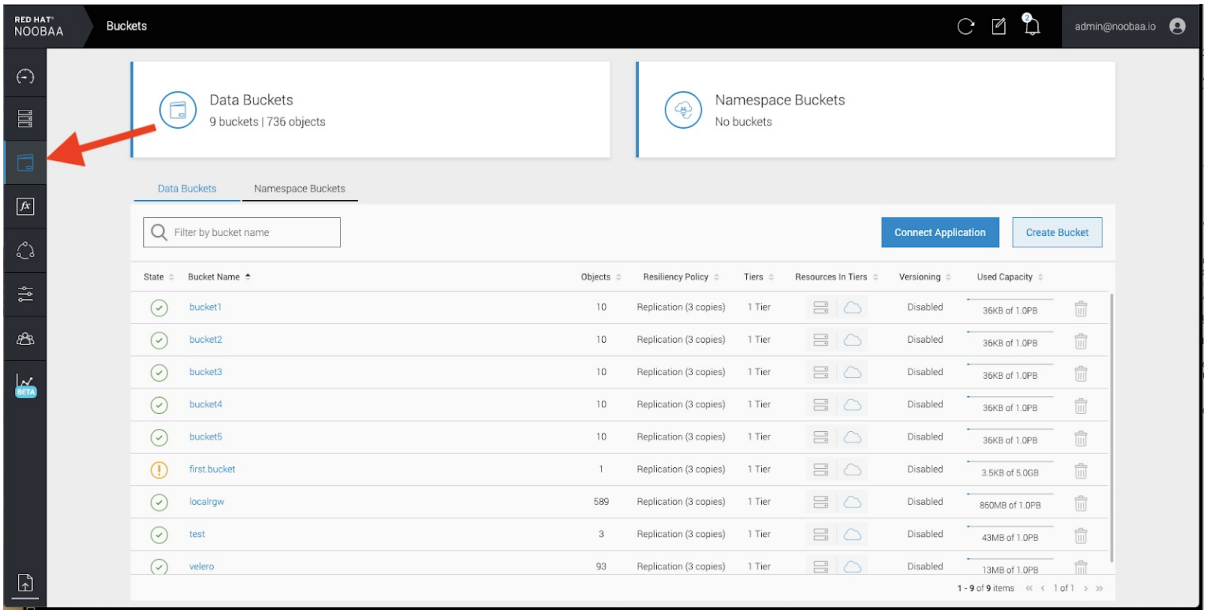
OBC についての詳細は、「Object Bucket Claim（オブジェクトバケット要求）」 を参照してください。

