Red Hat OpenShift Container Storage 4.8

Deploying and managing OpenShift Container Storage using Google Cloud

설치 및 관리 방법

Last Updated: 2023-04-08

Red Hat OpenShift Container Storage 4.8 Deploying and managing OpenShift Container Storage using Google Cloud

설치 및 관리 방법

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.

법적 공지

초록

이 문서에서는 Google Cloud에서 Red Hat OpenShift Container Storage를 설치 및 관리하는 방법에 대한 지침을 읽어 보십시오. Deploying and managing OpenShift Container Storage on Google Cloud is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

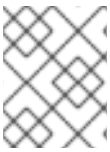
RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 이를 개선하는 방법을 알려 주십시오. 피드백을 제공하려면 다음을 수행하십시오.

- 특정 문구에 대한 간단한 주석은 다음과 같습니다.
  1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **Feedback** (피드백) 버튼이 있는지 확인합니다.
  2. 마우스 커서를 사용하여 주석 처리하려는 텍스트 부분을 강조 표시합니다.
  3. 강조 표시된 텍스트 아래에 표시되는 **피드백 추가** 팝업을 클릭합니다.
  4. 표시된 지침을 따릅니다.
- 보다 상세하게 피드백을 제출하려면 다음과 같이 Bugzilla 티켓을 생성하십시오.
  1. Bugzilla 웹 사이트로 이동하십시오.
  2. Component로 **Documentation**을 선택하십시오.
  3. **Description** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
  4. **Submit Bug**를 클릭하십시오.

PREFACE

Red Hat OpenShift Container Storage 4.8에서는 기존 RHOCP(OpenShift Container Platform) Google Cloud 클러스터에서 배포를 지원합니다.



참고

Google Cloud에서는 내부 Openshift Container Storage 클러스터만 지원됩니다. 배포 요구 사항에 대한 자세한 내용은 배포 계획을 참조하십시오.

OpenShift Container Storage를 내부 모드로 배포하려면 OpenShift Container Storage 배포 장에서 요구 사항으로 시작한 다음 배포 프로세스를 따르십시오. Google Cloud에 OpenShift Container Storage 배포 .

1장. OPENSHIFT CONTAINER STORAGE 배포 준비

동적 스토리지 장치를 사용하여 OpenShift Container Platform에 OpenShift Container Storage를 배포하면 내부 클러스터 리소스를 생성할 수 있는 옵션이 제공됩니다. 이렇게 하면 기본 서비스의 내부 프로비저닝이 생성되므로 애플리케이션에서 추가 스토리지 클래스를 사용할 수 있습니다.

Red Hat OpenShift Container Storage 배포를 시작하기 전에 다음 단계를 따르십시오.

1. 선택 사항: 외부 키 관리 시스템(KMS)을 사용하여 클러스터 전체 암호화를 활성화하려면 다음을 수행합니다.
   - 토큰이 있는 정책이 존재하고 Vault의 키 값 백엔드 경로가 활성화되어 있는지 확인합니다. Vault의 키 값 백엔드 경로 및 정책 활성화를 참조하십시오.
   - Vault 서버에서 서명된 인증서를 사용 중인지 확인합니다.
2. 최소 시작 노드 요구 사항 [기술 프리뷰]
   표준 배포 리소스 요구 사항이 충족되지 않은 경우 OpenShift Container Storage 클러스터는 최소 구성으로 배포됩니다. 계획 가이드의 리소스 요구 사항 섹션을 참조하십시오.

1.1. VAULT에서 키 값 백엔드 경로 및 정책 활성화

사전 요구 사항

- Vault에 대한 관리자 액세스.
- 나중에 변경할 수 없으므로 이름 지정 규칙을 따르는 고유한 **경로 이름을 백엔드 경로로** 선택합니다.

절차

1. Vault에서 KV(키/값) 백엔드 경로를 활성화합니다.
   Vault KV 비밀 엔진 API의 경우 버전 1입니다.

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

   Vault KV 비밀 엔진 API의 경우 버전 2입니다.

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

2. 다음 명령을 사용하여 시크릿에서 쓰기 또는 삭제 작업을 수행하도록 사용자를 제한하는 정책을 생성합니다.

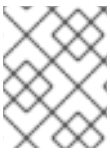
   ```
   echo '
   path "ocs/*" {
     capabilities = ["create", "read", "update", "delete", "list"]
   }
   path "sys/mounts" {
   capabilities = ["read"]
   }'| vault policy write ocs -
   ```

3. 위 정책과 일치하는 토큰을 생성합니다.

```
$ vault token create -policy=ocs -format json
```

2장. GOOGLE 클라우드에 OPENSHIFT CONTAINER STORAGE 배포

Google Cloud 설치 관리자 프로비저닝 인프라(IPI)에서 제공하는 동적 스토리지 장치를 사용하여 OpenShift Container Platform에 OpenShift Container Storage를 배포하면 내부 클러스터 리소스를 생성할 수 있습니다. 이로 인해 기본 서비스의 내부 프로비저닝이 생성되어 애플리케이션에서 추가 스토리지 클래스를 사용할 수 있습니다.



참고

Google Cloud에서는 내부 Openshift Container Storage 클러스터만 지원됩니다. 배포 요구 사항에 대한 자세한 내용은 배포 계획을 참조하십시오.

동적 스토리지 장치를 사용하여 배포하기 위한 아래 단계를 진행하기 전에 OpenShift Container Storage 배포 준비 장에서 요구 사항을 해결했는지 확인하십시오.

1. Red Hat OpenShift Container Storage Operator를 설치합니다 .
2. OpenShift Container Storage 클러스터 서비스 생성

2.1. RED HAT OPENSHIFT CONTAINER STORAGE OPERATOR 설치

Red Hat OpenShift Container Platform Operator Hub를 사용하여 Red Hat OpenShift Container Storage Operator를 설치할 수 있습니다.

사전 요구 사항

- cluster-admin 및 Operator 설치 권한이 있는 계정을 사용하여 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.
- Red Hat OpenShift Container Platform 클러스터에 작업자 노드가 3개 이상 있습니다.
- 필요한 추가 요구 사항을 충족했습니다. 자세한 내용은 배포 계획을 참조하십시오.

참고

- OpenShift Container Storage의 클러스터 수준 기본 노드 선택기를 재정의해야 하는 경우 다음 명령을 사용하여 **openshift-storage** 네임스페이스에 대한 빈 노드 선택기를 지정할 수 있습니다(이 경우 openshift-storage 네임스페이스 생성):

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

- 노드를 인프라로 테인트 **하여** Red Hat OpenShift Container Storage 리소스만 해당 노드에 예약되도록 합니다. 이를 통해 서브스크립션 비용을 절감할 수 있습니다. 자세한 내용은 스토리지 리소스 관리 및 할당 가이드 의 Red Hat OpenShift Container Storage 전용 작업자 노드를 사용하는 방법을 참조하십시오.

절차

1. OpenShift 웹 콘솔에 로그인합니다.
2. **Operators → OperatorHub**를 클릭합니다.

3. Operator 목록에서 **OpenShift Container Storage** 를 검색하고 클릭합니다.
4. **설치**를 클릭합니다.
5. **Operator 설치** 페이지에서 다음 옵션을 설정합니다.
   a. 채널은 **stable-4.8입니다**.
   b. **클러스터의 특정 네임스페이스로서의 설치 모드입니다**.
   c. **Operator 권장 네임스페이스 openshift-storage로 설치된 네임스페이스** 입니다. 네임스페이스 **openshift-storage** 가 없으면 운영자 설치 중에 생성됩니다.
   d. **자동** 또는 **수동 승인 전략**
   e. **설치**를 클릭합니다.
   자동 업데이트를 선택하면 OLM(Operator Lifecycle Manager)이 개입 없이 Operator의 실행 중인 인스턴스를 자동으로 업그레이드합니다.

   수동 업데이트를 선택하면 OLM에서 업데이트 요청을 생성합니다. 클러스터 관리자는 Operator를 새 버전으로 업데이트하려면 해당 업데이트 요청을 수동으로 승인해야 합니다.

검증 단계

- **OpenShift Container Storage** Operator에 설치에 성공한 녹색 눈금이 표시되는지 확인합니다.

2.2. 내부 모드에서 OPENSHIFT CONTAINER STORAGE 클러스터 서비스 생성

OpenShift Container Storage Operator를 설치한 후 다음 절차를 사용하여 OpenShift Container Storage 클러스터 서비스를 생성합니다.

사전 요구 사항

- OpenShift Container Storage Operator는 Operator Hub에서 설치해야 합니다. 자세한 내용은 Operator Hub를 사용하여 OpenShift Container Storage Operator 설치를 참조하십시오.
- Google Cloud의 기본 스토리지 클래스는 하드 디스크 드라이브(HDD)를 사용합니다. 성능 향상을 위해 SSD(Solid State Drive) 기반 디스크를 사용하려면 다음 **ssd-storeageclass.yaml 예에 표시된 pd-ssd 를 사용하여 스토리지 클래스를 생성해야 합니다.**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: faster
provisioner: kubernetes.io/gce-pd
parameters:
 type: pd-ssd
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
```

절차

1. OpenShift 웹 콘솔에 로그인합니다.

2. **Operators → 설치된 Operator**를 클릭하여 설치된 모든 Operator를 확인합니다.
   선택한 프로젝트가 **openshift-storage** 인지 확인합니다.

3. 스토리지 클러스터의 **OpenShift Container Storage > 인스턴스 생성** 링크를 클릭합니다.

4. **모드는 기본적으로 Internal 로 설정됩니다.**

5. **용량 및 노드**선택

   a. **스토리지 클래스** 선택. 기본적으로 **standard** 로 설정됩니다. 그러나 더 나은 성능을 위해 SSD 기반 디스크를 사용하기 위해 스토리지 클래스를 생성한 경우 해당 스토리지 클래스를 선택해야 합니다.

   b. 드롭다운 목록에서 **Requested Capacity** (요청된 용량)를 선택합니다. 기본적으로 **2TiB** 로 설정됩니다. 드롭다운을 사용하여 용량 값을 수정할 수 있습니다.

   참고

   초기 스토리지 용량을 선택하면 선택한 사용 가능한 용량(원리 스토리지의 3 배)만 클러스터 확장이 수행됩니다.

   c. **Select Nodes** (노드 선택) 섹션에서 사용 가능한 노드 3개를 선택합니다.
   여러 가용 영역이 있는 클라우드 플랫폼의 경우 노드가 다양한 위치/가용성 영역에 분산되어 있는지 확인합니다.

   선택한 노드가 집계된 30 개의 CPU 및 72GiB RAM의 OpenShift Container Storage 클러스터 요구 사항과 일치하지 않으면 최소 클러스터가 배포됩니다. 최소 노드 요구 사항은 계획 가이드의 리소스 요구 사항 섹션을 참조하십시오.

   d. **다음**을 클릭합니다.

6. (선택 사항) **보안 및 네트워크** 설정 설정

   a. **Enable encryption(암호화 사용)** 확인란을 선택하여 블록 및 파일 스토리지를 암호화합니다.

   b. 하나 또는 둘 다 **암호화 수준을** 선택하십시오 :

   - **전체 클러스터(블록 및 파일)를 암호화** 하기 위한 클러스터 전체 암호화.
   - **암호화가 활성화된 스토리지 클래스를** 사용하여 암호화된 영구 볼륨을 만드는 스토리지 클래스 암호화(블록만 해당).

   c. **Connect to an external key management service**확인란을 선택합니다. 이는 클러스터 전체 암호화에 대해 선택 사항입니다.

      i. **Key Management Service Provider**는 기본적으로 **Vault** 로 설정됩니다.

      ii. Vault **Service Name**, host **Address** of Vault server (https://<hostname 또는 ip>'), **Port number** and **Token** 을 입력합니다.

      iii. **고급 설정을 확장하여 Vault 구성에 따라 추가 설정 및 인증서 세부 정보를 입력합니다**

         A. OpenShift Container Storage 전용으로 **고유한 백엔드** 경로에 Key Value 보안 경로를 입력합니다.

         B. (선택 사항) **TLS 서버 이름과 Vault Enterprise 네임 스페이스를** 입력합니다.

C. 각 PEM 인코딩 **인증서 파일을 업로드하여 CA 인증서, 클라이언트 인증서 및 클라이언트 개인** 키를 제공합니다.

D. **저장**을 클릭합니다.

7. 여러 네트워크 인터페이스를 사용하려는 경우 단일 네트워크 또는 **사용자 지정** (Multus) 네트워크를 사용하는 경우 **Default** (SDN)를 선택합니다.

   a. 드롭다운에서 **Public Network Interface** (공용 네트워크 인터페이스)를 선택합니다.

   b. 드롭다운에서 **Cluster Network Interface** (클러스터 네트워크 인터페이스)를 선택합니다.

참고

추가 네트워크 인터페이스를 하나만 사용하는 경우 공용 네트워크 인터페이스의 단일 NetworkAttachementDefinition(예: ocs-public-cluster)을 선택하고 Cluster Network Interface를 비워 둡니다.

8. **다음**을 클릭합니다.

9. 구성 세부 정보를 검토합니다. 구성 설정을 수정하려면 **Back(뒤로)**을 클릭하여 이전 구성 페이지로 돌아갑니다.

10. **생성**을 클릭합니다.

11. KV(Vault Key/Value) 시크릿 엔진 API, 버전 2가 KMS(Key Management System)로 클러스터 전체 암호화에 사용되는 경우 configmap을 편집합니다.

    a. OpenShift 웹 콘솔에서 **워크로드 → ConfigMaps** 로 이동합니다.

    b. KMS 연결 세부 정보를 보려면 **ocs-kms-connection-details** 를 클릭합니다.

    c. configmap을 편집합니다.

       i. **작업 메뉴 (TI) → ConfigMap 편집을**클릭합니다.

       ii. **VAULT_BACKEND** 매개 변수를 **v2** 로 설정합니다.

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

       iii. **저장**을 클릭합니다.

검증 단계

1. 스토리지 클러스터 세부 정보 페이지에서 스토리지 클러스터 이름에 클러스터가 성공적으로 생성되었음을 나타내기 위해 옆에 녹색 눈금이 표시됩니다.

2. 설치된 스토리지 클러스터의 최종 **상태가 단계로 표시되는지 확인합니다. 녹색 눈금 표시가 있는 ready** 입니다.

   - **Operators → 설치된 Operator → 스토리지 클러스터**링크를 클릭하여 스토리지 클러스터 설치 상태를 확인합니다.

   - 또는 Operator **Details(운영자 세부 정보)** 탭에 있는 경우 **Storage Cluster(스토리지 클러스터)** 탭을 클릭하여 상태를 확인할 수 있습니다.

3. OpenShift Container Storage의 모든 구성 요소가 성공적으로 설치되었는지 확인하려면 OpenShift Container Storage 설치 확인을 참조하십시오.

3장. OPENSHIFT CONTAINER STORAGE 배포 확인

이 섹션을 사용하여 OpenShift Container Storage가 올바르게 배포되었는지 확인합니다.

3.1. POD 상태 확인

OpenShift Container Storage의 포드가 running 상태인지 확인하려면 다음 절차를 따르십시오.

절차

1. OpenShift 웹 콘솔에 로그인합니다.
2. OpenShift 웹 **콘솔의 왼쪽 창에서 워크로드 →**포드를 클릭합니다.
3. **프로젝트** 드롭다운 목록에서 **openshift-storage** 를 선택합니다.
   각 구성 요소에 대해 예상되는 Pod 수 및 노드 수에 따라 어떻게 달라지는지에 대한 자세한 내용은 표 3.1. "OpenShift Container 스토리지 클러스터에 해당하는 Pod" 을 참조하십시오.
4. **Running** 및 **Completed** 탭을 클릭하여 Pod가 실행 중이고 완료된 상태인지 확인합니다.

표 3.1. OpenShift Container 스토리지 클러스터에 해당하는 Pod

구성 요소	해당 Pod
OpenShift Container Storage Operator	• **oCS-operator-*** (모든 작업자 노드에서 Pod 1개) • **ocs-metrics-exporter-***
Rook-ceph Operator	**rook-ceph-operator-*** 모든 작업자 노드에서 Pod 1개)
Multicloud Object Gateway	• **NooBaa-operator-*** (작업자 노드에서 Pod 1개) • **NooBaa-core-*** (모든 스토리지 노드에서 Pod 1개) • **NooBaa-db-pg-*** (모든 스토리지 노드에 1 Pod) • **NooBaa-endpoint-*** (모든 스토리지 노드에서 Pod 1개)
MON	**rook-ceph-mon-*** 스토리지 노드에 분산된 3개의 Pod)
MGR	**rook-ceph-mgr-*** 모든 스토리지 노드에서 Pod 1개)

구성 요소	해당 Pod
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** 스토리지 노드에 분산된 2개의 Pod)

CSI	• **CephFS**
	  ◦ **CSI-cephfsplugin-*** (1개의 작업자 노드에서 Pod)
	  ◦ **CSI-cephfsplugin-provisioner-*** (작업자 노드에 분산된 2 Pod)
	• **rbd**
	  ◦ **CSI-rbdplugin-*** (각 작업자 노드에서 Pod 1개)
	  ◦ **CSI-rbdplugin-provisioner-*** (2 작업자 노드에 분산된 2 Pod)

rook-ceph-crashcollector	**rook-ceph-crashcollector-*** 각 스토리지 노드에서 Pod 1개)
OSD	• **Rook-ceph-osd-*** (1개의 장치마다 pod) • **Rook-ceph-osd-prepare-ocs-deviceset-*** (1개의 장치 용 pod)

3.2. OPENSHIFT CONTAINER STORAGE 클러스터가 정상인지 확인

OpenShift Container Storage 클러스터가 정상인지 확인하려면 절차의 단계를 따르십시오.

절차

1. **스토리지 → 개요**를 클릭하고 **블록 및 파일** 탭을 클릭합니다.
2. **상태 카드에서** *Storage Cluster* (스토리지 클러스터) 및 *Data Resiliency* (데이터 복원력)에 녹색 눈금이 있는지 확인합니다.
3. **세부 정보 카드에서** 클러스터 정보가 표시되는지 확인합니다.

블록 및 파일 대시보드를 사용하는 OpenShift Container Storage 클러스터의 상태에 대한 자세한 내용은 Monitoring OpenShift Container Storage 를 참조하십시오.

3.3. MULTICLOUD OBJECT GATEWAY가 정상인지 확인

OpenShift Container Storage Multicloud Object Gateway가 정상인지 확인하려면 절차의 단계를 따르십시오.

절차

1. OpenShift 웹 콘솔에서 **스토리지 → 개요**를 클릭하고 **오브젝트** 탭을 클릭합니다.
2. **상태 카드에서** *오브젝트 서비스 및 데이터 복원성 이 모두* Ready **상태(녹색 틱)인지 확인합니다.**
3. **세부 정보 카드에서 Multicloud Object Gateway 정보가 표시되는지 확인합니다.**

오브젝트 서비스 대시보드를 사용하는 OpenShift Container Storage 클러스터의 상태에 대한 자세한 내용은 OpenShift Container Storage 모니터링을 참조하십시오.

3.4. OPENSHIFT CONTAINER STORAGE 특정 스토리지 클래스가 있는지 확인

클러스터에 스토리지 클래스가 있는지 확인하려면 절차의 단계를 따르십시오.

절차

1. **OpenShift 웹 콘솔에서 스토리지 → 스토리지 클래스 를 클릭합니다.**
2. **OpenShift Container Storage 클러스터 생성을 사용하여 다음 스토리지 클래스가 생성되었는지 확인합니다.**
   - **ocs-storagecluster-ceph-rbd**
   - **ocs-storagecluster-cephfs**
   - **openshift-storage.noobaa.io**

4장. OPENSHIFT CONTAINER STORAGE 설치 제거

4.1. 내부 모드에서 OPENSHIFT CONTAINER STORAGE 설치 제거

이 섹션의 단계를 사용하여 **OpenShift Container Storage**를 제거합니다.

주석 제거

스토리지 클러스터의 주석은 제거 프로세스의 동작을 변경하는 데 사용됩니다. 설치 제거 동작을 정의하기 위해 스토리지 클러스터에 다음 두 개의 주석이 도입되었습니다.

- **uninstall.ocs.openshift.io/cleanup-policy: delete**
- **uninstall.ocs.openshift.io/mode: 정상**

아래 표는 이러한 주석과 함께 사용할 수 있는 다양한 값에 대한 정보를 제공합니다.

표 4.1. uninstall.ocs.openshift.io 제거 주석 설명

주석	현재의	Default	동작
cleanup-policy	delete	있음	Rook은 물리적 드라이브와 **DataDirHostPath**를 정리합니다.
cleanup-policy	유지	없음	Rook은 물리적 드라이브 및 **DataDirHostPath**를 정리 **하지** 않습니다.
mode	정상	있음	Rook 및 NooBaa는 PVC와 OBC가 관리자/사용자가 제거할 때까지 제거 프로세스를 **일시** 중지합니다.
mode	강제	없음	Rook 및 NooBaa는 Rook 및 NooBaa를 사용하여 프로비저닝된 PVC/OBC가 각각 존재하는 경우에도 제거를 진행합니다.

다음 명령을 사용하여 주석의 값을 편집하여 정리 정책 또는 설치 제거 모드를 변경할 수 있습니다.

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

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

사전 요구 사항

- **OpenShift Container Storage** 클러스터가 정상 상태인지 확인합니다. 리소스 또는 노드가 부족하여 일부 **Pod**가 성공적으로 종료되지 않으면 설치 제거 프로세스가 실패할 수 있습니다. 클러스터가 비정상인 경우 **OpenShift Container Storage**를 설치 제거하기 전에 **Red Hat** 고객 지원팀에 문의하십시오.

- 애플리케이션이 **OpenShift Container Storage**에서 제공하는 스토리지 클래스를 사용하여 **PVC(**영구 볼륨 클레임**)** 또는 개체 버킷 클레임**(OBC)**을 사용하지 않는지 확인합니다.

- 관리자가 생성한 사용자 지정 리소스**(**예**:** 사용자 지정 스토리지 클래스**, cephblockpools)**가 관리자가 생성한 경우 해당 리소스를 사용하는 리소스를 제거한 후 관리자가 삭제해야 합니다.

절차

1. **OpenShift Container Storage**를 사용하는 볼륨 스냅샷을 삭제합니다.

   a. 모든 네임스페이스의 볼륨 스냅샷을 나열합니다.

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

   b. 이전 명령의 출력에서 **OpenShift Container Storage**를 사용하는 볼륨 스냅샷을 식별하고 삭제합니다.

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

2. **OpenShift Container Storage**를 사용하는 **PVC** 및 **OBC**를 삭제합니다.

기본 설치 제거 모드(허용)에서 설치 제거 프로그램은 **OpenShift Container Storage**를 사용하는 모든 **PVC** 및 **OBC**가 삭제됩니다.

PVC를 삭제하지 않고 스토리지 클러스터를 삭제하려면 **uninstall** 모드 주석을 강제 설정하고 이 단계를 건너뛸 수 있습니다. 이렇게 하면 시스템의 고립된 **PVC**와 **OBC**가 생성됩니다.

a. **OpenShift Container Storage**를 사용하여 **OpenShift Container Platform** 모니터링 스택 **PVC**를 삭제합니다.

자세한 내용은 **4.2절. "OpenShift Container Storage에서 모니터링 스택 제거"**의 내용을 참조하십시오.

b. **OpenShift Container Storage**를 사용하여 **OpenShift Container Platform** 레지스트리 **PVC**를 삭제합니다.

자세한 내용은 **4.3절. "OpenShift Container Storage에서 OpenShift Container Platform 레지스트리 제거"**의 내용을 참조하십시오.

c. **OpenShift Container Storage**를 사용하여 **OpenShift Container Platform** 로깅 **PVC**를 삭제합니다.

자세한 내용은 **4.4절. "OpenShift Container Storage에서 클러스터 로깅 Operator 제거"**의 내용을 참조하십시오.

d. **OpenShift Container Storage**를 사용하여 프로비저닝된 다른 **PVC** 및 **OBC**를 삭제합니다.

- 다음 스크립트는 **OpenShift Container Storage**를 사용하여 프로비저닝된 **PVC** 및 **OBC**를 식별하는 샘플 스크립트입니다. 스크립트는 **Openshift Container Storage**에서 내부적으로 사용하는 **PVC**를 무시합니다.