7.4.3. 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。

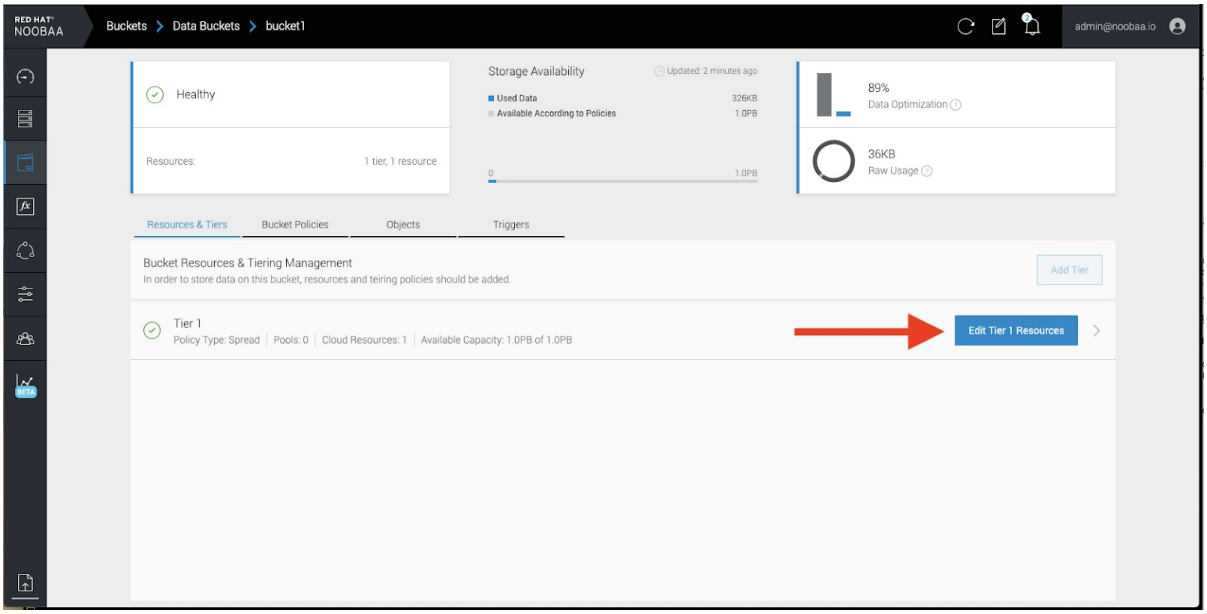


2. 左側の **buckets** アイコンをクリックします。バケットの一覧が表示されます。

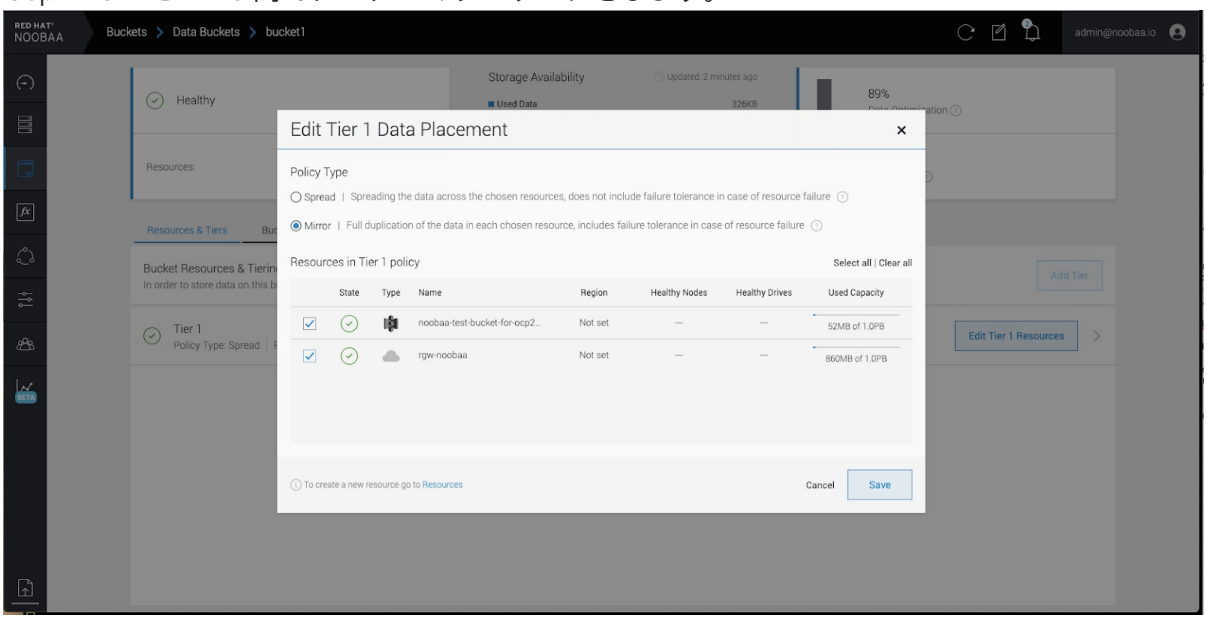


3. 更新するバケットを選択します。

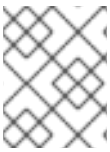
4. **Edit Tier 1 Resources** をクリックします。



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。

7.5. 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。

7.5.1. 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。

7.5.2. 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す。Object Bucket Claim (オブジェクトバケット要求) の場合、NooBaa はアカウント **obc-account.<generated bucket name>@noobaa.io** を自動的に作成します。

注記

バケットポリシー条件はサポートされていません。

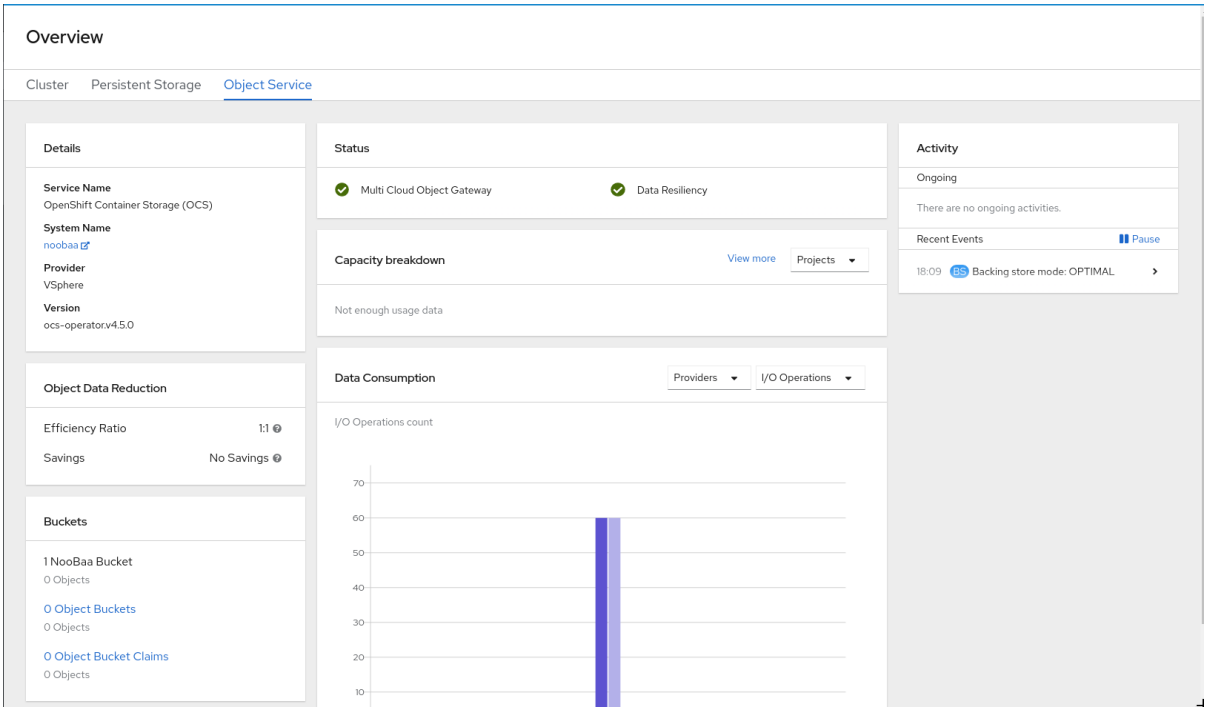
7.5.3. Multicloud Object Gateway での AWS S3 ユーザーの作成

前提条件

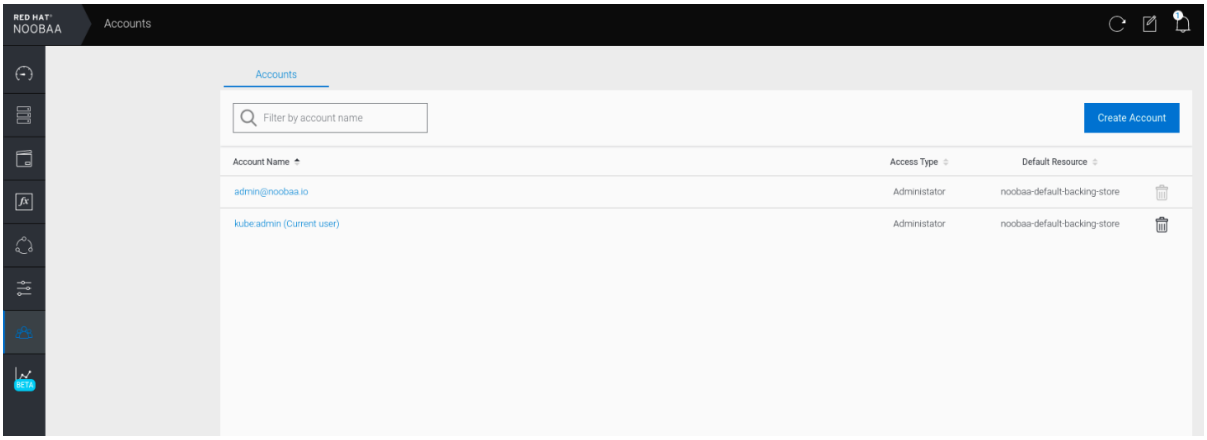
- 実行中の OpenShift Container Storage Platform
- Multicloud Object Gateway へのアクセス。「アプリケーションの使用による Multicloud Object Gateway へのアクセス」 を参照してください。