```
#!/bin/bash

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

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

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

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

참고

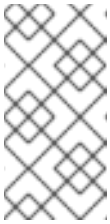
클라우드 플랫폼용 **RGW_PROVISIONER** 생략.

- **OBC**를 삭제합니다.

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

- **PVC**를 삭제합니다.

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



참고

클러스터에서 생성된 사용자 정의 백업 저장소, 버킷 클래스 등을 제거했는지 확인합니다.

3. **Storage Cluster** 오브젝트를 삭제하고 관련 리소스가 제거될 때까지 기다립니다.

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

4. **uninstall.ocs.openshift.io/cleanup-policy 가 삭제(기본값)로 설정되어 있는지 확인하고 상태가 Completed 인지 확인합니다.**

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

5. **/var/lib/rook 디렉토리가 비어 있는지 확인합니다. 이 디렉터리는 uninstall.ocs.openshift.io/cleanup-policy 주석이 삭제(기본값)로 설정된 경우에만 비어 있습니다.**

```
$ 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. **설치 시 암호화가 활성화된 경우 모든 OpenShift Container Storage 노드의 OSD 장치에서 dm -crypt managed device-mapper 매핑을 제거합니다.**

   a. **스토리지 노드의 호스트에 디버그 포드 및 chroot 를 만듭니다.**

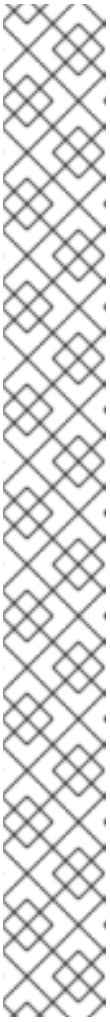
```
$ oc debug node/<node name>
$ chroot /host
```

   b. **장치 이름을 가져오고 OpenShift Container Storage 장치를 기록합니다.**

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

   c. **매핑된 장치를 제거합니다.**

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



참고

권한이 충분하지 않아 위의 명령이 중단되면 다음 명령을 실행합니다.

- **CTRL+Z** 를 눌러 위의 명령을 종료합니다.
- 중단된 프로세스의 **PID**를 찾습니다.

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

- **kill** 명령을 사용하여 프로세스를 종료합니다.

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

- 장치 이름이 제거되었는지 확인합니다.

```
$ dmsetup ls
```

7. 네임스페이스를 삭제하고 삭제가 완료될 때까지 기다립니다. **openshift-storage** 가 활성 프로젝트인 경우 다른 프로젝트로 전환해야 합니다.

예를 들면 다음과 같습니다.

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

다음 명령에서 **NotFound** 오류를 반환하면 프로젝트가 삭제됩니다.

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

참고

OpenShift Container Storage를 설치 제거하는 동안 네임스페이스 가 완전히 삭제되지 않고 종료 상태가 유지되는 경우 설치 제거 중 나머지 리소스 문제 해결 및 삭제 단계를 수행하여 네임스페이스가 종료되지 않는 오브젝트를 식별합니다.

8. 스토리지 노드의 레이블을 해제합니다.

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

9. 노드가 테인트된 경우 **OpenShift Container Storage** 테인트를 제거합니다.

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

10. **OpenShift Container Storage**를 사용하여 프로비저닝된 모든 **PV**가 삭제되었는지 확인합니다. **Released** 상태에 **PV**가 남아 있으면 삭제합니다.

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

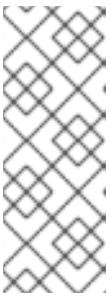
11. **Multicloud Object Gateway** 스토리지 클래스를 삭제합니다.

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

12. **CustomResourceDefinitions** 제거.

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

13. 선택 사항: 자격 증명 모음 키가 영구적으로 삭제되도록 하려면 자격 증명 모음 키와 연결된 메타데이터를 수동으로 삭제해야 합니다.



참고

Vault 키가 삭제된 것으로 표시되고 **OpenShift Container Storage** 설치 중에 영구적으로 삭제되지 않기 때문에 **Vault** 키가 **KMS(Key Management System)**를 사용한 클러스터 전체 암호화에 사용되는 경우에만 이 단계를 실행합니다. 필요한 경우 나중에 언제든지 복원할 수 있습니다.

a.

자격 증명 모음의 키를 나열합니다.

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

<backend_path>

암호화 키가 저장된 자격 증명 모음의 경로입니다.

예를 들면 다음과 같습니다.

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

출력 예:

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

b.

자격 증명 모음 키와 연결된 메타데이터를 나열합니다.

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

MCG(Multicloud Object Gateway) 키의 경우 다음을 수행합니다.

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

<key>

암호화 키입니다.

예를 들면 다음과 같습니다.

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

출력 예:

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

c.

메타데이터를 삭제합니다.

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

MCG 키의 경우:

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

<key>

암호화 키입니다.

예를 들면 다음과 같습니다.

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

출력 예:

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

d.

이러한 단계를 반복하여 모든 자격 증명 모음 키와 연결된 메타데이터를 삭제합니다.

14.

OpenShift Container Storage가 완전히 설치 제거되었는지 확인하려면 OpenShift Container Platform Web Console에서 다음을 수행합니다.

a.

Storage(스토리지)를 클릭합니다.

.

b.

Storage(스토리지)에 **Overview** (개요)가 더 이상 표시되지 않는지 확인합니다.

4.2. OPENSHIFT CONTAINER STORAGE에서 모니터링 스택 제거

이 섹션을 사용하여 **OpenShift Container Storage**에서 모니터링 스택을 정리합니다.

모니터링 스택 구성의 일부로 생성된 **PVC**는 **openshift-monitoring** 네임스페이스에 있습니다.

사전 요구 사항

- **PVC**는 **OpenShift Container Platform** 모니터링 스택을 사용하도록 구성됩니다.

  자세한 내용은 모니터링 스택 구성을 참조하십시오.

절차

1. **openshift-monitoring** 네임스페이스에서 현재 실행 중인 **Pod** 및 **PVC**를 나열합니다.

```
$ 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. **모니터링 구성 맵을 편집합니다.**

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

3. **다음 예제와 같이 OpenShift Container Storage 스토리지 클래스를 참조하는 config 섹션을 제거하여 저장합니다.**

편집하기 전에

```
.
.
.
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
.
.
.
```

편집 후

```
.
.
.
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
.
.
.
```

이 예에서는 **alertmanagerMain** 및 **prometheusK8s** 모니터링 구성 요소가 **OpenShift Container Storage PVC**를 사용하고 있습니다.

4. 관련 **PVC**를 삭제합니다. 스토리지 클래스를 사용하는 모든 **PVC**를 삭제하십시오.

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

4.3. OPENSHIFT CONTAINER STORAGE에서 OPENSHIFT CONTAINER PLATFORM 레지스트리 제거

OpenShift Container Storage에서 **OpenShift Container Platform** 레지스트리를 정리하려면 절차의 단계를 따르십시오.

대체 스토리지를 구성하려면 이미지 레지스트리를 참조하십시오.

OpenShift Container Platform 레지스트리 구성의 일부로 생성된 **PVC**는 **openshift-image-registry** 네임스페이스에 있습니다.

사전 요구 사항

- **OpenShift Container Storage PVC**를 사용하도록 이미지 레지스트리를 구성해야 합니다.

절차

1. **configs.imageregistry.operator.openshift.io** 오브젝트를 편집하고 스토리지 섹션에서 콘텐츠를 제거합니다.

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

편집하기 전에

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

편집 후

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

이 예에서는 **PVC**를 **registry-cephfs-rwx-pvc** 라고 하며 이제 안전하게 삭제할 수 있습니다.

2. **PVC**를 삭제합니다.

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

4.4. OPENSHIFT CONTAINER STORAGE에서 클러스터 로깅 OPERATOR 제거

OpenShift Container Storage에서 클러스터 로깅 **Operator**를 정리하려면 절차의 단계를 따르십시오.

클러스터 로깅 **Operator** 구성의 일부로 생성된 **PVC**는 **openshift-logging** 네임스페이스에 있습니다.

사전 요구 사항

- **OpenShift Container Storage PVC**를 사용하려면 클러스터 로깅 인스턴스를 구성해야 합니다.

절차

1. 네임스페이스에서 **ClusterLogging** 인스턴스를 제거합니다.

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

이제 **openshift-logging** 네임스페이스의 **PVC**를 삭제할 수 있습니다.

2. **PVC**를 삭제합니다.

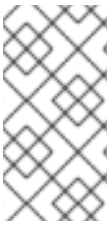
```
$ oc delete pvc <pvc-name> -n openshift-logging --wait=true --timeout=5m
```

5장. 스토리지 클래스 및 스토리지 풀

OpenShift Container Storage Operator는 사용 중인 플랫폼에 따라 기본 스토리지 클래스를 설치합니다. 이 기본 스토리지 클래스는 **Operator**가 소유하고 제어하며 삭제하거나 수정할 수 없습니다. 그러나 스토리지 클래스에 다른 동작을 갖도록 하려면 사용자 지정 스토리지 클래스를 생성할 수 있습니다.

다음 기능을 제공하는 스토리지 클래스에 매핑되는 스토리지 풀을 여러 개 생성할 수 있습니다.

- 고가용성을 가진 애플리케이션을 두 개의 복제본과 함께 영구 볼륨을 사용하여 애플리케이션 성능을 향상시킬 수 있습니다.
- 압축이 활성화된 스토리지 클래스를 사용하여 영구 볼륨 클레임의 공간을 절약합니다.



참고

외부 모드 **OpenShift Container Storage** 클러스터에서는 여러 스토리지 클래스와 여러 풀이 지원되지 않습니다.

참고

단일 장치 세트의 최소 클러스터로, 두 개의 새 스토리지 클래스만 만들 수 있습니다. 모든 스토리지 클러스터를 확장하면 새로운 스토리지 클래스 두 개를 추가로 사용할 수 있습니다.

5.1. 스토리지 클래스 및 풀 생성

기존 풀을 사용하여 스토리지 클래스를 생성하거나 스토리지 클래스를 생성하는 동안 새 풀을 생성할 수 있습니다.

사전 요구 사항

- **OpenShift Container Platform** 웹 콘솔에 로그인하고 **OpenShift Container Storage** 클러스터가 **Ready** 상태인지 확인합니다.

절차

1. 스토리지 → 스토리지 클래스를 클릭합니다.

2. 스토리지 클래스 생성을 클릭합니다.

3. 스토리지 클래스 **Name** (이름) 및 **Description** (설명)을 입력합니다.

4. 회수 정책에 대해 삭제 또는 유지를 선택합니다. 기본적으로 삭제 가 선택됩니다.

5. 영구 볼륨을 프로비저닝하는 데 사용되는 플러그인인 **RBD Provisioner**를 선택합니다.

6. 목록에서 기존 스토리지 풀을 선택하거나 새 풀을 생성합니다.

   새 풀 만들기

   a. **Create New Pool** (새 풀 만들기)을 클릭합니다.

   b. **Pool name**(풀 이름)을 입력합니다.

   c. **2-way-Replication** 또는 **Data Protection Policy**(데이터 보호 정책)로 **3-way-Replication** 을 선택합니다.

   d. 데이터를 압축해야 하는 경우 압축 활성화 를 선택합니다.

      압축을 활성화하면 애플리케이션 성능에 영향을 줄 수 있으며, 작성될 데이터가 이미 압축되거나 암호화되어 있을 때 비효율적일 수 있습니다. 압축을 활성화하기 전에 작성된 데이터는 압축되지 않습니다.

   e. **Create**(만들기)를 클릭하여 새 스토리지 풀을 만듭니다.

   f. 풀을 만든 후 **Finish** (완료)를 클릭합니다.

7. (선택 사항) **Enable Encryption** (암호화 활성화) 확인란을 선택합니다.

8. 생성을 클릭하여 스토리지 클래스를 생성합니다.