手順

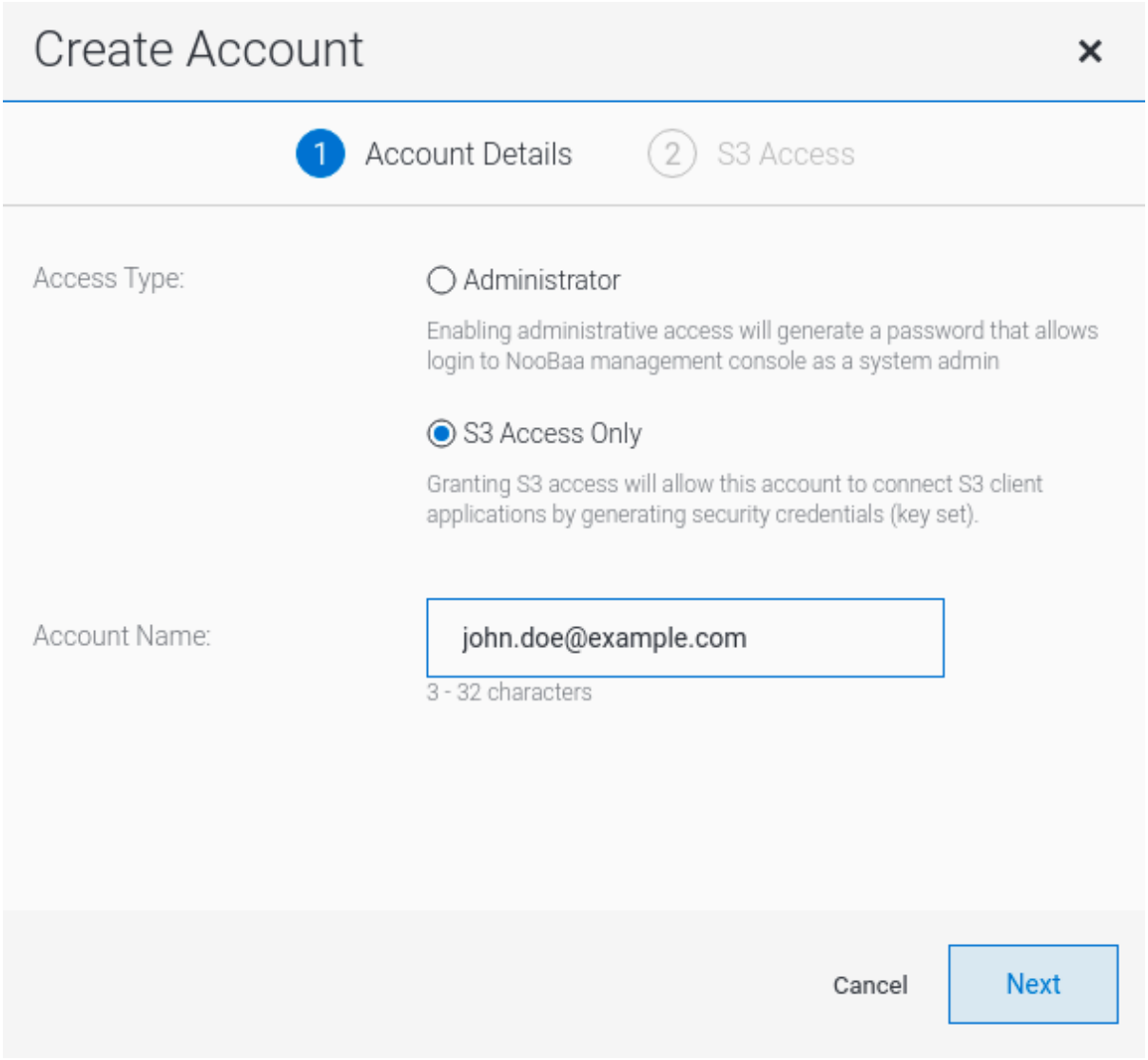
1. OpenShift Storage コンソールで、**Overview → Object Service →** に移動し、**noobaa** リンクを選択します。



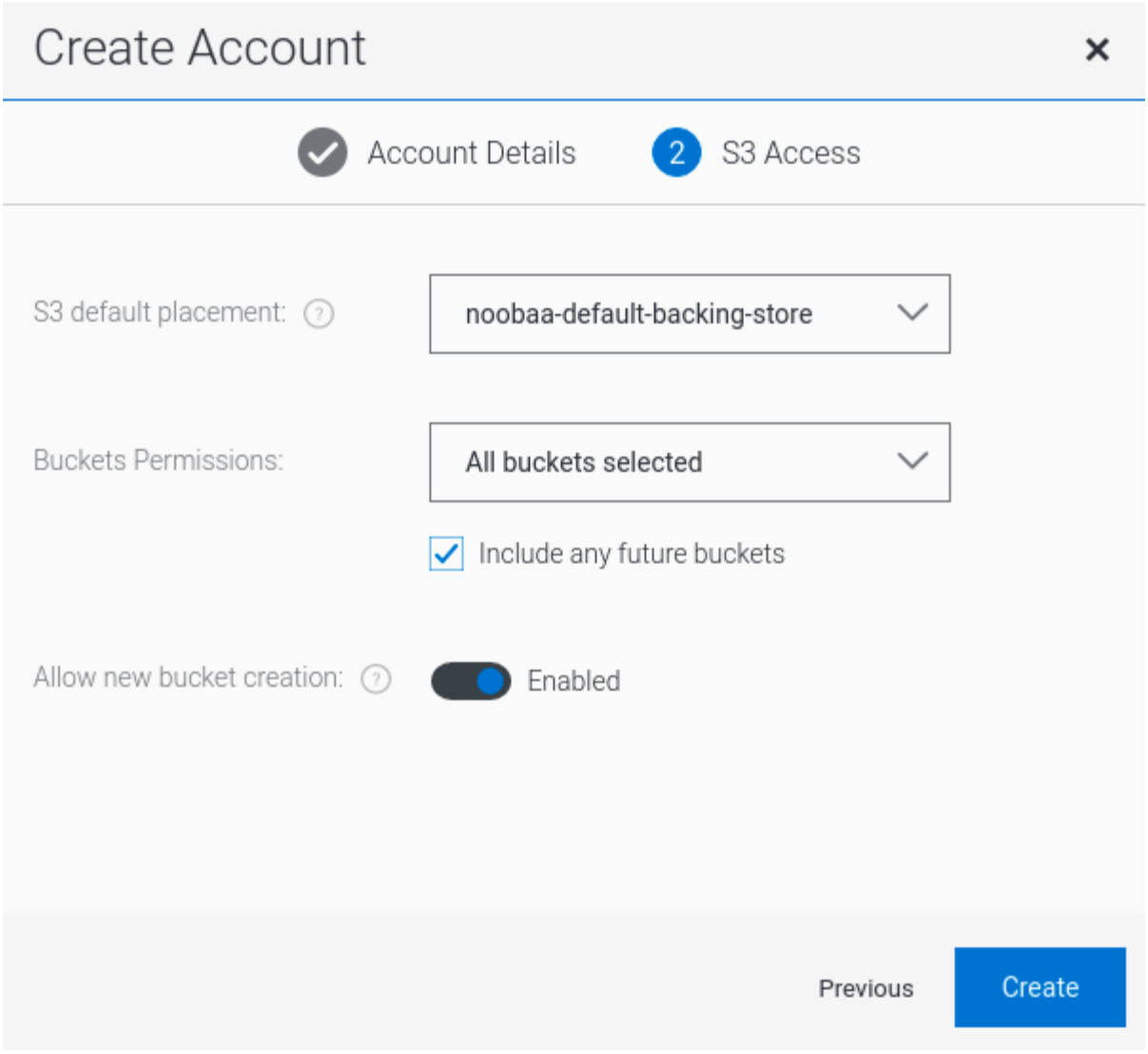
2. **Accounts** タブで、**Create Account** をクリックします。