5.2. 영구 볼륨 암호화를 위한 스토리지 클래스 생성

다음 절차에 따라 영구 볼륨 암호화에 외부 키 관리 시스템(KMS)을 사용하여 암호화가 활성화된 스토리지 클래스를 생성합니다. 영구 볼륨 암호화는 **RBD PV**에서만 사용할 수 있습니다.

사전 요구 사항

- **OpenShift Container Storage** 클러스터가 **Ready** 상태입니다.
- 외부 **KMS**(키 관리 시스템)에서
  - 토큰이 있는 정책이 존재하고 **Vault**의 키 값 백엔드 경로가 활성화되어 있는지 확인합니다. **Vault**에서 키 값 및 정책 활성화 를 참조하십시오.
  - **Vault** 서버에서 서명된 인증서를 사용 중인지 확인합니다.
- 다음과 같이 테넌트의 네임스페이스에 보안을 생성합니다.
  - **OpenShift Container Platform** 웹 콘솔에서 워크로드 → 시크릿 으로 이동합니다.
  - 생성 → 키/값 시크릿을 클릭합니다.
  - **Secret Name**(시크릿 이름)을 **ceph-csi-kms-token** 으로 입력합니다.
  - 토큰 으로 **Key** 를 입력합니다.
  - 값 입력. **Vault**의 토큰입니다. **Browse** (찾아보기)를 클릭하여 토큰이 포함된 파일을 선

택하고 업로드하거나 텍스트 상자에 토큰을 직접 입력할 수 있습니다.

- 생성을 클릭합니다.

참고

토큰은 **ceph-csi-kms-token** 을 사용하는 모든 암호화된 **PVC**가 삭제된 후에만 삭제할 수 있습니다.

절차

1. 스토리지 → 스토리지 클래스로 이동합니다.
2. 스토리지 클래스 생성을 클릭합니다.
3. 스토리지 클래스 **Name** (이름) 및 **Description** (설명)을 입력합니다.
4. 회수 정책에 대해 삭제 또는 유지를 선택합니다. 기본적으로 삭제가 선택됩니다.
5. 영구 볼륨을 프로비저닝하는 데 사용되는 플러그인인 **RBD Provisioner openshift-storage.rbd.csi.ceph.com** 을 선택합니다.
6. 볼륨 데이터 가 목록에서 저장될 스토리지 풀을 선택하거나 새 풀을 생성합니다.
7. **Enable Encryption** 확인란을 선택합니다.
   a. **Key Management Service Provider** 는 기본적으로 **Vault**로 설정됩니다.
   b. **Vault Service Name, host Address of Vault server(https://<hostname** 또는 **ip>'), Port number** 를 입력합니다.

c.
고급 설정을 확장하여 **Vault** 구성에 따라 추가 설정 및 인증서 세부 정보를 입력합니다.

i.
OpenShift Container Storage 전용 이고 고유한 백엔드 경로에 키 값 시크릿 경로를 입력합니다.

ii.
(선택 사항) **TLS** 서버 이름과 **Vault Enterprise** 네임 스페이스를 입력합니다.

iii.
각 **PEM** 인코딩 인증서 파일을 업로드하여 **CA** 인증서, 클라이언트 인증서 및 클라이언트 개인 키를 제공합니다.

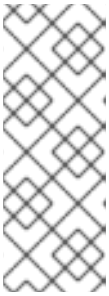
iv.
저장을 클릭합니다.

d.
연결을 클릭합니다.

8.
외부 키 관리 서비스 연결 세부 정보를 검토합니다. 정보를 수정하려면 연결 세부 정보 변경을 클릭하고 필드를 편집합니다.

9.
생성을 클릭합니다.

10.
Hashicorp Vault 설정에서 백엔드 경로에서 사용하는 **Key/Value(KV)** 시크릿 엔진 **API** 버전의 자동 탐지를 허용하지 않는 경우 **configmap**을 편집하여 **VAULT_BACKEND** 매개변수를 추가합니다.



참고

VAULT_BACKEND 는 **configmap**에 추가되어 백엔드 경로와 연결된 **KV** 시크릿 엔진 **API** 버전을 지정하는 선택적 매개변수입니다. 값이 백엔드 경로에 설정된 **KV** 시크릿 엔진 **API** 버전과 일치하는지 확인합니다. 그렇지 않으면 **PVC**(영구 볼륨 클레임) 생성 중에 실패할 수 있습니다.

a.
새로 생성된 스토리지 클래스에서 사용 중인 **encryptionKMSID** 를 식별합니다.

i. **OpenShift 웹 콘솔에서 스토리지 → 스토리지 클래스로 이동합니다.**

ii. **스토리지 클래스 이름 → YAML 탭을 클릭합니다.**

iii. **스토리지 클래스에서 사용 중인 encryptionKMSID 를 캡처합니다.**

예제:

```
encryptionKMSID: 1-vault
```

b. **OpenShift 웹 콘솔에서 워크로드 → ConfigMaps 로 이동합니다.**

c. **KMS 연결 세부 정보를 보려면 click csi-kms-connection-details 를 클릭합니다.**

d. **configmap을 편집합니다.**

i. **작업 메뉴 (Tl) → ConfigMap 편집을 클릭합니다.**

ii. **이전에 식별된 encryptionKMSID 에 대해 구성된 백엔드에 따라 VAULT_BACKEND 매개변수를 추가합니다.**

KV 시크릿 엔진 API용 kv, KV 시크릿 엔진 API 버전 1 및 kv-v2 를 VAULT_BACKEND 매개변수로 지정할 수 있습니다.

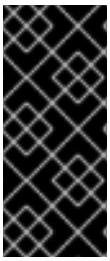
예제:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: csi-kms-connection-details
[...]
data:
  1-vault: >-

    {
      "KMS_PROVIDER": "vaulttokens",
```

```
  "KMS_SERVICE_NAME": "vault",
  [...]
  "VAULT_BACKEND": "kv-v2"
}
```

iii. 저장을 클릭합니다.



중요

Red Hat은 기술 파트너와 협력하여 이 문서를 고객에게 서비스로 제공하고 있습니다. 하지만 **Red Hat**은 **H¢corp** 제품에 대한 지원을 제공하지 않습니다. 이 제품에 대한 기술 지원이 필요한 경우 **Hsea corp**에 문의하십시오.

다음 단계

- 스토리지 클래스를 사용하여 암호화된 영구 볼륨을 만들 수 있습니다. 자세한 내용은 영구 볼륨 클레임 관리를 참조하십시오.

6장. OPENSHIFT CONTAINER PLATFORM 서비스의 스토리지 구성

OpenShift Container Storage를 사용하여 이미지 레지스트리, 모니터링 및 로깅과 같은 **OpenShift Container Platform** 서비스에 스토리지를 제공할 수 있습니다.

이러한 서비스에 대한 스토리지를 구성하는 프로세스는 **OpenShift Container Storage** 배포에 사용되는 인프라에 따라 다릅니다.

주의

이러한 서비스를 위한 다양한 스토리지 용량이 항상 있는지 확인합니다. 이러한 중요한 서비스의 스토리지가 공간이 부족하면 클러스터가 작동할 수 없게 되고 복구하기가 매우 어려워집니다.

이러한 서비스를 위해 더 짧은 큐레이션 및 보존 간격을 구성하는 것이 좋습니다. 자세한 내용은 **OpenShift Container Platform** 설명서의 영구 스토리지 구성의 **Curator** 일정 구성 및 ***Prometheus*** *지표 데이터 하위 섹션의 보존 시간* 수정을 참조하십시오.

이러한 서비스의 스토리지 공간이 부족할 경우 **Red Hat** 고객 지원에 문의하십시오.

6.1. OPENSHIFT CONTAINER STORAGE를 사용하도록 이미지 레지스트리 구성

OpenShift Container Platform은 클러스터에서 표준 워크로드로 실행되는 컨테이너 이미지 레지스트리에 빌드된 기능을 제공합니다. 일반적으로 레지스트리는 클러스터에 빌드된 이미지의 게시 대상과 클러스터에서 실행되는 워크로드의 이미지 소스로 사용됩니다.

이 섹션의 지침에 따라 **OpenShift Container Storage**를 컨테이너 이미지 레지스트리에 대한 스토리지로 구성합니다. **Google Cloud**에서는 레지스트리의 스토리지를 변경할 필요가 없습니다.

주의

이 프로세스는 기존 이미지 레지스트리에서 새 이미지 레지스트리로 데이터를 마이그레이션하지 않습니다. 기존 레지스트리에 컨테이너 이미지가 이미 있는 경우 이 프로세스를 완료하기 전에 레지스트리를 백업하고 이 프로세스가 완료되면 이미지를 다시 등록합니다.

사전 요구 사항

- **OpenShift** 웹 콘솔에 대한 관리자 액세스 권한이 있습니다.
- **OpenShift Container Storage Operator**는 **openshift-storage** 네임스페이스에 설치되고 실행됩니다. **OpenShift** 웹 콘솔에서 **Operator** → 설치된 **Operator** 를 클릭하여 설치된 **Operator** 를 확인합니다.
- 이미지 레지스트리 **Operator**가 **openshift-image-registry** 네임스페이스에 설치되고 실행됩니다. **OpenShift** 웹 콘솔에서 관리 → 클러스터 설정 → 클러스터 **Operator** 를 클릭하여 클러스터 운영자를 확인합니다.
- 프로비저너 **openshift-storage.cephfs.csi.ceph.com** 이 있는 스토리지 클래스를 사용할 수 있습니다. **OpenShift** 웹 콘솔에서 스토리지 → 스토리지 클래스를 클릭하여 사용 가능한 스토리지 클래스를 확인합니다.

절차

1. 사용할 이미지 레지스트리에 대한 영구 볼륨 클레임을 생성합니다.
   a. **OpenShift** 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭합니다.
   b. 프로젝트를 **openshift-image-registry** 로 설정합니다.
   c. 영구 볼륨 클레임 생성을 클릭합니다.
      i.

위에서 검색한 스토리지 클래스 목록에서 **provisioner openshift-storage.cephfs.csi.ceph.com** 을 사용하여 스토리지 클래스를 지정합니다.

ii. 영구 볼륨 클레임 이름을 지정합니다(예: **ocs4registry**).

iii. **RWX (**공유 액세스 모드 **)**를 지정합니다.

iv. 최소 **100GB**의 크기를 지정합니다.

v. 생성을 클릭합니다.

새 영구 볼륨 클레임의 상태가 **Bound** 로 표시될 때까지 기다립니다.

2. 새 영구 볼륨 클레임을 사용하도록 클러스터의 이미지 레지스트리를 구성합니다.

a. **Administration** → **Custom Resource Definitions** 를 클릭합니다.

b. **the imageregistry.operator.openshift.io** 그룹과 연결된 **Config** 사용자 정의 리소스 정의를 클릭합니다.

c. **Instances(**인스턴스**)** 탭을 클릭합니다.

d. 클러스터 인스턴스 옆의 작업 메뉴**(PS)** → 구성 편집을 클릭합니다.

e. 새 영구 볼륨 클레임을 이미지 레지스트리의 영구 스토리지로 추가합니다.

i. 필요한 경우 기존 **storage :** 섹션을 교체하는 **spec:** 에서 다음 내용을 추가합니다.

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

예를 들면 다음과 같습니다.

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

ii. 저장을 클릭합니다.

3. 새 구성이 사용 중인지 확인합니다.

a. 워크로드 → 포드를 클릭합니다.

b. 프로젝트를 **openshift-image-registry** 로 설정합니다.

c. 새 **image-registry-*** 포드가 **Running** 상태로 표시되고 이전 **image-registry-*** 포드가 종료되는지 확인합니다.

d. 새 **image-registry-*** 포드를 클릭하여 포드 세부 정보를 확인합니다.

e. **Volumes(**볼륨**)** 로 아래로 스크롤하고 **registry-storage** 볼륨에 새 영구 볼륨 클레임과 일치하는 유형이 있는지 확인합니다**(**예**: ocs4registry).**

6.2. OPENSHIFT CONTAINER STORAGE를 사용하도록 모니터링 구성

OpenShift Container Storage는 **Prometheus** 및 **Alert Manager**로 구성된 모니터링 스택을 제공합니다.

이 섹션의 지침에 따라 **OpenShift Container Storage**를 모니터링 스택의 스토리지로 구성합니다.

중요

스토리지 공간이 부족하면 모니터링이 작동하지 않습니다. 항상 모니터링을 위한 많은 스토리지 용량이 있는지 확인합니다.

Red Hat은 이 서비스의 짧은 보존 간격을 구성하는 것이 좋습니다. 자세한 내용은 OpenShift Container Platform 설명서의 모니터링 데이터 모니터링 가이드에 대한 보존 시간 수정을 참조하십시오.

사전 요구 사항

- OpenShift 웹 콘솔에 대한 관리자 액세스 권한이 있습니다.
- OpenShift Container Storage Operator는 openshift-storage 네임스페이스에 설치되고 실행됩니다. OpenShift 웹 콘솔에서 Operator → 설치된 Operator 를 클릭하여 설치된 Operator 를 확인합니다.
- openshift-monitoring 네임스페이스에 Operator가 설치되고 실행됩니다. OpenShift 웹 콘솔에서 관리 → 클러스터 설정 → 클러스터 Operator 를 클릭하여 클러스터 운영자를 확인합니다.
- 프로비저너 openshift-storage.rbd.csi.ceph.com 이 있는 스토리지 클래스를 사용할 수 있습니다. OpenShift 웹 콘솔에서 스토리지 → 스토리지 클래스를 클릭하여 사용 가능한 스토리지 클래스를 확인합니다.

절차

1. OpenShift 웹 콘솔에서 워크로드 → 구성 맵 으로 이동합니다.
2. 프로젝트 드롭다운을 openshift-monitoring 으로 설정합니다.
3. 구성 맵 생성을 클릭합니다.
4. 다음 예제를 사용하여 새 cluster-monitoring-config 구성 맵을 정의합니다.

괄호 안의 컨텐츠(**<, >**)를 자체 값으로 바꿉니다(예: 보존): **24H** 또는 스토리지: **40Gi.**

배포 프로그램 **openshift-storage.rbd.csi.ceph.com**을 사용하는 스토리지 클래스로 **storageClassName** 을 교체합니다. **storageclass** 의 이름 아래에 지정된 예에서는 **ocs-storagecluster-ceph-rbd** 입니다.

cluster-monitoring-config 구성 맵 예

```
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. 생성을 클릭하여 구성 맵을 저장하고 생성합니다.

검증 단계

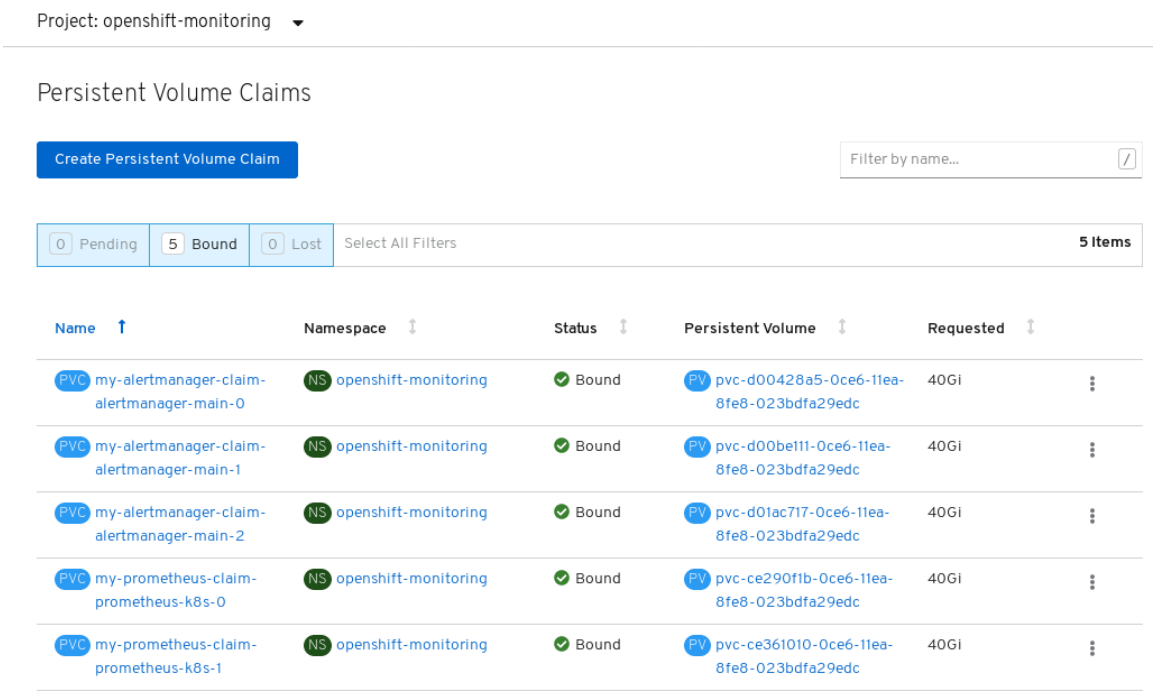
1. 영구 볼륨 클레임이 포드에 바인드되었는지 확인합니다.

a. 스토리지 → 영구 볼륨 클레임 으로 이동합니다.

b. 프로젝트 드롭다운을 **openshift-monitoring** 으로 설정합니다.

c. **5**개의 영구 볼륨 클레임이 **Bound** 상태로 표시되는지, **3**개의 **alertmanager-main-*** 포드 및 **2**개의 **prometheus- k8s-*** 포드에 연결되어 있는지 확인합니다.

생성 및 바인딩된 스토리지 모니터링



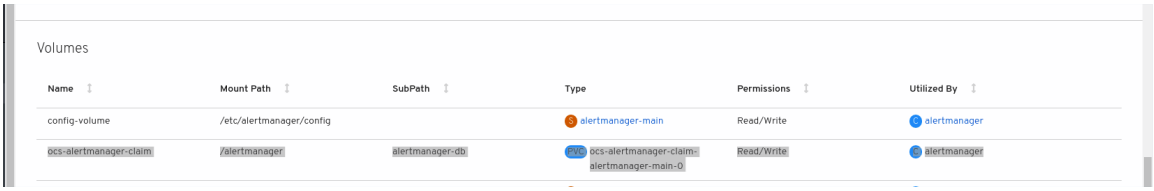
2. 새 **alertmanager-main-*** 포드가 **Running** 상태로 표시되는지 확인합니다.

a. 워크로드 → **Pod** 로 이동합니다.

b. 새 **alertmanager-main-*** 포드를 클릭하여 포드 세부 정보를 확인합니다.

c. **Volumes**(볼륨) 까지 아래로 스크롤하고 볼륨에 새 영구 볼륨 클레임(예:**ocs-alertmanager- main-0**) 중 하나와 일치하는 유형 인 **ocs-alertmanager-claim**이 있는지 확인합니다.

alertmanager-main-* Pod에 연결된 영구 볼륨 클레임

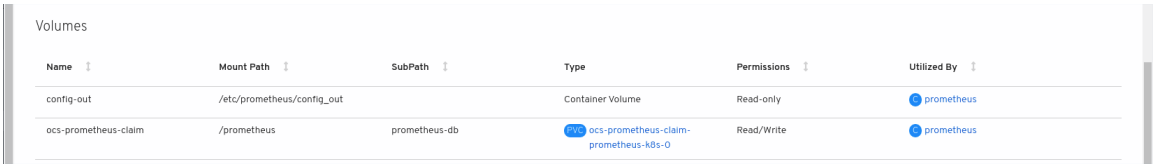


3. **새 prometheus-k8s-* 포드가 Running 상태로 표시되는지 확인합니다.**

   a. **새 prometheus-k8s-* 포드를 클릭하여 포드 세부 정보를 확인합니다.**

   b. **Volumes(볼륨) 까지 아래로 스크롤하고 볼륨에 새 영구 볼륨 클레임(예:ocs-prometheus-claim-prometheus- k8s-0) 중 하나와 일치하는 유형, ocs-prometheus-k8s-0 이 있는지 확인합니다.**

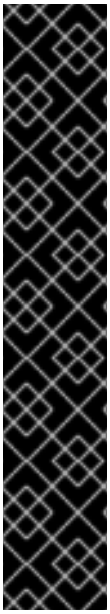
prometheus-k8s-* Pod에 연결된 영구 볼륨 클레임



6.3. OPENSHIFT CONTAINER STORAGE를 위한 클러스터 로깅

클러스터 로깅을 배포하여 다양한 **OpenShift Container Platform** 서비스의 로그를 집계할 수 있습니다. 클러스터 로깅을 배포하는 방법에 대한 자세한 내용은 클러스터 로깅 배포를 참조하십시오.

초기 **OpenShift Container Platform** 배포 시 **OpenShift Container Storage**는 기본적으로 구성되지 않으며 **OpenShift Container Platform** 클러스터는 노드에서 사용할 수 있는 기본 스토리지를 전적으로 사용합니다. **OpenShift Container Storage**에서 지원하는 **OpenShift Container Storage** 백업 로깅 **(Elasticsearch)**을 지원하도록 **OpenShift** 로깅**(ElasticSearch)**의 기본 구성을 편집할 수 있습니다.



중요

이러한 서비스를 위한 다양한 스토리지 용량이 항상 있는지 확인합니다. 이러한 중요한 서비스의 스토리지 공간이 부족하면 로깅 애플리케이션이 작동할 수 없게 되며 복구하기가 매우 어려워집니다.

이러한 서비스를 위해 더 짧은 큐레이션 및 보존 간격을 구성하는 것이 좋습니다. 자세한 내용은 **OpenShift Container Platform** 설명서의 클러스터 로깅 큐레이터 를 참조하십시오.

이러한 서비스의 스토리지 공간이 부족할 경우 **Red Hat** 고객 지원에 문의하십시오.

6.3.1. 영구 스토리지 구성

스토리지 클래스 이름 및 크기 매개변수를 사용하여 **Elasticsearch** 클러스터의 영구 스토리지 클래스 및 크기를 구성할 수 있습니다. **Cluster Logging Operator**는 이러한 매개변수를 기반으로 **Elasticsearch** 클러스터의 각 데이터 노드에 대한 영구 볼륨 클레임을 생성합니다. 예를 들면 다음과 같습니다.

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

이 예제에서는 클러스터의 각 데이터 노드가 **200GiB** 의 **ocs-storagecluster-ceph-rbd** 스토리지를 요청하는 영구 볼륨 클레임에 바인딩되도록 지정합니다. 각 기본 분할은 단일 복제본에서 지원합니다. **shard** 복사본은 모든 노드에 복제되며 항상 사용 가능하며 단일 중복 정책으로 인해 두 개 이상의 노드가 있는 경우 복사본을 복구할 수 있습니다. **Elasticsearch** 복제 정책에 대한 자세한 내용은 클러스터 로깅 배포 및 구성 정보 에서 ***Elasticsearch*** *복제 정책*을 참조하십시오.

참고

storage 블록을 생략하면 기본 스토리지에서 지원하는 배포가 생성됩니다. 예를 들면 다음과 같습니다.

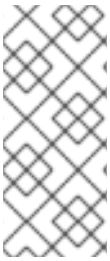
```
spec:
   logStore:
     type: "elasticsearch"
```

```
    elasticsearch:
      nodeCount: 3
      storage: {}
```

자세한 내용은 클러스터 로깅 구성을 참조하십시오.

6.3.2. OpenShift Container Storage를 사용하도록 클러스터 로깅 구성

이 섹션의 지침에 따라 OpenShift Container Storage를 OpenShift 클러스터 로깅 스토리지로 구성합니다.



참고

OpenShift Container Storage에서 로깅을 처음 구성할 때 모든 로그를 가져올 수 있습니다. 그러나 로깅을 제거하고 다시 설치한 후에는 이전 로그가 제거되고 새 로그만 처리됩니다.

사전 요구 사항

- OpenShift 웹 콘솔에 대한 관리자 액세스 권한이 있습니다.
- OpenShift Container Storage Operator는 openshift-storage 네임스페이스에 설치되고 실행됩니다.
- openshift-logging 네임스페이스에 클러스터 로깅 Operator가 설치되고 실행됩니다.

절차

1. OpenShift 웹 콘솔의 왼쪽 창에서 Administration → Custom Resource Definitions 를 클릭합니다.
2. 사용자 정의 리소스 정의 페이지에서 ClusterLogging 을 클릭합니다.
3. 사용자 정의 리소스 정의 개요 페이지의 작업 메뉴에서 인스턴스 보기를 선택하거나 Instances 탭을 클릭합니다.

4. 클러스터 로깅 페이지에서 클러스터 로깅 생성을 클릭합니다.

데이터를 로드하기 위해 페이지를 새로 고쳐야 할 수도 있습니다.

5. YAML에서 the storageClassName 을 프로비저너 openshift-storage.rbd.csi.ceph.com을 사용하는 스토리지 클래스로 바꿉니다. storageclass 의 이름 아래에 지정된 예에서는 ocs-storagecluster-ceph-rbd 입니다.

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

OpenShift Container Storage 노드를 테인트한 경우 로깅을 위해 daemonset Pod를 예약할 수 있도록 허용 오차를 추가해야 합니다.

```
spec:
[...]
  collection:
    logs:
      fluentd:
        tolerations:
        - effect: NoSchedule
          key: node.ocs.openshift.io/storage
          value: 'true'
      type: fluentd
```

6. 저장을 클릭합니다.

검증 단계

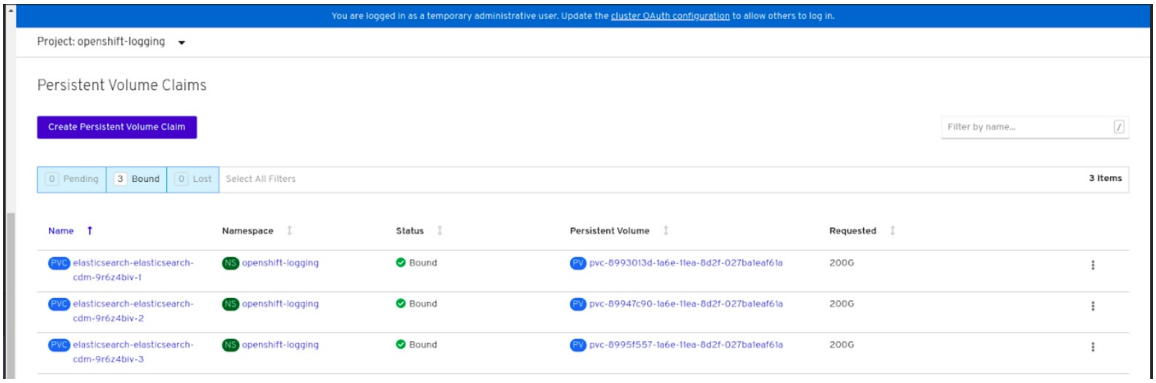
1. 영구 볼륨 클레임이 **elasticsearch** 포드에 바인딩되었는지 확인합니다.

a. 스토리지 → 영구 볼륨 클레임 으로 이동합니다.

b. 프로젝트 드롭다운을 **openshift-logging** 으로 설정합니다.

c. 영구 볼륨 클레임이 **elasticsearch-*** 포드에 연결된 **Bound** 상태로 표시되는지 확인합니다.

그림 **6.1.** 클러스터 로깅 생성 및 바인딩



2. 새 클러스터 로깅이 사용 중인지 확인합니다.

a. 워크로드 → 포드를 클릭합니다.

b. 프로젝트를 **openshift-logging** 으로 설정합니다.

c. 새로운 **elasticsearch-*** 포드가 **Running** 상태로 표시되는지 확인합니다.

d. 새 **elasticsearch-*** **pod**를 클릭하여 포드 세부 정보를 확인합니다.

e.

Volumes(볼륨) 까지 아래로 스크롤하고 **elasticsearch** 볼륨에 새 영구 볼륨 클레임(예: **elasticsearch-elasticsearch-cdm-9r624biv-3)** 과 일치하는 유형이 있는지 확인합니다.

f.

영구 볼륨 클레임 이름을 클릭하고 **PersistentVolumeClaim Overview(개요)** 페이지에서 스토리지 클래스 이름을 확인합니다.

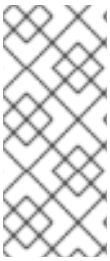
참고

Elasticsearch Pod에 연결된 **PV**에서 **PV** 전체 시나리오를 방지하려면 더 짧은 큐레이터 시간을 사용해야 합니다.

보존 설정에 따라 **Elasticsearch** 데이터를 삭제하도록 **Curator**를 구성할 수 있습니다. 다음 기본 인덱스 데이터 보존을 기본값으로 **5**일로 설정하는 것이 좋습니다.

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

자세한 내용은 **Elasticsearch 데이터 Curation**을 참조하십시오.



참고

영구 볼륨 클레임에서 지원하는 클러스터 로깅을 설치 제거하려면 해당 배포 가이드의 설치 제거 장에서 **OpenShift Container Storage**에서 클러스터 로깅 운영자를 제거하는 절차를 사용하십시오.

7장. OPENSHIFT CONTAINER STORAGE를 사용하여 OPENSHIFT CONTAINER PLATFORM 애플리케이션 백업

OpenShift Container Platform을 설치하는 동안 OpenShift Container Storage를 직접 설치할 수 없습니다. 그러나 Operator Hub를 사용하여 기존 OpenShift Container Platform에 OpenShift Container Storage를 설치한 다음 OpenShift Container Platform 애플리케이션을 OpenShift Container Storage에서 지원하도록 구성할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform이 설치되어 있으며 OpenShift 웹 콘솔에 대한 관리 액세스 권한이 있습니다.
- OpenShift Container Storage는 openshift-storage 네임스페이스에 설치 및 실행됩니다.

절차

1. OpenShift 웹 콘솔에서 다음 중 하나를 수행합니다.
   - 워크로드 → 배포를 클릭합니다.

     Deployments(배포) 페이지에서 다음 중 하나를 수행할 수 있습니다.
     - 기존 배포를 선택하고 Action (작업) 메뉴에서 Add Storage (스토리지 추가) 옵션을 클릭합니다.
     - 새 배포를 만든 다음 스토리지를 추가합니다.
       i. Create Deployment (배포 만들기)를 클릭하여 새 배포를 만듭니다.
       ii. 요구 사항에 따라 YAML 을 편집하여 배포를 생성합니다.
       iii. 생성을 클릭합니다.

iv. 페이지 오른쪽 상단에 있는 **Actions**(작업) 드롭다운 메뉴에서 **Add Storage** (스토리지 추가)를 선택합니다.

- 워크로드 → 배포 구성을 클릭합니다.

  배포 구성 페이지에서 다음 중 하나를 수행할 수 있습니다.

  - 기존 배포를 선택하고 **Action** (작업) 메뉴에서 **Add Storage** (스토리지 추가) 옵션을 클릭합니다.
  - 새 배포를 만든 다음 스토리지를 추가합니다.

    i. **Create Deployment Config** (배포 구성 만들기)를 클릭하여 새 배포를 만듭니다.

    ii. 요구 사항에 따라 **YAML** 을 편집하여 배포를 생성합니다.

    iii. 생성을 클릭합니다.

    iv. 페이지 오른쪽 상단에 있는 **Actions**(작업) 드롭다운 메뉴에서 **Add Storage** (스토리지 추가)를 선택합니다.

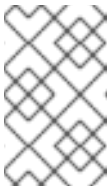
2. 스토리지 추가 페이지에서 다음 옵션 중 하나를 선택할 수 있습니다.

   - **Use existing claim** (기존 클레임 사용) 옵션을 클릭하고 드롭다운 목록에서 적절한 **PVC**를 선택합니다.
   - **Create new claim** (새 클레임 만들기) 옵션을 클릭합니다.

     a. **Storage Class** (스토리지 클래스) 드롭다운 목록에서 적절한 **CephFS** 또는 **RBD** 스토리지 클래스를 선택합니다.

b.
영구 볼륨 클레임의 이름을 제공합니다.

c.
ReadWriteOnce (RWO) 또는 RWX(ReadWriteMany) 액세스 모드를 선택합니다.



참고

RROX(ReadOnlyMany)는 지원되지 않으므로 비활성화됩니다.

d.
원하는 스토리지 용량의 크기를 선택합니다.

참고

블록 PV를 확장할 수 있지만 영구 볼륨 클레임을 만든 후에는 스토리지 용량을 줄일 수 없습니다.

3.
컨테이너 내부의 마운트 경로 볼륨에 마운트 경로 및 하위 경로(필요한 경우)를 지정합니다.

4.
저장을 클릭합니다.

검증 단계

1.
구성에 따라 다음 중 하나를 수행합니다.

- 워크로드 → 배포를 클릭합니다.
- 워크로드 → 배포 구성을 클릭합니다.

2.
필요에 따라 프로젝트를 설정합니다.

3.
스토리지를 추가한 배포를 클릭하여 배포 세부 정보를 표시합니다.

4. **Volumes(볼륨)** 로 아래로 스크롤하고 배포에 사용자가 할당한 영구 볼륨 클레임과 일치하는 **Type** (유형)이 있는지 확인합니다.

5. 영구 볼륨 클레임 이름을 클릭하고 영구 볼륨 클레임 개요 페이지에서 스토리지 클래스 이름을 확인합니다.

8장. RED HAT OPENSHIFT CONTAINER STORAGE 전용 작업자 노드를 사용하는 방법

인프라 노드를 사용하여 Red Hat OpenShift Container Storage 리소스를 예약하면 Red Hat OpenShift Container Platform 서브스크립션 비용이 절감됩니다. infra node-role 레이블이 있는 RHOCP(Red Hat OpenShift Container Platform) 노드에는 OpenShift Container Storage 서브스크립션이 필요하지만 RHOCP 서브스크립션은 필요하지 않습니다.

시스템 API 지원 여부에 관계없이 환경 전반에서 일관성을 유지하는 것이 중요합니다. 이 때문에 모든 경우에 worker 또는 infra로 레이블이 지정된 특수 노드 범주가 있거나 두 역할이 모두 있는 것이 좋습니다. 자세한 내용은 8.3절. “인프라 노드 수동 생성” 섹션을 참조하십시오.

8.1. 인프라 노드 분석

OpenShift Container Storage와 함께 사용할 인프라 노드에는 다음과 같은 특성이 있습니다. 노드에서 RHOCP 인타이틀먼트를 사용하지 않는지 확인하려면 infra node-role 레이블이 필요합니다. infra node-role 레이블은 OpenShift Container Storage를 실행하는 노드에 OpenShift Container Storage 인타이틀먼트만 필요한지 확인해야 합니다.

- node-role.kubernetes.io/infra로 레이블이 지정됩니다.

인프라 노드에서 OpenShift Container Storage 리소스만 예약하려면 a NoSchedule effect 를 사용하여 OpenShift Container Storage 테인트를 추가해야 합니다.

- node.ocs.openshift.io/storage="true"로 테인트되었습니다.

레이블은 RHOCP 노드를 인프라 노드로 식별하여 RHOCP 서브스크립션 비용을 적용하지 않도록 합니다. 테인트는 테인트된 노드에 비 OpenShift Container Storage 리소스를 예약할 수 없습니다.

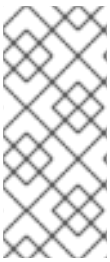
OpenShift Container Storage 서비스를 실행하는 데 사용할 인프라 노드에 필요한 테인트 및 라벨의 예는 다음과 같습니다.

```
spec:
  taints:
  - effect: NoSchedule
    key: node.ocs.openshift.io/storage
    value: "true"
  metadata:
    creationTimestamp: null
    labels:
```

```
        node-role.kubernetes.io/worker: ""
        node-role.kubernetes.io/infra: ""
        cluster.ocs.openshift.io/openshift-storage: ""
```

8.2. 인프라 노드를 생성하기 위한 머신 세트

환경에서 **Machine API**가 지원되는 경우 인프라 노드를 프로비저닝할 머신 세트의 템플릿에 레이블을 추가해야 합니다. 시스템 **API**에서 생성한 노드에 라벨을 수동으로 추가하는 패턴 방지. 이렇게 하면 배포로 생성된 포드에 레이블을 추가하는 것과 유사합니다. 두 경우 모두 **Pod/node**에 오류가 발생하면 대체 **Pod**/노드에 적절한 레이블이 없습니다.