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



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。



7.6. 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。

7.6.1. 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、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: noobaa
```

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

a. **<obc-name>** を、一意の Object Bucket Claim（オブジェクトバケット要求）の名前に置き換えます。

b. **<obc-bucket-name>** を、Object Bucket Claim（オブジェクトバケット要求）の一意のバケット名に置き換えます。

2. YAML ファイルにさらに行を追加して、Object Bucket Claim（オブジェクトバケット要求）の使用を自動化できます。以下の例はバケット要求の結果のマッピングです。これは、データを含む設定マップおよび認証情報のあるシークレットです。この特定のジョブは NooBaa からオブジェクトバケットを要求し、バケットとアカウントを作成します。

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

```
        valueFrom:
          secretKeyRef:
            name: <obc-name>
            key: AWS_ACCESS_KEY_ID
      - name: AWS_SECRET_ACCESS_KEY
        valueFrom:
          secretKeyRef:
            name: <obc-name>
            key: AWS_SECRET_ACCESS_KEY
```

a. <obc-name> のすべてのインスタンスを、Object Bucket Claim（オブジェクトバケット要求）の名前に置き換えます。

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

3. 更新された YAML ファイルを適用します。

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

a. **<yaml.file>** を YAML ファイルの名前に置き換えます。

4. 新しい設定マップを表示するには、以下を実行します。

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

a. **obc-name** を、Object Bucket Claim（オブジェクトバケット要求）の名前に置き換えます。
出力には、以下の環境変数が表示されることが予想されます。

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

7.6.2. 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:
Wi9kcFluSWxHRzlWaFlzNk1hc0xma2JXcjM1MVhqa051SlBleXpmOQ==
kind: Secret
metadata:
  creationTimestamp: "2019-10-24T13:30:07Z"
  finalizers:
  - objectbucket.io/finalizer
  labels:
    app: noobaa
    bucket-provisioner: openshift-storage.noobaa.io-obc
    noobaa-domain: openshift-storage.noobaa.io
  name: test21obc
  namespace: openshift-storage
  ownerReferences:
  - apiVersion: objectbucket.io/v1alpha1
    blockOwnerDeletion: true
    controller: true
    kind: ObjectBucketClaim
    name: test21obc
    uid: 64f04cba-f662-11e9-bc3c-0295250841af
  resourceVersion: "40751"
  selfLink: /api/v1/namespaces/openshift-storage/secrets/test21obc
  uid: 65117c1c-f662-11e9-9094-0a5305de57bb
type: Opaque
```

シークレットは S3 アクセス認証情報を提供します。

5. 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。

7.6.3. 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

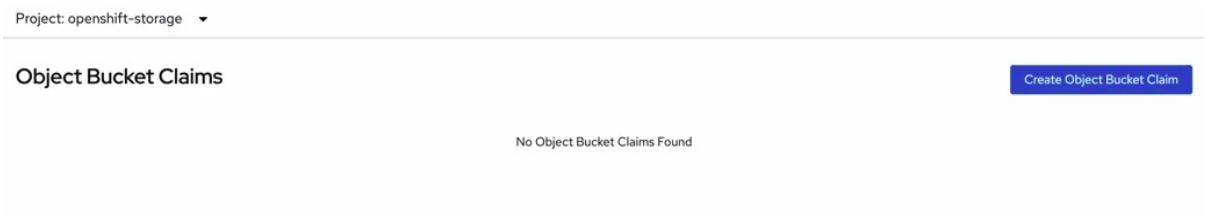
OpenShift Web コンソールを使用して Object Bucket Claim (オブジェクトバケット要求) を作成できます。

前提条件

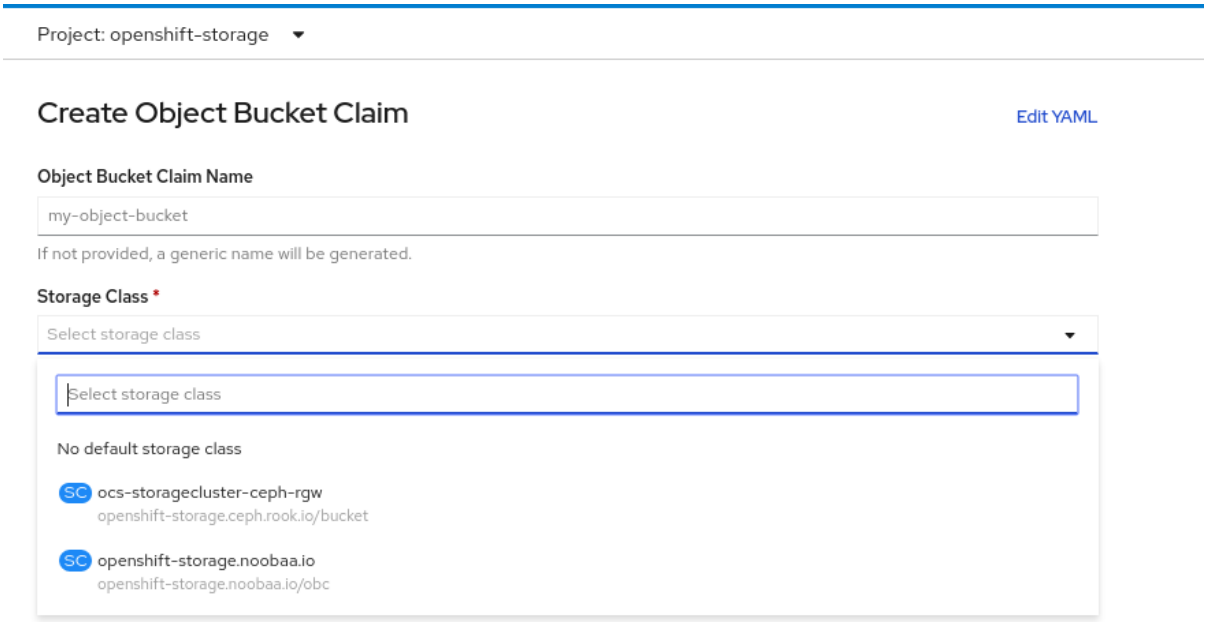
- OpenShift Web コンソールへの管理者アクセス。
- アプリケーションが OBC と通信できるようにするには、configmap およびシークレットを使用する必要があります。これについての詳細は、「動的 Object Bucket Claim（オブジェクトバケット要求）」 を参照してください。