참고

EC2 환경에서는 각각 별도의 가용성 영역에서 인프라 노드를 프로비저닝하도록 구성된 시스템 집합(예: **us-east-2a, us-east-2b, us-east-2c**)이 필요합니다. 현재 **OpenShift Container Storage**는 세 개 이상의 가용성 영역에서 배포를 지원하지 않습니다.

다음 머신 세트 템플릿 예제에서는 인프라 노드에 필요한 적절한 테인트 및 라벨을 사용하여 노드를 생성합니다. **OpenShift Container Storage** 서비스를 실행하는 데 사용됩니다.

```
template:
  metadata:
    creationTimestamp: null
    labels:
      machine.openshift.io/cluster-api-cluster: kb-s25vf
      machine.openshift.io/cluster-api-machine-role: worker
      machine.openshift.io/cluster-api-machine-type: worker
      machine.openshift.io/cluster-api-machineset: kb-s25vf-infra-us-west-2a
  spec:
    taints:
    - effect: NoSchedule
      key: node.ocs.openshift.io/storage
      value: "true"
    metadata:
      creationTimestamp: null
      labels:
        node-role.kubernetes.io/infra: ""
        cluster.ocs.openshift.io/openshift-storage: ""
```

8.3. 인프라 노드 수동 생성

환경에서 **Machine API**가 지원되지 않는 경우에만 라벨을 노드에 직접 적용해야 합니다. 수동 생성에는 **OpenShift Container Storage** 서비스를 예약하는 데 **3**개 이상의 **RHOCP** 작업자 노드를 사용할 수 있어야 하며 이러한 노드에 충분한 **CPU** 및 메모리 리소스가 있어야 합니다. **RHOCP** 서브스크립션 비용을 방지하려면 다음이 필요합니다.

```
oc label node <node> node-role.kubernetes.io/infra=""
oc label node <node> cluster.ocs.openshift.io/openshift-storage=""
```

인프라 노드에서만 OpenShift Container Storage 리소스를 예약하고 OpenShift Container Storage 이외의 다른 워크로드를 거절하도록 a NoSchedule OpenShift Container Storage 테인트를 추가해야 합니다.

```
oc adm taint node <node> node.ocs.openshift.io/storage="true":NoSchedule
```

주의

node-role node-role.kubernetes.io/worker=""를 제거하지 마십시오.

node-role.kubernetes.io/worker="" 를 제거하면 OpenShift 스케줄러와 MachineConfig 리소스 둘 다 변경하지 않는 한 문제가 발생할 수 있습니다.

이미 제거된 경우 각 인프라 노드에 다시 추가해야 합니다. node-role node-role.kubernetes.io/infra="" 및 OpenShift Container Storage 테인트를 추가하면 인타이틀먼트 제외 요구 사항을 준수할 수 있습니다.

9장. 스토리지 노드 확장

OpenShift Container Storage의 스토리지 용량을 확장하려면 다음 중 하나를 수행할 수 있습니다.

- 스토리지 노드 확장 - 기존 **OpenShift Container Storage** 작업자 노드에 스토리지 용량 추가
- 스토리지 노드 확장 - 스토리지 용량이 포함된 새 작업자 노드 추가

9.1. 스토리지 노드 확장 요구사항

스토리지 노드를 확장하기 전에 다음 섹션을 참조하여 특정 **Red Hat OpenShift Container Storage** 인스턴스의 노드 요구 사항을 파악하십시오.

- 플랫폼 요구 사항
- 스토리지 장치 요구 사항
  - 동적 스토리지 장치
  - 용량 계획

주의

항상 다양한 스토리지 용량이 있는지 확인합니다.

스토리지가 완전히 가득 차면 스토리지에서 용량을 추가하거나 삭제하거나 콘텐츠를 마이그레이션하여 공간을 확보할 수 없습니다. 완전한 스토리지는 복구하기가 매우 어렵습니다.

클러스터 스토리지 용량이 **75%(near-full)** 및 **85%(전체)**에 도달하면 용량 경고가 발생합니다. 항상 용량 경고를 즉시 처리하고 정기적으로 스토리지를 검토하여 스토리지 공간이 부족하지 않도록 합니다.

스토리지 공간이 완전히 부족할 경우 **Red Hat** 고객 지원에 문의하십시오.

9.2. GOOGLE CLOUD 인프라의 OPENSHIFT CONTAINER STORAGE 노드에 용량을 추가하여 스토리지 확장

구성된 **Red Hat OpenShift Container Storage** 작업자 노드에 스토리지 용량 및 성능을 추가하려면 다음 절차를 사용하십시오.

사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform.**
- **OpenShift** 웹 콘솔의 관리 권한.
- 배포 중에 프로비저닝된 스토리지 클래스 이외의 스토리지 클래스를 사용하여 확장하려면 먼저 추가 스토리지 클래스를 정의합니다. 자세한 내용은 스토리지 클래스 생성을 참조하십시오.

절차

1. **OpenShift** 웹 콘솔에 로그인합니다.

2. **Operators → 설치된 Operator**를 클릭합니다.

3. **OpenShift Container Storage Operator**를 클릭합니다.

4. **Storage Cluster(**스토리지 클러스터 **)** 탭을 클릭합니다.

5. 표시된 목록에는 항목이 하나만 있어야 합니다. 맨 오른쪽에 있는 **(^^)**를 클릭하여 옵션 메뉴를 확장합니다.

6. **options(**옵션**)** 메뉴에서 **Add Capacity (**용량 추가**)**를 선택합니다.

7. 스토리지 클래스를 선택합니다.

   HDD를 사용하는 기본 스토리지 클래스를 사용하는 경우 스토리지 클래스를 **standard**로 설정합니다. 그러나 더 나은 성능을 위해 **SSD** 기반 디스크를 사용하기 위해 스토리지 클래스를 생성한 경우 해당 스토리지 클래스를 선택해야 합니다.

   Raw Capacity(고유 용량 **)** 필드에는 스토리지 클래스 생성 중에 설정된 크기가 표시됩니다. **OpenShift Container Storage**는 복제본 수를 **3**개 사용하므로 소비되는 총 스토리지 양은 이 양의 **3**배입니다.

8. **Add(**추가 **)**를 클릭하고 클러스터 상태가 **Ready** 로 변경될 때까지 기다립니다.

검증 단계

- **Overview → Block and File(**개요 → 블록 및 파일 **)** 탭으로 이동한 다음 **Raw Capacity breakdown** 카드를 확인합니다.

  선택한 항목에 따라 용량이 증가합니다.

참고

원시 용량은 복제를 고려하지 않고 전체 용량을 보여줍니다.

- 새 OSD와 해당 새 PVC가 생성되었는지 확인합니다.
  - 새로 생성된 OSD의 상태를 보려면 다음을 수행합니다.
    a. OpenShift 웹 콘솔에서 워크로드 → 포드를 클릭합니다.
    b. 프로젝트 드롭다운 목록에서 openshift-storage 를 선택합니다.
  - PVC 상태를 보려면 다음을 수행합니다.
    a. OpenShift 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭합니다.
    b. 프로젝트 드롭다운 목록에서 openshift-storage 를 선택합니다.
- (선택 사항) 클러스터에서 클러스터 전체 암호화가 활성화된 경우 새 OSD 장치가 암호화되었는지 확인합니다.

a. 새 OSD 포드가 실행 중인 노드를 식별합니다.

```
$ oc get -o=custom-columns=NODE:.spec.nodeName pod/<OSD pod name>
```

예를 들면 다음과 같습니다.

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

b. 이전 단계에서 식별된 각 노드에 대해 다음을 수행합니다.

i. 디버그 포드를 만들고 선택한 호스트에 대해 chroot 환경을 엽니다.

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

ii. **"lsblk"**를 실행하고 **ocs-deviceset** 이름 옆의 **"crypt"** 키워드를 확인하십시오.

```
$ lsblk
```

중요

노드 또는 **OSD**를 제거하여 축소할지 여부에 관계없이 클러스터 감소는 현재 지원되지 않습니다.

9.3. 새 노드를 추가하여 스토리지 용량 확장

스토리지 용량을 확장하려면 다음을 수행해야 합니다.

- 기존 작업자 노드가 지원되는 최대 **OSD**에서 이미 실행되고 있는 경우 스토리지 용량을 늘리려면 새 노드를 추가합니다. 이는 초기 구성 중에 선택한 용량의 **OSD 3**개를 늘립니다.
- 새 노드가 성공적으로 추가되었는지 확인합니다.
- 노드를 추가한 후 스토리지 용량 확장

9.3.1. Google Cloud 설치 관리자 프로비저닝 인프라에 노드 추가

사전 요구 사항

- **RHOCP(OpenShift Container Platform)** 클러스터에 로그인해야 합니다.

절차

1. 컴퓨팅 → 머신 세트로 이동합니다.
2. 노드를 추가할 머신 세트에서 시스템 개수 편집을 선택합니다.
3. 노드 크기를 추가하고 **Save(**저장 **)**를 클릭합니다.

4. 컴퓨팅 → 노드를 클릭하고 새 노드가 **Ready** 상태인지 확인합니다.

5. **OpenShift Container Storage** 레이블을 새 노드에 적용합니다.

   a. 새 노드의 경우 작업 메뉴 **(PS)** → 레이블 편집.

   b. **cluster.ocs.openshift.io/openshift-storage** 를 추가하고 **Save(**저장 **)**를 클릭합니다.

참고

각각 다른 영역에 하나씩 **3** 개의 노드를 추가하는 것이 좋습니다. 노드 **3**개를 추가하고 모든 노드에 대해 이 절차를 수행해야 합니다.

검증 단계

- 새 노드가 추가 되었는지 확인하려면 새 노드 추가 확인을 참조하십시오.

9.3.2. 새 노드 추가 확인

1. 다음 명령을 실행하고 새 노드가 출력에 있는지 확인합니다.

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

2. 워크로드 → **Pod** 를 클릭하여 새 노드의 다음 **Pod**가 **Running** 상태인지 확인합니다.

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

9.3.3. 스토리지 용량 확장

OpenShift Container Storage에 새 노드를 추가한 후 용량을 추가하여 스토리지 확장에 설명된 대로 스토리지 용량을 확장해야 합니다.

10장. MULTICLOUD OBJECT GATEWAY

10.1. MULTICLOUD OBJECT GATEWAY 정보

MCG(Multicloud Object Gateway)는 **OpenShift**용 경량 오브젝트 스토리지 서비스로, 사용자가 소규모를 시작한 다음 필요에 따라 온프레미스, 여러 클러스터에서 클라우드 네이티브 스토리지를 사용하여 확장할 수 있습니다.

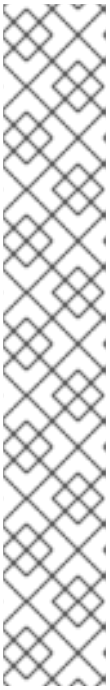
10.2. 애플리케이션을 사용하여 MULTICLOUD OBJECT GATEWAY에 액세스

AWS S3을 대상으로 하는 모든 애플리케이션 또는 **AWS S3 Software Development Kit(SDK)**를 사용하는 코드를 사용하여 오브젝트 서비스에 액세스할 수 있습니다. 애플리케이션은 **MCG** 끝점, 액세스 키 및 시크릿 액세스 키를 지정해야 합니다. 터미널 또는 **MCG CLI**를 사용하여 이 정보를 검색할 수 있습니다.

사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform**
- 더 쉽게 관리 할 수 있도록 **MCG** 명령줄 인터페이스를 다운로드하십시오.

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



참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들면 다음과 같습니다.

- **IBM Power Systems**의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
```

- **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

- 또는 Download RedHat OpenShift Container Storage(RedHat OpenShift Container Storage 다운로드) 페이지에 있는 OpenShift Container Storage RPM에서 mcg 패키지를 설치할 수도 있습니다.

참고

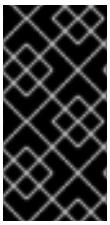
아키텍처에 따라 올바른 제품 변형을 선택합니다.

다음 두 가지 방법으로 관련 끝점, 액세스 키 및 시크릿 액세스 키에 액세스할 수 있습니다.

- 10.2.1절. “터미널에서 Multicloud Object Gateway에 액세스”
- 10.2.2절. “MCG 명령줄 인터페이스에서 Multicloud Object Gateway에 액세스”

가상 호스팅 스타일을 사용하여 MCG 버킷 액세스

> 예 10.1. 예제
>
> 클라이언트 애플리케이션이 https://<bucket-name>.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com에 액세스하려고 하는 경우
>
> 여기서 <bucket-name> 은 MCG 버킷의 이름입니다.
>
> 예: https://mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com
>
> mcg-test-bucket.s3-openshift-storage.apps.mycluster-cluster.qe.rh-ocs.com 에서 S3 서비스를 가리키려면 DNS 항목이 필요합니다.



중요

클라이언트 애플리케이션이 가상 호스팅 스타일을 사용하여 MCG 버킷을 가리키도록 DNS 항목이 있는지 확인합니다.

10.2.1. 터미널에서 Multicloud Object Gateway에 액세스

절차

describe 명령을 실행하여 액세스 키(**AWS_ACCESS_KEY_ID** 값) 및 시크릿 액세스 키(**AWS_SECRET_ACCESS_KEY** 값)를 포함하여 **MCG** 끝점에 대한 정보를 봅니다.

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

출력은 다음과 유사합니다.

```
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

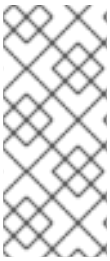
액세스 키 (AWS_ACCESS_KEY_ID 값)

2

시크릿 액세스 키 (**AWS_SECRET_ACCESS_KEY** 값)

3

MCG 엔드 포인트



참고

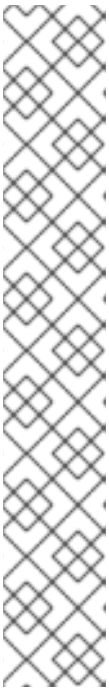
oc describe noobaa 명령의 출력에는 사용 가능한 내부 및 외부 **DNS** 이름이 나열됩니다. 내부 **DNS**를 사용하는 경우 트래픽은 사용 가능합니다. 외부 **DNS**는 **Load Balancing**을 사용하여 트래픽을 처리하므로 시간당 비용이 있습니다.

10.2.2. MCG 명령줄 인터페이스에서 Multicloud Object Gateway에 액세스

사전 요구 사항

- **MCG** 명령줄 인터페이스를 다운로드합니다.

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



참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들면 다음과 같습니다.

- **IBM Power Systems**의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
```

- **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

절차

status 명령을 실행하여 엔드포인트, 액세스 키 및 시크릿 액세스 키에 액세스합니다.

```
noobaa status -n openshift-storage
```

출력은 다음과 유사합니다.

```
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) 엔드포인트

2

액세스 키

3

시크릿 액세스 키

이제 애플리케이션에 연결하기 위해 관련 끝점, 액세스 키 및 비밀 액세스 키가 있어야 합니다.

예 10.2. 예제

AWS S3 CLI가 애플리케이션인 경우 다음 명령은 **OpenShift Container Storage**의 버킷을 나열합니다.

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

10.3. MULTICLOUD OBJECT GATEWAY CONSOLE에 대한 사용자 액세스 허용

사용자에게 **Multicloud Object Gateway Console**에 액세스할 수 있도록 하려면 사용자가 다음 조건을 충족하는지 확인합니다.

- 사용자는 **cluster-admins** 그룹에 있습니다.
- 사용자는 **system:cluster-admins** 가상 그룹에 있습니다.

사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform.**

절차

1. **Multicloud Object Gateway** 콘솔에 대한 액세스를 활성화합니다.

클러스터에서 다음 단계를 한 번 수행합니다.

a. **cluster-admins** 그룹을 만듭니다.

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

b. 그룹을 **cluster-admin** 역할에 바인딩합니다.

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

2. **cluster-admins** 그룹에서 사용자를 추가하거나 제거하여 **Multicloud Object Gateway** 콘솔에 대한 액세스를 제어합니다.

- **cluster-admins** 그룹에 사용자를 추가하려면 다음을 수행합니다.

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

여기서 **<user-name>** 은 추가할 사용자의 이름입니다.

참고

cluster-admins 그룹에 사용자 세트를 추가하는 경우 **OpenShift Container Storage** 대시보드에 액세스할 수 있도록 새로 추가된 사용자를 **cluster-admin** 역할에 바인딩할 필요가 없습니다.

- **cluster-admins** 그룹에서 사용자를 제거하려면 다음을 수행합니다.

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

여기서 **<user-name>** 은 제거할 사용자의 이름입니다.

검증 단계

1. **OpenShift** 웹 콘솔에서 **Multicloud Object Gateway Console**에 대한 액세스 권한이 있는 사

용자로 로그인합니다.

2. 스토리지 → 개요 → 오브젝트 탭 → **Multicloud Object Gateway** 링크를 선택합니다.

3. **Multicloud Object Gateway Console**에서 액세스 권한이 있는 동일한 사용자로 로그인합니다.

4. **Allow selected permissions** (선택한 권한 허용)를 클릭합니다.

10.4. 하이브리드 또는 MULTICLOUD 용 스토리지 리소스 추가

10.4.1. 새 백업 저장소 생성

OpenShift Container Storage에서 새 백업 저장소를 생성하려면 다음 절차를 사용하십시오.

사전 요구 사항

- **OpenShift**에 대한 관리자 액세스.