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。



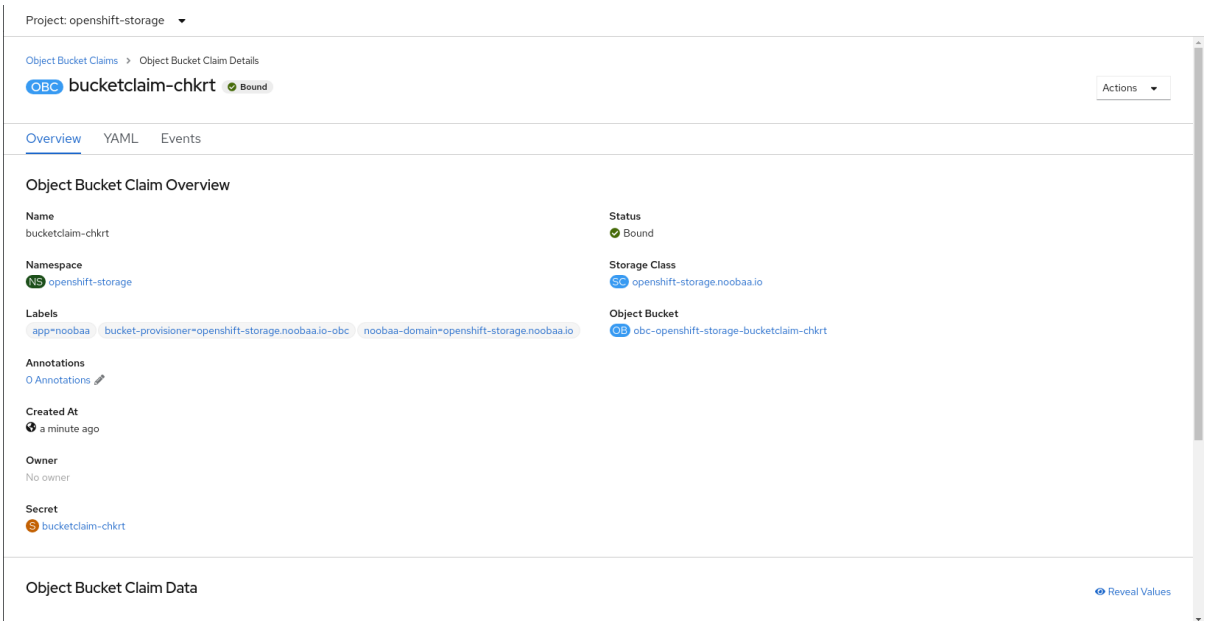
4. Object Bucket Claim（オブジェクトバケット要求）の名前を入力し、ドロップダウンメニューから適切なストレージクラスを選択します。



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。

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

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

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

- ストレージサービス
- S3 エンドポイントサービス

7.7.1. 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、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。