절차

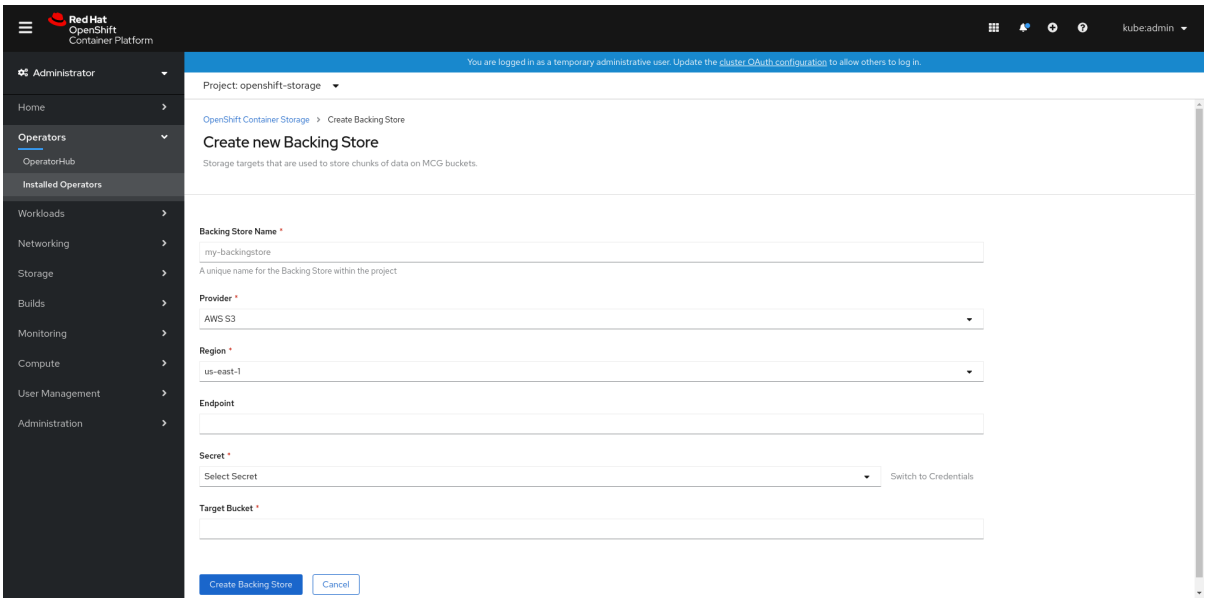
1. **OpenShift** 웹 콘솔의 왼쪽 창에서 **Operators** → 설치된 **Operator** 를 클릭하여 설치된 **Operator**를 확인합니다.

2. **OpenShift Container Storage Operator**를 클릭합니다.

3. **OpenShift Container Storage Operator** 페이지에서 오른쪽 스크롤하고 백업 저장소 탭을 클릭합니다.

4. **Create Backing Store** (백업 저장소 만들기)를 클릭합니다.

그림 10.1. 백업 저장소 페이지 만들기



5. **Create New Backing Store(새 백업 저장소 생성) 페이지에서 다음을 수행합니다.**

a. **백업 저장소 이름을 입력합니다.**

b. **공급업체 선택.**

c. **지역 선택.**

d. **엔드포인트 입력. 이는 선택 사항입니다.**

e. **드롭다운 목록에서 Secret (시크릿)을 선택하거나 고유한 보안을 생성합니다. 필요한 경우 Credentials(자격 증명) 보기로 전환 하여 필요한 시크릿을 입력할 수 있습니다.**

OCP 보안 생성에 대한 자세한 내용은 Openshift Container Platform 설명서의 보안 생성 섹션을 참조하십시오.

각 백업 저장소에는 다른 보안이 필요합니다. 특정 백업 저장소에 대한 시크릿 생성에 대한 자세한 내용은 10.4.2절. "MCG 명령줄 인터페이스를 사용하여 하이브리드 또는 Multicloud용 스토리지 리소스 추가" 을 참조하고 YAML을 사용하여 스토리지 리소스 추가 절차를 따르십시오.

참고

이 메뉴는 **Google Cloud** 및 로컬 **PVC**를 제외한 모든 공급자와 관련이 있습니다.

f. 대상 버킷을 입력합니다. 대상 버킷은 원격 클라우드 서비스에서 호스팅되는 컨테이너 스토리지입니다. **MCG**가 시스템에 이 버킷을 사용할 수 있음을 알려주는 연결을 만들 수 있습니다.

6. **Create Backing Store** (백업 저장소 만들기)를 클릭합니다.

검증 단계

1. **Operators** → 설치된 **Operators**를 클릭합니다.
2. **OpenShift Container Storage Operator**를 클릭합니다.
3. 새 백업 저장소를 검색하거나 백업 저장소 탭을 클릭하여 모든 백업 저장소를 확인합니다.

10.4.2. MCG 명령줄 인터페이스를 사용하여 하이브리드 또는 Multicloud용 스토리지 리소스 추가

MCG(Multicloud Object Gateway)는 클라우드 공급자와 클러스터 전체에서 데이터를 포괄하는 프로세스를 간소화합니다.

MCG에서 사용할 수 있는 백업 스토리지를 추가해야 합니다.

배포 유형에 따라 다음 절차 중 하나를 선택하여 백업 스토리지를 생성할 수 있습니다.

- **AWS** 지원 백업 저장소를 생성하려면 를 참조하십시오. 10.4.2.1절. “AWS 지원 백업 저장소 생성”
- **IBM COS** 지원 백업 저장소를 생성하려면 다음을 참조하십시오. 10.4.2.2절. “IBM COS 지원 백업 저장소 생성”

- Azure 지원 백업 저장소를 생성하려면 를 참조하십시오. 10.4.2.3절. “Azure 지원 백업 저장소 생성”

- GCP 백업 백업 저장소를 생성하려면 를 참조하십시오. 10.4.2.4절. “GCP 지원 백업 저장소 생성”

- 로컬 영구 볼륨 지원 백업 저장소를 생성하려면 다음을 참조하십시오. 10.4.2.5절. “로컬 영구 볼륨 백업 백업 저장소 생성”

VMware 배포의 경우 10.4.3절. “s3 호환 Multicloud Object Gateway 백업 저장소 생성”로 건너뜁니다.

10.4.2.1. AWS 지원 백업 저장소 생성

사전 요구 사항

- MCG(Multicloud Object Gateway) 명령줄 인터페이스를 다운로드합니다.

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

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들어 IBM Z 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

- 또는 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages에 있는 OpenShift Container Storage RPM에서 mcg 패키지를 설치할 수도 있습니다.

참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1.

MCG 명령줄 인터페이스에서 다음 명령을 실행합니다.

```
noobaa backingstore create aws-s3 <backingstore_name> --access-key=<AWS ACCESS KEY> --secret-key=<AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

a.

<backingstore_name> 을 백업 저장소의 이름으로 바꿉니다.

b.

<AWS ACCESS KEY> 및 **<AWS SECRET ACCESS KEY>** 를 이 목적을 위해 생성한 **AWS** 액세스 키 **ID** 및 시크릿 액세스 키로 바꿉니다.

c.

<bucket-name> 을 기존 **AWS** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

출력은 다음과 유사합니다.

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

YAML을 사용하여 스토리지 리소스를 추가할 수도 있습니다.

1.

인증 정보를 사용하여 보안을 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
  namespace: openshift-storage
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

a.

Base64를 사용하여 자체 **AWS** 액세스 키 **ID** 및 시크릿 액세스 키를 제공하고 인코딩해야 하며 **<AWS ACCESS KEY ID ENCODED in BASE64>** 및 **< AWS SECRET ACCESS KEY ENCODED in BASE64>** 대신 결과를 사용해야 합니다.

.

b. **<backingstore-secret-name>** 을 고유한 이름으로 바꿉니다.

2. 특정 백업 저장소에 대해 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <bucket-name>
  type: aws-s3
```

a. **<bucket-name>** 을 기존 **AWS** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

b. **<backingstore-secret-name>** 을 이전 단계에서 생성한 보안의 이름으로 바꿉니다.

10.4.2.2. IBM COS 지원 백업 저장소 생성

사전 요구 사항

- **MCG(Multicloud Object Gateway)** 명령줄 인터페이스를 다운로드합니다.

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

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들면 다음과 같습니다.

- **IBM Power Systems**의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
```

- **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

- 또는 **https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages**에 있는 **OpenShift Container Storage RPM**에서 **mcg** 패키지를 설치할 수도 있습니다.

참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1. **MCG** 명령줄 인터페이스에서 다음 명령을 실행합니다.

```
noobaa backingstore create ibm-cos <backingstore_name> --access-key=<IBM ACCESS KEY> --secret-key=<IBM SECRET ACCESS KEY> --endpoint=<IBM COS ENDPOINT> --target-bucket <bucket-name> -n openshift-storage
```

a. **<backingstore_name>** 을 백업 저장소의 이름으로 바꿉니다.

b. **<IBM ACCESS KEY>, <IBM SECRET ACCESS KEY>, <IBM COS ENDPOINT>** 를 **IBM** 액세스 키 **ID,** 시크릿 액세스 키 및 기존 **IBM** 버킷 위치에 해당하는 해당 지역 엔드포인트로 바꿉니다.

IBM 클라우드에서 위의 키를 생성하려면 대상 버킷에 대한 서비스 자격 증명을 생성하는 동안 **HMAC** 인증 정보를 포함해야 합니다.

c.

<bucket-name> 을 기존 **IBM** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

출력은 다음과 유사합니다.

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

YAML을 사용하여 스토리지 리소스를 추가할 수도 있습니다.

1.

인증 정보를 사용하여 보안을 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <backingstore-secret-name>
type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN
BASE64>
```

a.

Base64를 사용하여 자체 **IBM COS** 액세스 키 **ID** 및 비밀 액세스 키를 제공하고 인코딩해야 하며 **<IBM COS COS** 키 **ID ENCODED IN BASE64>** 및 **< IBM COS SECRET ACCESS KEY ENCODED IN BASE64>** 대신 결과를 사용해야 합니다.

b.

<backingstore-secret-name> 을 고유한 이름으로 바꿉니다.

2.

특정 백업 저장소에 대해 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
```

```
ibmCos:
  endpoint: <endpoint>
  secret:
    name: <backingstore-secret-name>
    namespace: openshift-storage
  targetBucket: <bucket-name>
type: ibm-cos
```

a. **<bucket-name>** 을 기존 **IBM COS** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

b. **<endpoint>** 를 기존 **IBM** 버킷 이름의 위치에 해당하는 지역 끝점으로 바꿉니다. 이 인수는 백업 저장소에 사용할 엔드포인트를 **Multicloud Object Gateway**에 지시하고 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

c. **<backingstore-secret-name>** 을 이전 단계에서 생성한 보안의 이름으로 바꿉니다.

10.4.2.3. Azure 지원 백업 저장소 생성

사전 요구 사항

- **MCG(Multicloud Object Gateway)** 명령줄 인터페이스를 다운로드합니다.

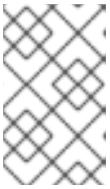
```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들어 **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

- 또는 **https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages**에 있는 **OpenShift Container Storage RPM**에서 **mcg** 패키지를 설치할 수도 있습니다.



참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1. MCG 명령줄 인터페이스에서 다음 명령을 실행합니다.

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

a. <backingstore_name> 을 백업 저장소의 이름으로 바꿉니다.

b. <AZURE ACCOUNT KEY> 및 <AZURE ACCOUNT NAME> 을 이 목적을 위해 생성한 AZURE 계정 키 및 계정 이름으로 바꿉니다.

c. <blob 컨테이너 이름> 을 기존 Azure Blob 컨테이너 이름으로 바꿉니다. 이 인수는 Multicloud Object Gateway에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

출력은 다음과 유사합니다.

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

YAML을 사용하여 스토리지 리소스를 추가할 수도 있습니다.

1. 인증 정보를 사용하여 보안을 생성합니다.

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

a. **Base64**를 사용하여 자체 **Azure** 계정 이름과 계정 키를 제공하고 인코딩해야 하며 **<AZURE ACCOUNT NAME ENCODED IN BASE64>** 및 **< AZURE ACCOUNT KEY ENCODED IN BASE64>** 대신 결과를 사용해야 합니다.

b. **<backingstore-secret-name>** 을 고유한 이름으로 바꿉니다.

2. 특정 백업 저장소에 대해 다음 **YAML**을 적용합니다.

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

a. **<blob-container-name>** 을 기존 **Azure Blob** 컨테이너 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

b. **<backingstore-secret-name>** 을 이전 단계에서 생성한 보안의 이름으로 바꿉니다.

10.4.2.4. GCP 지원 백업 저장소 생성

사전 요구 사항

- **MCG(Multicloud Object Gateway)** 명령줄 인터페이스를 다운로드합니다.

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

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들어 **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

- 또는 **https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages**에 있는 **OpenShift Container Storage RPM**에서 **mcg** 패키지를 설치할 수도 있습니다.

참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1. **MCG** 명령줄 인터페이스에서 다음 명령을 실행합니다.

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

a. **<backingstore_name>** 을 백업 저장소의 이름으로 바꿉니다.

b. **<PATH TO GCP PRIVATE KEY JSON FILE>** 을 이 목적을 위해 생성된 **GCP** 개인 키의 경로로 바꿉니다.

c. **<GCP** 버킷 이름**>** 을 기존 **GCP** 오브젝트 스토리지 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

출력은 다음과 유사합니다.

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

YAML을 사용하여 스토리지 리소스를 추가할 수도 있습니다.

1. 인증 정보를 사용하여 보안을 생성합니다.

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

   a. **Base64**를 사용하여 자체 **GCP** 서비스 계정 개인 키를 제공하고 인코딩해야 하며 **<GCP PRIVATE KEY ENCODED IN BASE64>** 대신 결과를 사용해야 합니다.

   b. **<backingstore-secret-name>**을 고유한 이름으로 바꿉니다.

2. 특정 백업 저장소에 대해 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
spec:
  googleCloudStorage:
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    targetBucket: <target bucket>
  type: google-cloud-storage
```

   a. **<target bucket>** 을 기존 **Google** 스토리지 버킷으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

   b. **<backingstore-secret-name>** 을 이전 단계에서 생성한 보안의 이름으로 바꿉니다.

10.4.2.5. 로컬 영구 볼륨 백업 백업 저장소 생성

사전 요구 사항

- **MCG(Multicloud Object Gateway)** 명령줄 인터페이스를 다운로드합니다.

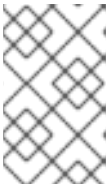
```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-x86_64-rpms
# yum install mcg
```

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들어 **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

- 또는 **https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/packages**에 있는 **OpenShift Container Storage RPM**에서 **mcg** 패키지를 설치할 수도 있습니다.



참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1. **MCG** 명령줄 인터페이스에서 다음 명령을 실행합니다.

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

   a. **<backingstore_name>** 을 백업 저장소의 이름으로 바꿉니다.

   b. **<NUMBER OF VOLUMES>** 를 생성하려는 볼륨 수로 바꿉니다. 볼륨 수를 늘리면 스토리지가 확장됩니다.

   c. **<VOLUME SIZE>** 를 각 볼륨의 필수 크기 **(GB)**로 바꿉니다.

d.

ocs-storagecluster-ceph-rbd를 사용하는 데 권장되는 로컬 스토리지 클래스로 **<LOCAL STORAGE CLASS>** 를 바꿉니다.

출력은 다음과 유사합니다.

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

YAML을 사용하여 스토리지 리소스를 추가할 수도 있습니다.

1.

특정 백업 저장소에 대해 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore_name>
  namespace: openshift-storage
spec:
   pvPool:
    numVolumes: <NUMBER OF VOLUMES>
    resources:
      requests:
        storage: <VOLUME SIZE>
    storageClass: <LOCAL STORAGE CLASS>
  type: pv-pool
```

a.

<backingstore_name> 을 백업 저장소의 이름으로 바꿉니다.

b.

<NUMBER OF VOLUMES> 를 생성하려는 볼륨 수로 바꿉니다. 볼륨 수를 늘리면 스토리지가 확장됩니다.

c.

<VOLUME SIZE> 를 각 볼륨의 필수 크기**(GB)**로 바꿉니다. 문자 **G**는 유지되어야 합니다

d.

ocs-storagecluster-ceph-rbd를 사용하는 데 권장되는 로컬 스토리지 클래스로 **<LOCAL STORAGE CLASS>** 를 바꿉니다.

10.4.3. s3 호환 Multicloud Object Gateway 백업 저장소 생성

Multicloud Object Gateway는 S3 호환 오브젝트 스토리지를 백업 저장소(예: Red Hat Ceph Storage의 RADOS Gateway)로 사용할 수 있습니다. 다음 절차에서는 Red Hat Ceph Storage의 RADOS 게이트웨이에 대해 S3 호환 Multicloud Object Gateway 백업 저장소를 생성하는 방법을 보여줍니다. RGW를 배포할 때 Openshift Container Storage Operator는 Multicloud Object Gateway에 대한 S3 호환 백업 저장소를 자동으로 생성합니다.

절차

1. **MCG(Multicloud Object Gateway) 명령줄 인터페이스에서 다음 NooBaa 명령을 실행합니다.**

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

a. **<RGW ACCESS KEY> 및 <RGW SECRET KEY> 를 가져오려면 RGW 사용자 시크릿 이름을 사용하여 다음 명령을 실행합니다.**

```
oc get secret <RGW USER SECRET NAME> -o yaml -n openshift-storage
```

b. **Base64에서 액세스 키 ID와 액세스 키를 디코딩하고 유지합니다.**

c. **<RGW USER ACCESS KEY> 및 <RGW USER SECRET ACCESS KEY> 를 이전 단계의 적절한 디코딩된 데이터로 바꿉니다.**

d. **<bucket-name> 을 기존 RGW 버킷 이름으로 바꿉니다. 이 인수는 Multicloud Object Gateway에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.**

e. **<RGW 엔드포인트> 를 가져오려면 RADOS 개체 게이트웨이 S3 끝점 액세스를 참조하십시오.**

출력은 다음과 유사합니다.

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

YAML을 사용하여 백업 저장소를 생성할 수도 있습니다.

1. **CephObjectStore** 사용자를 만듭니다. 이렇게 하면 **RGW** 인증 정보가 포함된 보안도 생성됩니다.

```
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>** 및 **<Display-name>** 을 고유한 사용자 이름 및 표시 이름으로 바꿉니다.

2. **S3** 호환 백업 저장소에 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BackingStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <backingstore-name>
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <RGW endpoint>
    secret:
      name: <backingstore-secret-name>
      namespace: openshift-storage
    signatureVersion: v4
    targetBucket: <RGW-bucket-name>
  type: s3-compatible
```

   a. **<backingstore-secret-name>** 을 이전 단계에서 **CephObjectStore** 로 생성된 시크릿 이름으로 바꿉니다.

   b. **<bucket-name>** 을 기존 **RGW** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

c.

<RGW 엔드포인트> 를 가져오려면 RADOS 개체 게이트웨이 S3 끝점 액세스를 참조하십시오.

10.4.4. 사용자 인터페이스를 사용하여 하이브리드 및 Multicloud용 스토리지 리소스 추가

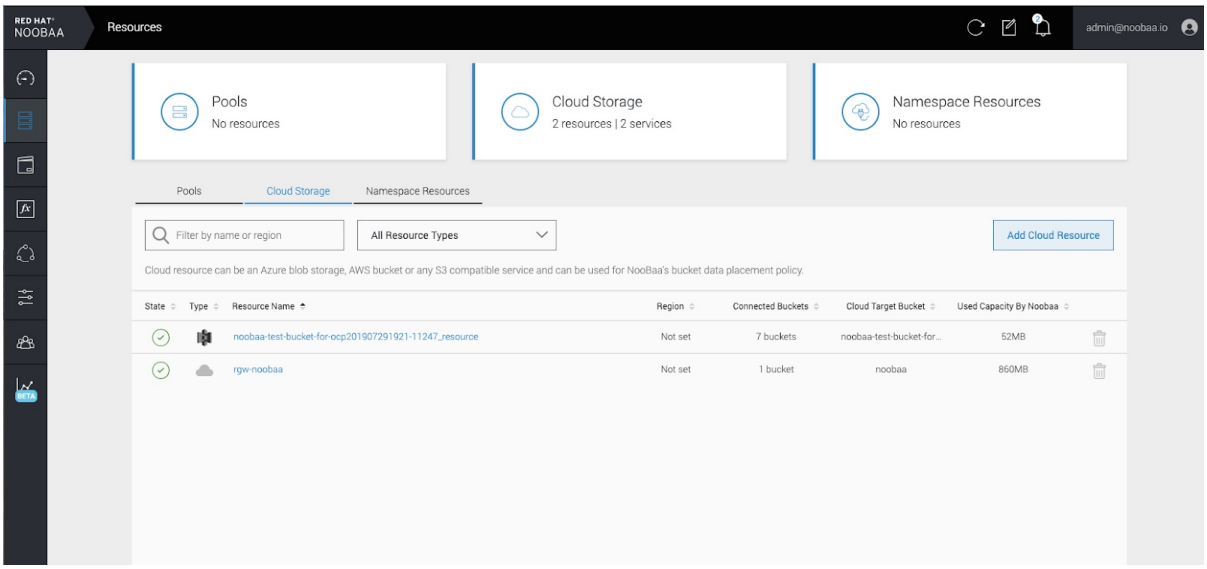
절차

1.

OpenShift Storage 콘솔에서 스토리지 → 개요 → 오브젝트 탭 → Multicloud Object Gateway 링크를 클릭합니다.

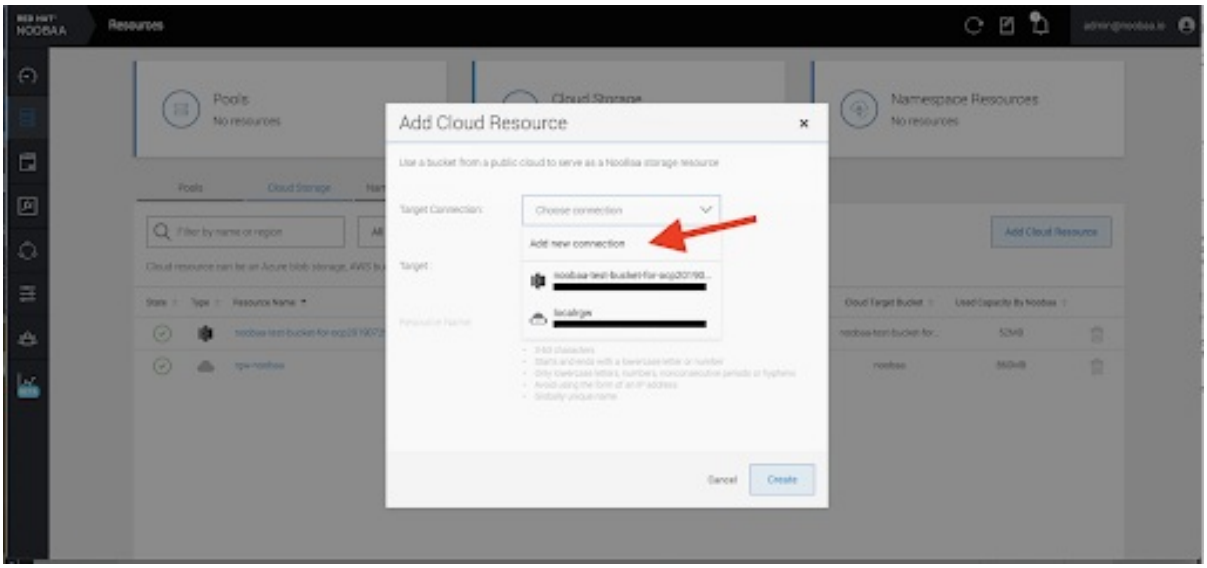
2.

아래에 강조 표시된 왼쪽에서 Resources(리소스) 탭을 선택합니다. 채워지는 목록에서 클라우드 리소스 추가를 선택합니다.

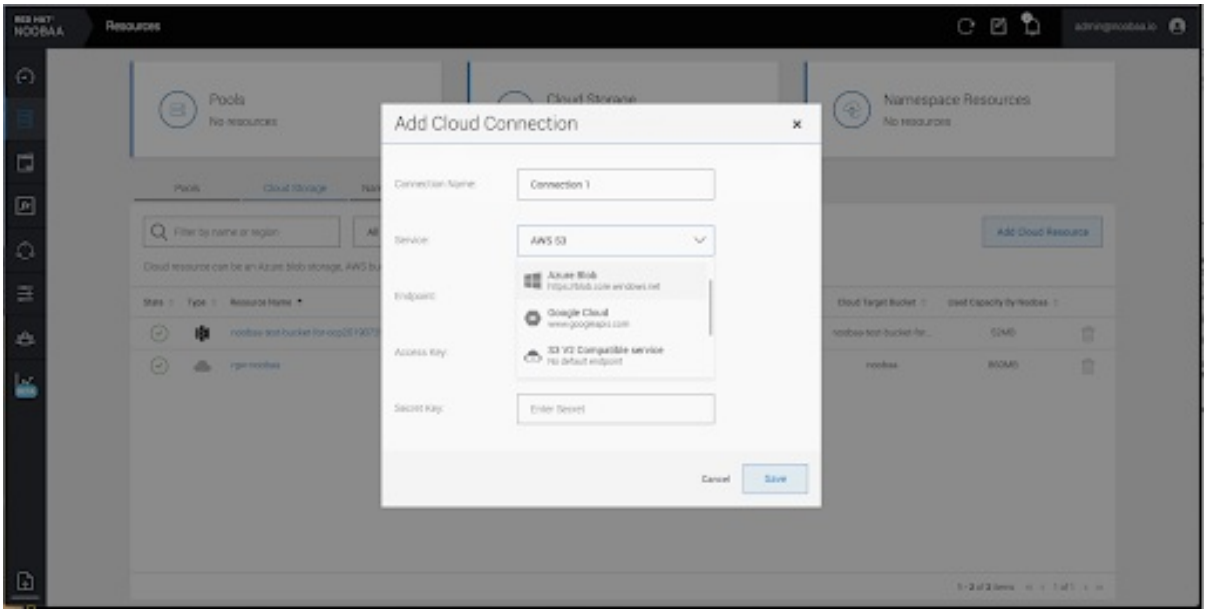


3.

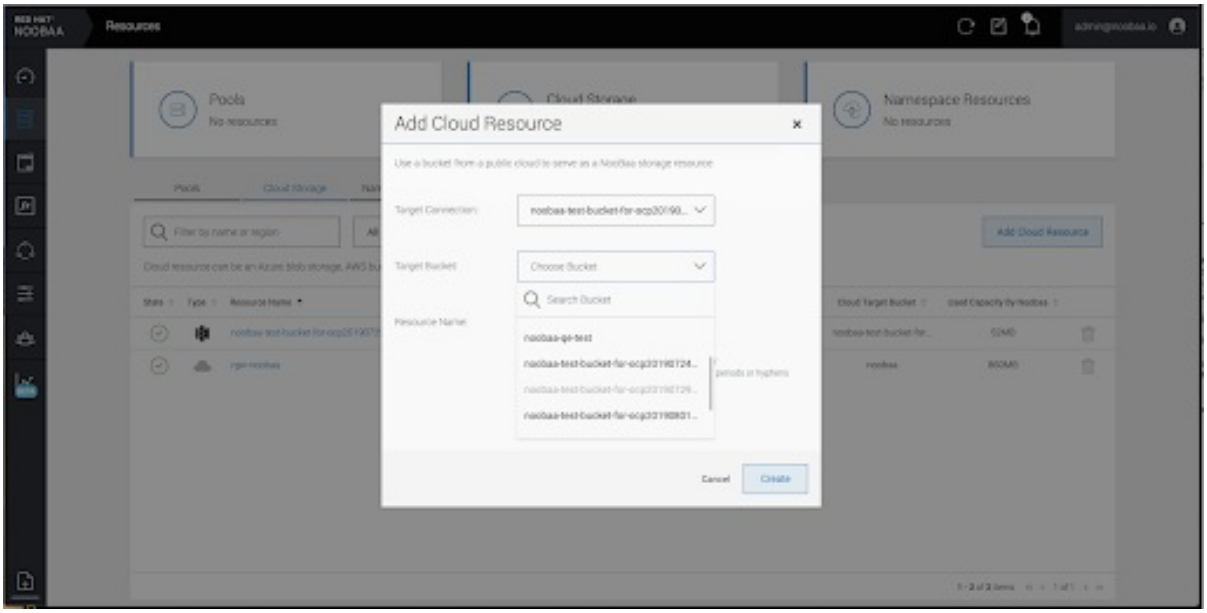
Add new connection(새 연결 추가)을 선택합니다.



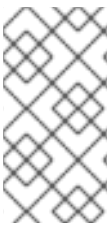
4. 관련 기본 클라우드 공급자 또는 **S3** 호환 옵션을 선택하고 세부 정보를 입력합니다.



5. 새로 생성된 연결을 선택하고 기존 버킷에 매핑합니다.



6. 이 단계를 반복하여 필요한 만큼의 백업 저장소를 만듭니다.



참고

NooBaa UI에서 생성된 리소스는 **OpenShift UI** 또는 **MCG CLI**에서 사용할 수 없습니다.

10.4.5. 새 버킷 클래스 생성

버킷 클래스는 **OBC(**오브젝트 버킷 클래스**)**에 대한 계층화 정책 및 데이터 배치를 정의하는 버킷 클래스를 나타내는 **CRD**입니다.

OpenShift Container Storage에서 버킷 클래스를 생성하려면 다음 절차를 사용하십시오.

절차

1. **OpenShift** 웹 콘솔의 왼쪽 창에서 **Operators** → 설치된 **Operator** 를 클릭하여 설치된 **Operator**를 확인합니다.
2. **OpenShift Container Storage Operator**를 클릭합니다.
3. **OpenShift Container Storage Operator** 페이지에서 오른쪽 스크롤하고 버킷 클래스 탭을 클릭합니다.
4. **Create** 버킷 클래스(버킷 클래스 만들기)를 클릭합니다.
5. 새 버킷 클래스 생성 페이지에서 다음을 수행합니다.
   a. 버킷 클래스 유형을 선택하고 버킷 클래스 이름을 입력합니다.
      i. 버킷 클래스 유형을 선택합니다. 다음 옵션 중 하나를 선택합니다.
         - 네임스페이스

           데이터는 중복 제거, 압축 또는 암호화를 수행하지 않고 **NamespaceStores**에 저장됩니다.
         - **Standard**

           데이터는 **MCG(Multicloud Object Gateway)**에서 사용하며 중복, 압축 및 암호화됩니다.

기본적으로 **Standard** 가 선택됩니다.

ii. 버킷 클래스 이름을 입력합니다.

iii. 다음을 클릭합니다.

b. **Placement Policy**(배치 정책)에서 **Tier 1 - Policy Type**(계층 **1 - Policy Type**)을 선택하고 **Next** (다음)를 클릭합니다. 요구 사항에 따라 옵션 중 하나를 선택할 수 있습니다.

- 분산 을 사용하면 선택한 리소스 전체에 데이터를 분산할 수 있습니다.
- 미러 를 사용하면 선택한 리소스에서 데이터를 완전히 복제할 수 있습니다.
- **Add Tier** (계층 추가)를 클릭하여 다른 정책 계층을 추가합니다.

c. **Tier 1 - Policy Type**을 **Spread**로 선택한 경우 사용 가능한 목록에서 **atleast** 하나의 백업 저장소 리소스를 선택하고 **Next** (다음)를 클릭합니다. 또는 새 백업 저장소를 만들 수도 있습니다.

참고

이전 단계에서 **Policy Type**(정책 유형)을 **mirror**로 선택할 때 **atleast 2** 백업 저장소를 선택해야 합니다.

d. 버킷 클래스 설정을 검토하고 확인합니다.

e. **Create** 버킷 클래스(버킷 클래스 만들기)를 클릭합니다.

검증 단계

1. **Operators** → 설치된 **Operators**를 클릭합니다.

2. **OpenShift Container Storage Operator**를 클릭합니다.

3. 새 버킷 클래스를 검색하거나 버킷 클래스 탭을 클릭하여 모든 버킷 클래스를 확인합니다.

10.4.6. 버킷 클래스 편집

Openshift 웹 콘솔에서 편집 버튼을 클릭하여 **YAML** 파일을 통해 버킷 클래스 구성 요소를 편집하려면 다음 절차를 사용합니다.

사전 요구 사항

- **OpenShift**에 대한 관리자 액세스.

절차

1. **OpenShift** 웹 콘솔에 로그인합니다.

2. **Operators** → 설치된 **Operators**를 클릭합니다.

3. **OpenShift Container Storage Operator**를 클릭합니다.

4. **OpenShift Container Storage Operator** 페이지에서 오른쪽 스크롤하고 버킷 클래스 탭을 클릭합니다.

5. 편집할 버킷 클래스 옆에 있는 작업 메뉴(**kube**)를 클릭합니다.

6. **Edit** 버킷 **Class(** 버킷 클래스 편집)를 클릭합니다.

7. **YAML** 파일로 리디렉션되어 이 파일에 필요한 변경 작업을 수행하고 저장을 클릭합니다.

10.4.7. 버킷 클래스의 백업 저장소 편집

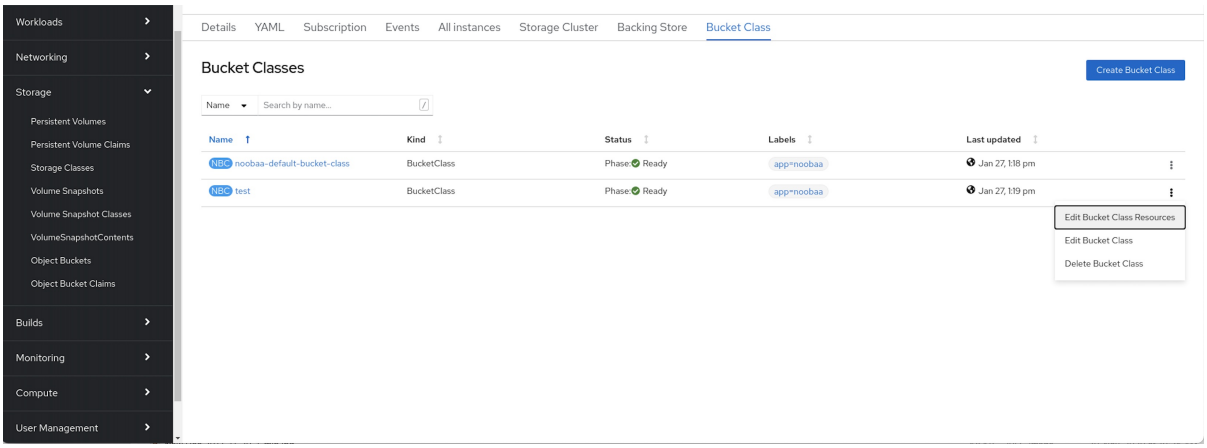
다음 절차에 따라 기존 **Multicloud Object Gateway** 버킷 클래스를 편집하여 버킷 클래스에 사용된 기본 백업 저장소를 변경합니다.

사전 요구 사항

- **OpenShift** 웹 콘솔에 대한 관리자 액세스.
- 버킷 클래스.
- 백업 저장소.

절차

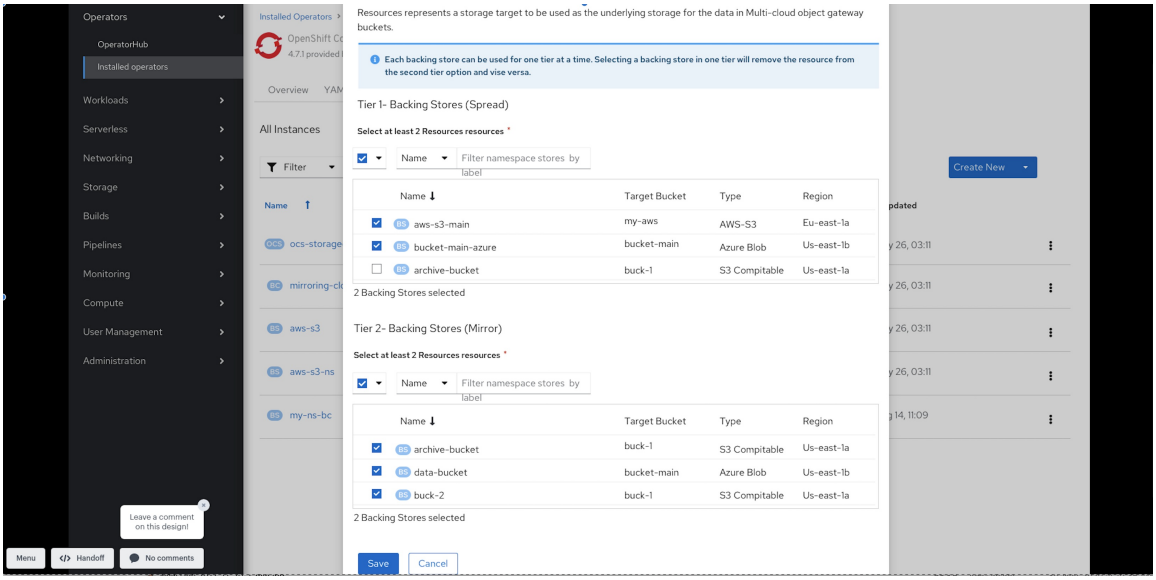
1. **Operators** → 설치된 **Operator**를 클릭하여 설치된 **Operator**를 확인합니다.
2. **OpenShift Container Storage Operator**를 클릭합니다.
3. 버킷 클래스 탭을 클릭합니다.
4. 편집할 버킷 클래스 옆에 있는 작업 메뉴(**kube**)를 클릭합니다.