7.7.2. ストレージノードを使用したスケーリング

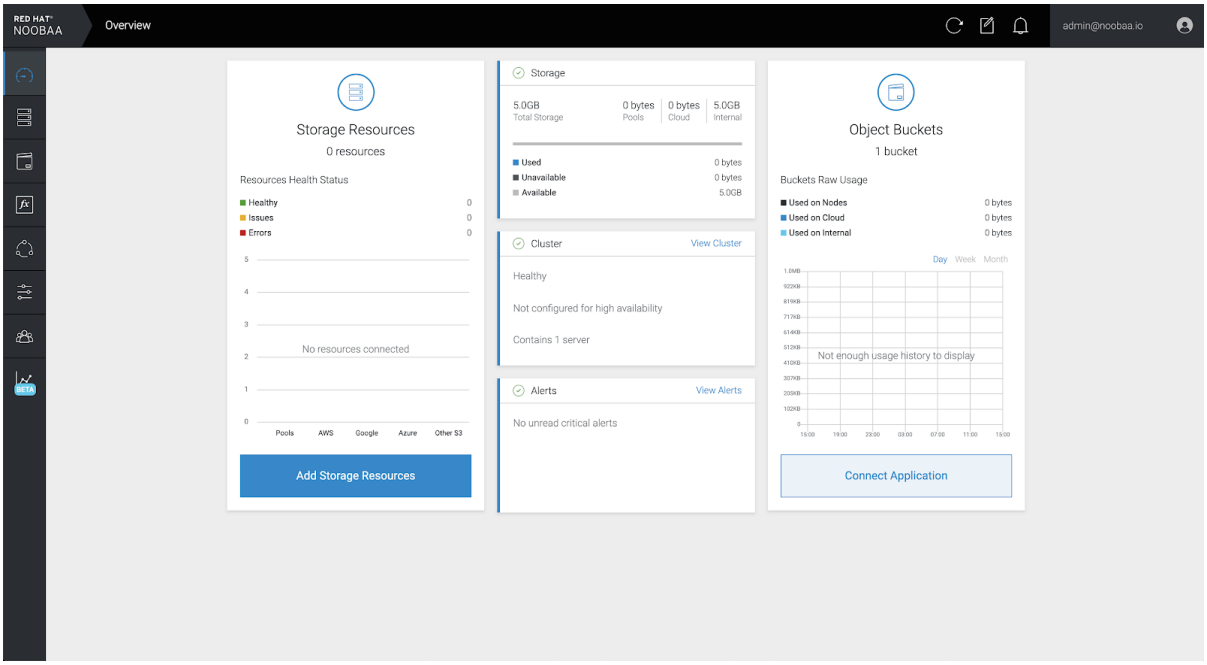
前提条件

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

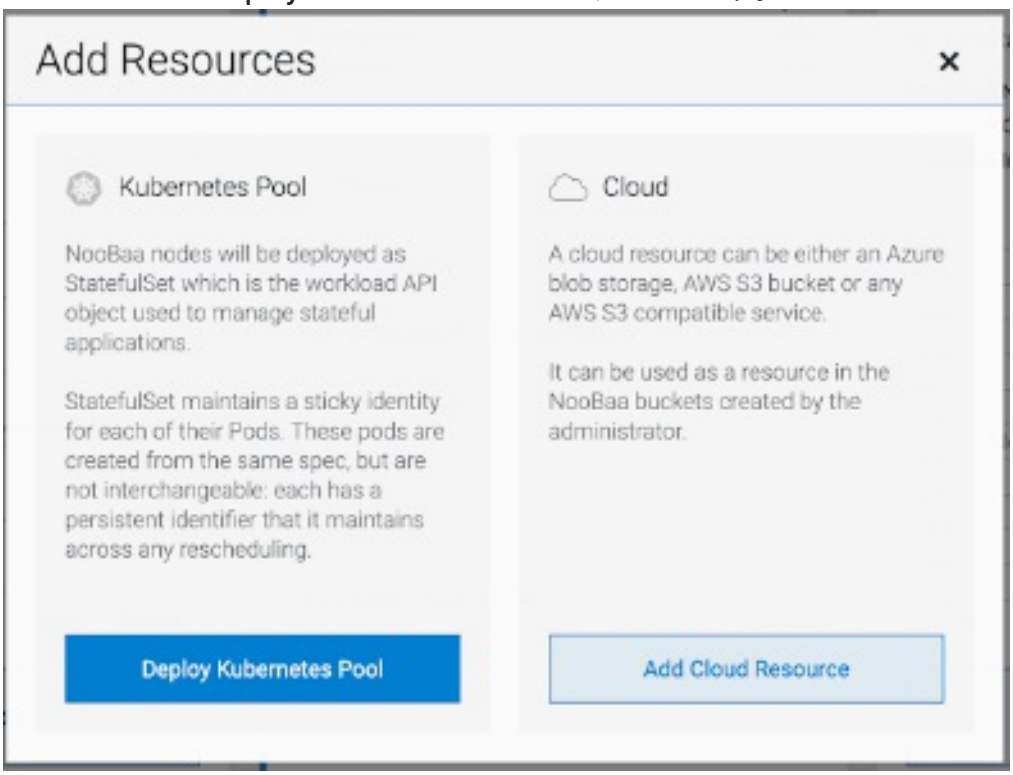
Multicloud Object Gateway のストレージノードは 1 つ以上の永続ボリュームに割り当てられた NooBaa デーモンコンテナーであり、ローカルオブジェクトサービスデータストレージに使用されます。NooBaa デーモンは Kubernetes ノードにデプロイできます。これは、StatefulSet Pod で構成される Kubernetes プールを作成して実行できます。

手順

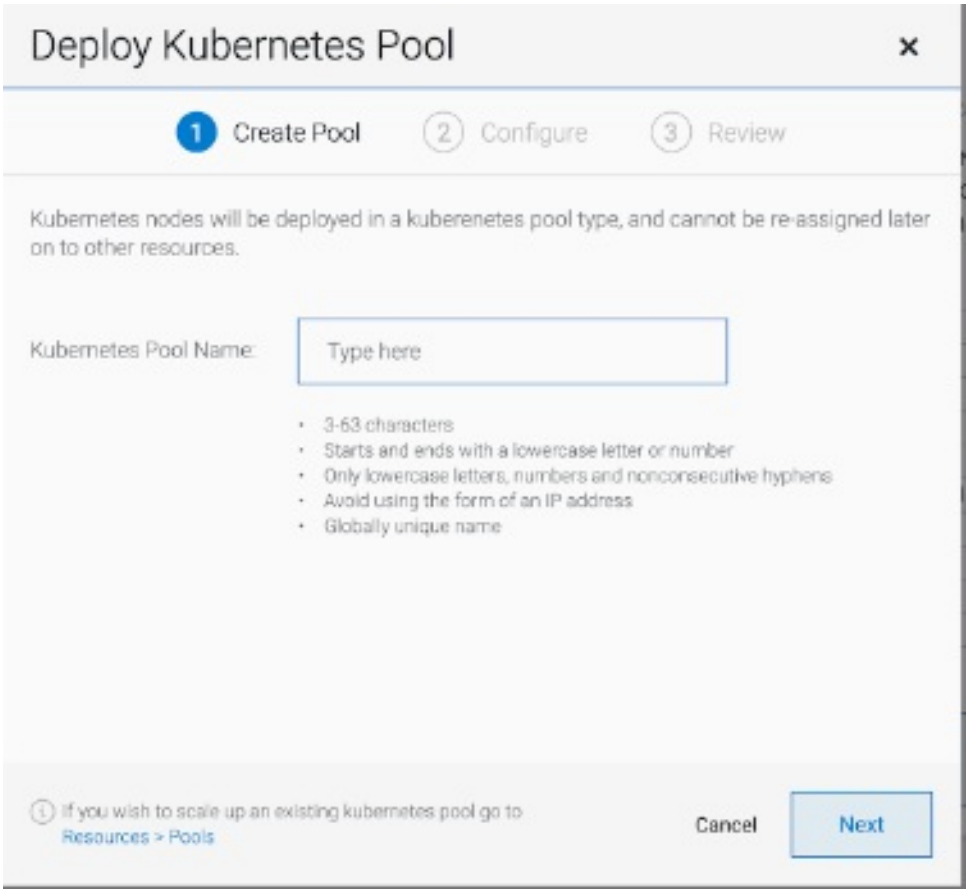
1. Mult-Cloud Object Gateway ユーザーインターフェースの **Overview** ページで、 **Add Storage Resources** をクリックします。



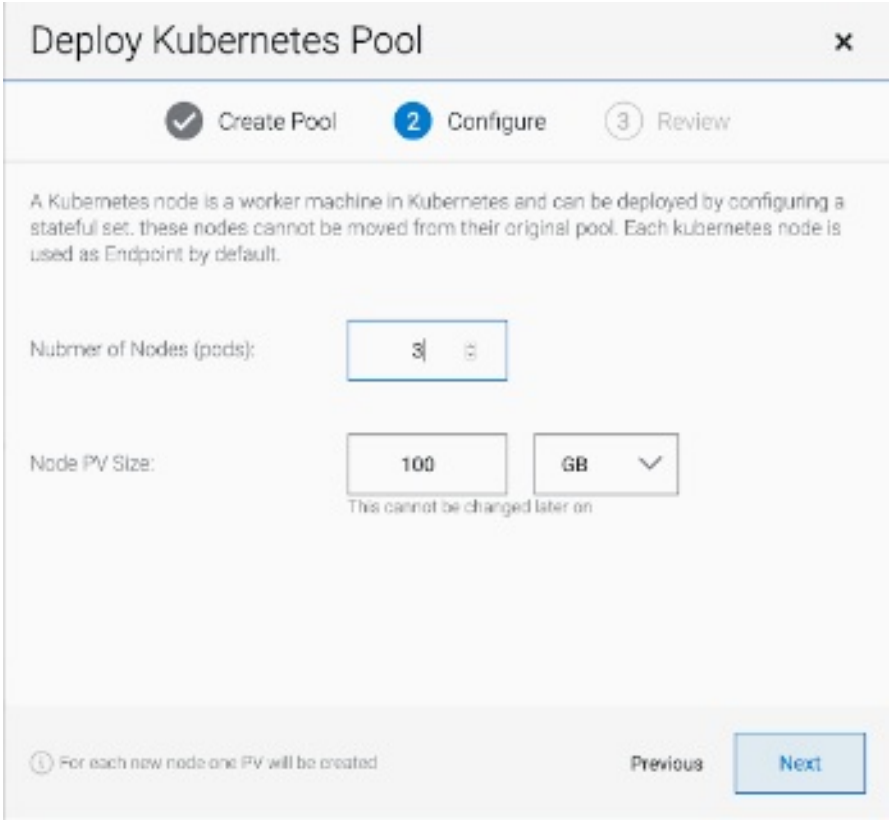
2. ウィンドウから **Deploy Kubernetes Pool** をクリックします。



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。

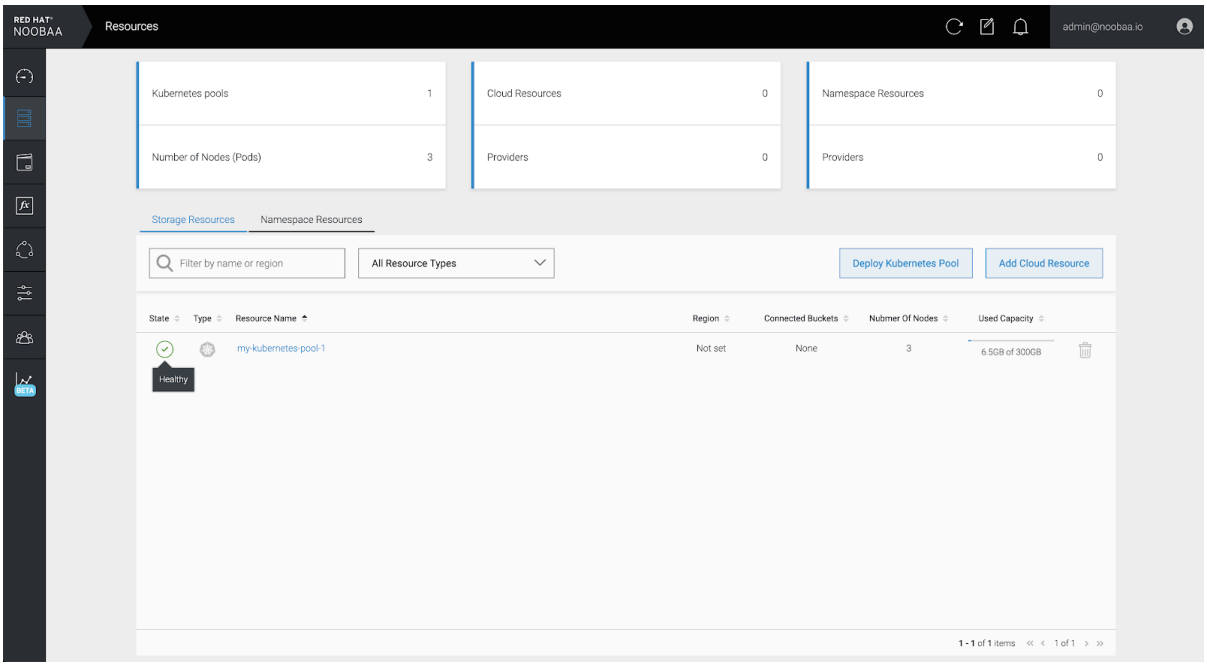


4. **Configure** 手順で、要求される Pod 数と各 PV のサイズを設定します。新規 Pod ごとに、1 つの PV が作成されます。



5. **Review** 手順で、新規プールの詳細を検索し、ローカルまたは外部デプロイメントのいずれかの使用するデプロイメント方法を選択します。ローカルデプロイメントが選択されている場合、Kubernetes ノードはクラスター内にデプロイされます。外部デプロイメントが選択されている場合、外部で実行するための YAML ファイルが提供されます。

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



第8章 永続ボリューム要求の管理

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

8.2. 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。

8.3. 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。

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

OpenShift Container Storage は Persistent Volume Claim（永続ボリューム要求、PVC）の拡張をサポートし、永続ストレージリソースの管理に柔軟性を提供します。

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

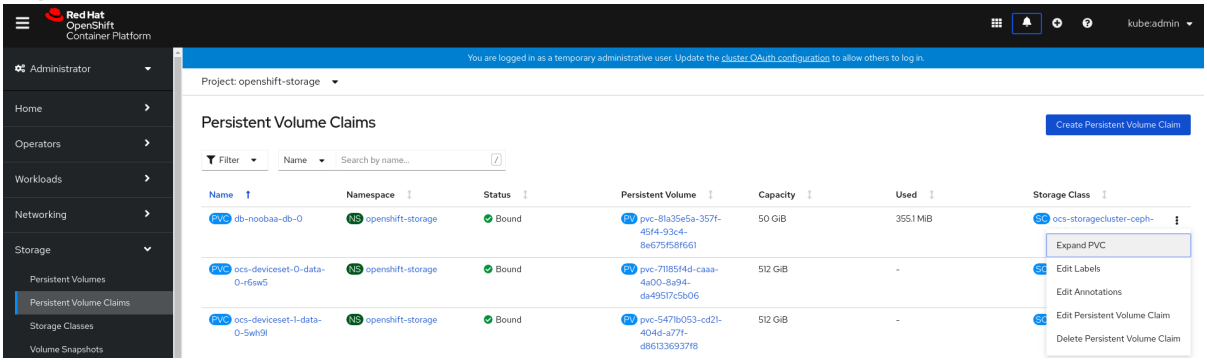
このテクノロジープレビュー機能は、OpenShift Container Storage バージョン 4.5 の新規インストールでのみ利用できます。これは、以前の OpenShift Container Storage リリースからアップグレードされたクラスターには適用されません。

前提条件

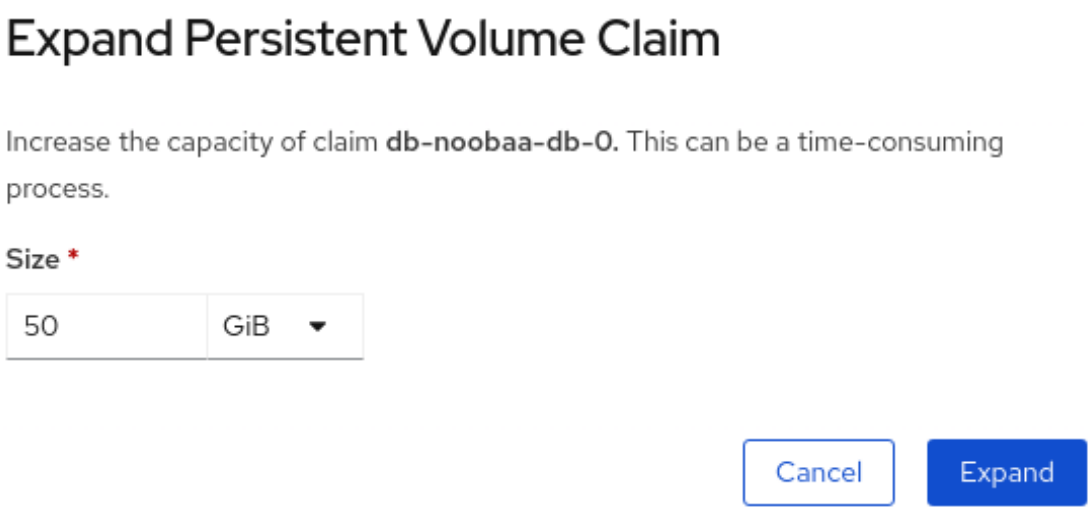
- OpenShift Web コンソールへの管理者アクセス。

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。



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。

8.5. 動的プロビジョニング

8.5.1. 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。

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

Red Hat OpenShift Container Storage は、コンテナー環境向けに最適化されたソフトウェアで定義されるストレージです。これは OpenShift Container Platform の Operator として実行され、コンテナーの統合され、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。

8.5.3. 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-ebs**	複数クラスターを複数の異なるゾーンで使用する際の動的プロビジョニングの場合、各ノードに **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>** のタグを付けます。ここで、**<cluster_name>** および **<cluster_id>** はクラスターごとに固有の値になります。

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 クラスターを実行し、現行クラスターのノードが存在しないゾーンで PV が作成されないようにすることが推奨されます。
VMware vSphere	**kubernetes.io/vsphere-volume**	

重要

選択したプロビジョナープラグインでは、関連するクラウド、ホスト、またはサードパーティープロバイダーを、関連するドキュメントに従って設定する必要もあります。

第9章 RED HAT VIRTUALIZATION プラットフォームでの障害のあるストレージノードの置き換え

OpenShift Container Storage の Red Hat Virtualization プラットフォームのストレージにより、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、Red Hat Virtualization 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。

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. ノードをドレイン (解放) します。

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

注記

障害が発生したノードがネットワークに接続されていない場合には、以下のコマンドを使用してそのノードで実行されている 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 ")}'
$ oc adm drain <node_name> --force --delete-local-data --ignore-daemonsets
```

6. **by-id** のデバイスをメモします。

   a. ノードを一覧表示します。

```
$ oc get nodes
```

出力例:

```
NAME         STATUS   ROLES    AGE     VERSION
rhvworker01    Ready    worker   6h45m   v1.16.2
rhvworker02    Ready    worker   6h45m   v1.16.2
rhvworker03    Ready    worker   6h45m   v1.16.2
```

b. 既存ノードの一意の **by-id** デバイス名を見つけます。

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

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
sdb                          8:16   0   500G  0 disk
sr0                         11:0    1   374K  0 rom
sr1                         11:1    1  1024M  0 rom
rbd0                        252:0    0    30G  0 disk /var/lib/kubelet/pods/1fe52fb8-ca70-40e1-
ac74-28802baf3b80/volumes/kubernetes.io~csi/pvc-705982e5-0a7d-42f5-916f-
69340541d52d/mount
rbd1                        252:16   0    30G  0 disk /var/lib/kubelet/pods/b48b22ba-532f-4d4f-
9f58-debfd03c3221/volumes/kubernetes.io~csi/pvc-bb7cdb1c-03fe-4b31-97e3-
3c8fa9758f16/mount
sh-4.4# ls -l /dev/disk/by-id/ | grep sdb
lrwxrwxrwx. 1 root root  9 Oct  7 14:58 scsi-0QEMU_QEMU_HARDDISK_a01cab90-
d3cd-4822-a7eb-a5553d4b619a -> ../../sdb
lrwxrwxrwx. 1 root root  9 Oct  7 14:58 scsi-SQEMU_QEMU_HARDDISK_a01cab90-
d3cd-4822-a7eb-a5553d4b619a -> ../../sdb
sh-4.4# exit
exit
sh-4.2# exit
exit
Removing debug pod ...
```

これを繰り返して、すべての既存ノードのデバイス ID を特定します。

c. 障害のあるノードに対応するマシンを削除します。新しいノードが自動的に追加されます。

   i. **Compute → Machines** をクリックします。必要なマシンを検索します。

   ii. 必要なマシンの横にある Action メニュー (⋮)→ **Delete Machine** をクリックします。

   iii. **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。

   iv. 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。

d. [オプション]: 障害のある Red Hat Virtualization インスタンスが自動的に削除されない場合、インスタンスをコンソールで終了します。

7. 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。

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、OpenS+hift 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. 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。**by-id** のデバイスを特定する方法は、「 利用可能なストレージデバイス の検索」を参照してください。

```
$ 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/scsi-SQEMU_QEMU_HARDDISK_901abab1-b164-4bb4-b9a8-
6bdbce2de23b
      - /dev/disk/by-id/scsi-SQEMU_QEMU_HARDDISK_a01cab90-d3cd-4822-a7eb-
a5553d4b619a
        # SQEMU_QEMU_HARDDISK_a01cab90-d3cd-4822-a7be-a5553d4b619a
      - /dev/disk/by-id/scsi-SQEMU_QEMU_HARDDISK_c7388957-3e09-4135-bac1-
```

```
af7551467d0b

      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** を使用する以下の新規デバイスが追加されていることを確認できます。

SQEMU_QEMU_HARDDISK_a01cab90-d3cd-4822-a7eb-a5553d4b619a

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
...
```

10. 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:

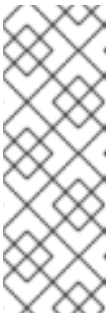
```
persistentvolume "local-pv-8176b2bf" deleted
```

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
```

11. 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
```

12. **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。

13. **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。

第10章 RED HAT VIRTUALIZATION プラットフォームでの障害のあるストレージデバイスの置き換え

Red Hat Virtualization プラットフォーム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、Red Hat Virtualization プラットフォームでの障害のあるストレージノードの置き換えについて参照してください。