5. **Edit** 버킷 **Class Resources(** 버킷 클래스 리소스 편집)를 클릭합니다.
6. 버킷 클래스 리소스 편집 페이지에서 버킷 클래스에 백업 저장소를 추가하거나 버킷 클래스에서 백업 저장소를 제거하여 버킷 클래스 리소스를 편집합니다. 하나 또는 두 개의 계층과 다양

한 배치 정책을 사용하여 생성된 버킷 클래스 리소스를 편집할 수도 있습니다.

- 버킷 클래스에 백업 저장소를 추가하려면 백업 저장소의 이름을 선택합니다.
- 버킷 클래스에서 백업 저장소를 제거하려면 백업 저장소의 이름을 지웁니다.



7. 저장을 클릭합니다.

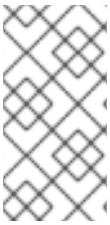
10.5. 네임스페이스 버킷 관리

네임 스페이스 버킷을 사용하면 여러 공급자의 데이터 리포지토리를 함께 연결할 수 있으므로 단일 통합 뷰를 통해 모든 데이터와 상호 작용할 수 있습니다. 각 공급자와 연결된 오브젝트 버킷을 네임 스페이스 버킷에 추가하고 네임 스페이스 버킷을 통해 데이터에 액세스하여 모든 오브젝트 버킷을 한 번에 확인합니다. 이를 통해 원하는 스토리지 공급자에 쓰는 동시에 다른 여러 스토리지 공급자로부터 읽을 수 있어 새 스토리지 공급자로 마이그레이션하는 데 드는 비용을 크게 줄일 수 있습니다.

1. 공급업체를 **Multicloud Object Gateway**에 연결합니다.
2. 네임스페이스 버킷에 추가할 수 있도록 각 공급자에 대한 네임스페이스 리소스를 생성합니다.
3. 네임 스페이스 버킷에 네임 스페이스 리소스를 추가하고 적절한 네임 스페이스 리소스를 읽고 쓸 수 있도록 버킷을 구성합니다.

S3 API를 사용하여 네임스페이스 버킷의 오브젝트와 상호 작용할 수 있습니다. 자세한 내용은 네임스

페이스 버킷의 오브젝트에 대한 S3 API 끝점 을 참조하십시오.



참고

네임 스페이스 버킷은 쓰기 대상이 사용 가능하고 작동하는 경우에만 사용할 수 있습니다.

10.5.1. Multicloud Object Gateway에 공급자 연결 추가

Multicloud Object Gateway에서 공급자에 액세스할 수 있도록 각 공급업체에 대한 연결을 추가해야 합니다.

사전 요구 사항

- OpenShift 콘솔에 대한 관리자 액세스.

절차

1. OpenShift 콘솔에서 스토리지 → 개요 를 클릭하고 오브젝트 탭을 클릭합니다.
2. Multicloud Object Gateway (다중 오브젝트 게이트웨이)를 클릭하고 메시지가 표시되면 로그인합니다.
3. 계정을 클릭하고 계정을 선택하여 연결을 추가합니다.
4. My Connections (내 연결)를 클릭합니다.
5. Add Connection (연결 추가)을 클릭합니다.
   a. 연결 이름 입력.
   b. 기본적으로 클라우드 공급자가 서비스 드롭다운에 표시됩니다. 다른 공급자를 사용하도록 선택을 변경합니다.

c. 클라우드 공급자의 기본 엔드 포인트는 기본적으로 엔드포인트 필드에 표시됩니다. 필요한 경우 대체 끝점을 입력합니다.

d. 이 클라우드 공급자에 대한 액세스 키를 입력합니다.

e. 이 클라우드 공급자의 비밀 키를 입력합니다.

f. 저장을 클릭합니다.

10.5.2. Multicloud Object Gateway를 사용하여 네임스페이스 리소스 추가

기존 스토리지를 Multicloud Storage Gateway에 네임스페이스 리소스로 추가하여 Amazon Web Services S3 버킷, Microsoft Azure Blob 및 IBM Cloud Object Storage 버킷과 같은 기존 스토리지 대상을 통합하기 위해 네임 스페이스 버킷에 포함할 수 있습니다.

사전 요구 사항

- OpenShift 콘솔에 대한 관리자 액세스.
- 대상 연결(공급업체)이 이미 Multicloud Object Gateway에 추가되었습니다. 자세한 내용은 10.5.1절. “Multicloud Object Gateway에 공급자 연결 추가” 을 참조하십시오.

절차

1. OpenShift 콘솔에서 스토리지 → 개요 를 클릭하고 개체 탭을 클릭합니다.
2. Multicloud Storage Gateway (다중 스토리지 게이트웨이)를 클릭하고 메시지가 표시되면 로그인합니다.
3. Resources(리소스)를 클릭하고 Namespace Resources(네임스페이스 리소스) 탭을 클릭합니다.
4. 네임스페이스 리소스 생성을 클릭합니다.

a. **Target Connection** (대상 연결)에서 이 네임스페이스의 스토리지 공급자에 사용할 연결을 선택합니다.

새 연결을 추가해야 하는 경우 새 연결 추가를 클릭하고 공급자 세부 정보를 입력합니다. 자세한 내용은 10.5.1절. “Multicloud Object Gateway에 공급자 연결 추가” 을 참조하십시오.

b. 대상 버킷 에서 대상으로 사용할 버킷의 이름을 선택합니다.

c. 네임스페이스 리소스의 리소스 이름을 입력합니다.

d. 생성을 클릭합니다.

검증

- 새 리소스가 **State** 열에 녹색 확인 표시로 나열되고 연결된 네임스페이스 버킷 열에 **0**개의 버킷이 나열되는지 확인합니다.

10.5.3. Multicloud Object Gateway를 사용하여 네임 스페이스 버킷에 리소스 추가

다양한 공급업체에서 스토리지를 통합하기 위해 네임 스페이스 버킷에 네임스페이스 리소스를 추가합니다. 또한 하나의 공급업체만 새 데이터를 수락하고 모든 공급업체에서 기존 데이터를 읽을 수 있도록 읽기 및 쓰기 동작을 구성할 수도 있습니다.

사전 요구 사항

- 버킷에서 처리할 모든 네임 스페이스 리소스가 **Multicloud Object Gateway**에 추가되었는지 확인합니다. Multicloud Object Gateway를 사용하여 네임스페이스 리소스 추가.

절차

1. **OpenShift** 콘솔에서 스토리지 → 개요 를 클릭하고 오브젝트 탭을 클릭합니다.

2. **Multicloud Object Gateway** (다중 오브젝트 게이트웨이)를 클릭하고 메시지가 표시되면 로그인합니다.

3. 버킷을 클릭하고 네임 스페이스 버킷 탭을 클릭합니다.

4. 네임스페이스 버킷 생성을 클릭합니다.

   a. **Choose Name** (이름 선택) 탭에서 네임스페이스 버킷 의 이름을 지정하고 **Next** (다음)를 클릭합니다.

   b. **Set Placement** (배치 설정) 탭에서 다음을 수행합니다.

      i. **Read Policy** (읽기 정책)에서 네임스페이스 버킷이 데이터를 읽어야 하는 각 네임스페이스 리소스의 확인란을 선택합니다.

      ii. **Write Policy** (쓰기 정책)에서 네임스페이스 버킷에서 데이터를 써야 하는 네임스페이스 리소스를 지정합니다.

      iii. 다음을 클릭합니다.

   c. 프로덕션 환경의 **Set Caching Policy** (캐싱 정책 설정) 탭에서 변경하지 마십시오. 이 탭은 개발 프리뷰로 제공되며 제한 사항을 지원할 수 있습니다.

   d. 생성을 클릭합니다.

검증

- **State** 열, 예상되는 읽기 리소스 수 및 예상 쓰기 리소스 이름에 녹색 확인 표시가 포함된 네임스페이스 버킷이 나열되는지 확인합니다.

10.5.4. 네임스페이스 버킷의 오브젝트에 대한 Amazon S3 API 끝점

Amazon S3(Simple Storage Service) API를 사용하여 네임스페이스 버킷의 오브젝트와 상호 작용할 수 있습니다.

Red Hat OpenShift Container Storage 4.6 이후에서는 다음과 같은 네임 스페이스 버킷 작업을 지원

합니다.

- ListObjectVersions
- ListObjects
- PutObject
- CopyObject
- ListParts
- CreateMultipartUpload
- CompleteMultipartUpload
- UploadPart
- UploadPartCopy
- AbortMultipartUpload
- GetObjectAcl
- GetObject
- HeadObject

- DeleteObject
- DeleteObjects

이러한 작업 및 사용 방법에 대한 최신 정보는 **Amazon S3 API** 참조 문서를 참조하십시오.

추가 리소스

- Amazon S3 REST API 참조
- Amazon S3 CLI 참조

10.5.5. Multicloud Object Gateway CLI 및 YAML을 사용하여 네임스페이스 버킷 추가

네임스페이스 버킷에 대한 자세한 내용은 네임스페이스 버킷 관리를 참조하십시오.

배포 유형 및 **YAML** 또는 **Multicloud Object Gateway CLI**를 사용할지 여부에 따라 다음 절차 중 하나를 선택하여 네임 스페이스 버킷을 추가합니다.

- YAML을 사용하여 AWS S3 네임 스페이스 버킷 추가
- YAML을 사용하여 IBM COS 네임 스페이스 버킷 추가
- Multicloud Object Gateway CLI를 사용하여 AWS S3 네임 스페이스 버킷 추가
- Multicloud Object Gateway CLI를 사용하여 IBM COS 네임 스페이스 버킷 추가

10.5.5.1. YAML을 사용하여 AWS S3 네임 스페이스 버킷 추가

사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform**

- **Multicloud Object Gateway**에 액세스하고 **2**장, 애플리케이션을 사용하여 **Multicloud Object Gateway**에액세스

절차

1. 인증 정보를 사용하여 보안을 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
name: <namespacestore-secret-name>
type: Opaque
data:
AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

a. **Base64**를 사용하여 자체 **AWS** 액세스 키 **ID** 및 시크릿 액세스 키를 제공하고 인코딩해야 하며 **<AWS ACCESS KEY ID ENCODED in BASE64>** 및 **< AWS SECRET ACCESS KEY ENCODED in BASE64>** 대신 결과를 사용해야 합니다**. ii. <namespacestore-secret-name>** 을 고유한 이름으로 바꿉니다**.**

2. **OpenShift CRD(Custom Resource Definitions)**를 사용하여 네임스페이스 저장소 리소스를 생성합니다**.** 네임 스페이스 저장소는 **Multicloud Object Gateway** 네임 스페이스 버킷의 데이터에 대한 읽기 또는 쓰기 대상으로 사용할 기본 스토리지를 나타냅니다**.** 네임스페이스 저장소 리소스를 생성하려면 다음 **YAML**을 적용합니다**.**

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <resource-name>
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    targetBucket: <target-bucket>
  type: aws-s3
```

a.
<resource-name> 을 리소스에 제공할 이름으로 바꿉니다.

b.
<namespacestore-secret-name> 을 **1** 단계에서 생성된 보안으로 교체합니다.

c.
<namespace-secret> 을 보안을 찾을 수 있는 네임스페이스로 바꿉니다.

d.
<target-bucket> 을 네임 스페이스 저장소에 대해 생성한 대상 버킷으로 바꿉니다.

3.
네임스페이스 버킷에 대한 네임스페이스 정책을 정의하는 네임스페이스 버킷 클래스를 생성합니다. 네임스페이스 정책에는 단일 또는 **multi** 의 유형이 필요합니다.

- **type single** 의 네임스페이스 정책에는 다음과 같은 구성이 필요합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type:
    single:
      resource: <resource>
```

<my-bucket-class> 를 고유한 네임 스페이스 버킷 클래스 이름으로 바꿉니다.

<resource> 를 네임 스페이스 버킷의 읽기 및 쓰기 대상을 정의할 단일 네임 스페이스 저장소의 이름으로 교체합니다.

- **type multi** 의 네임스페이스 정책에는 다음과 같은 구성이 필요합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
```

```
  namespace: openshift-storage
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
      - <read-resources>
      - <read-resources>
```

<my-bucket-class> 를 고유한 버킷 클래스 이름으로 바꿉니다.

<write-resource> 를 단일 네임 스페이스 저장소 이름으로 교체하여 네임 스페이스 버킷의 쓰기 대상을 정의합니다.

<read-resources> 를 네임 스페이스 버킷의 읽기 대상을 정의할 **namespace-stores** 의 이름 목록으로 바꿉니다.

4. 다음 **YAML**을 적용하여 **2**단계에 정의된 버킷 클래스를 사용하는 **OBC(**개체 버킷 클래스**)** 리소스를 사용하여 버킷을 생성합니다.

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
  namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: noobaa.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

참고

IBM Power Systems 및 **IBM Z** 인프라의 경우 **openshift-storage.noobaa.io**로 **use storageClassName**

a. **<my-bucket-class>** 를 이전 단계에서 생성한 버킷 클래스로 바꿉니다.

Operator가 **OBC**를 프로비저닝하면 **Multicloud Object Gateway**에서 버킷이 생성되고 **Operator**는 **OBC**의 동일한 네임스페이스에서 **OBC**의 동일한 이름으로 **Secret** 및 **ConfigMap**을 생성합니다.

10.5.5.2. YAML을 사용하여 IBM COS 네임 스페이스 버킷 추가

사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform**
- **Multicloud Object Gateway**에 액세스하고 **2**장, 애플리케이션을 사용하여 **Multicloud Object Gateway**에액세스

절차

1. 인증 정보를 사용하여 보안을 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
name: <namespacestore-secret-name>
type: Opaque
data:
IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN
BASE64>
```

   a. **Base64**를 사용하여 자체 **IBM COS** 액세스 키 **ID** 및 비밀 액세스 키를 제공하고 인코딩해야 하며 **<IBM COS COS** 키 **ID ENCODED IN BASE64>** 및 **'< IBM COS SECRET ACCESS KEY ENCODED IN BASE64>** 대신 결과를 사용해야 합니다.

   b. **<namespacestore-secret-name>** 을 고유한 이름으로 바꿉니다.

2. **OpenShift CRD(Custom Resource Definitions)**를 사용하여 네임스페이스 저장소 리소스를 생성합니다. 네임 스페이스 저장소는 **Multicloud Object Gateway** 네임 스페이스 버킷의 데이터에 대한 읽기 또는 쓰기 대상으로 사용할 기본 스토리지를 나타냅니다. 네임스페이스 저장소 리소스를 생성하려면 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: bs
  namespace: openshift-storage
```

```
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    signatureVersion: v2
    targetBucket: <target-bucket>
  type: ibm-cos
```

a. **<IBM COS ENDPOINT>** 를 적절한 **IBM COS** 엔드포인트로 바꿉니다.

b. **<namespacestore-secret-name>** 을 **1** 단계에서 생성된 보안으로 교체합니다.

c. **<namespace-secret>** 을 보안을 찾을 수 있는 네임스페이스로 바꿉니다.

d. **<target-bucket>** 을 네임 스페이스 저장소에 대해 생성한 대상 버킷으로 바꿉니다.

3. 네임스페이스 버킷에 대한 네임스페이스 정책을 정의하는 네임스페이스 버킷 클래스를 생성합니다. 네임스페이스 정책에는 단일 또는 **multi** 의 유형이 필요합니다.

- **type single** 의 네임스페이스 정책에는 다음과 같은 구성이 필요합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type:
    single:
      resource: <resource>
```

<my-bucket-class> 를 고유한 네임 스페이스 버킷 클래스 이름으로 바꿉니다.

<resource> 를 단일 네임 스페이스 저장소 이름으로 교체하여 네임 스페이스 버킷의 읽기 및 쓰기 대상을 정의합니다.

- **type multi** 의 네임스페이스 정책에는 다음과 같은 구성이 필요합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <my-bucket-class>
  namespace: openshift-storage
spec:
  namespacePolicy:
    type: Multi
    multi:
      writeResource: <write-resource>
      readResources:
      - <read-resources>
      - <read-resources>
```

<my-bucket-class> 를 고유한 버킷 클래스 이름으로 바꿉니다.

<write-resource> 를 단일 네임 스페이스 저장소 이름으로 교체하여 네임 스페이스 버킷의 쓰기 대상을 정의합니다.

<read-resources> 를 네임 스페이스 버킷의 읽기 대상을 정의하는 **namespace-store** 의 이름 목록으로 바꿉니다.

4. 다음 **YAML**을 적용하여 **2**단계에 정의된 버킷 클래스를 사용하는 **OBC(**개체 버킷 클래스**)** 리소스를 사용하여 버킷을 생성합니다.

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <resource-name>
  namespace: openshift-storage
spec:
  generateBucketName: <my-bucket>
  storageClassName: noobaa.noobaa.io
  additionalConfig:
    bucketclass: <my-bucket-class>
```

참고

IBM Power Systems 및 IBM Z 인프라의 경우 openshift-storage.noobaa.io로 use storageClassName

a.

<my-bucket-class> 를 이전 단계에서 생성한 버킷 클래스로 바꿉니다.

Operator가 OBC를 프로비저닝하면 Multicloud Object Gateway에서 버킷이 생성되고 Operator는 OBC의 동일한 네임스페이스에서 OBC의 동일한 이름으로 Secret 및 ConfigMap을 생성합니다.

10.5.5.3. Multicloud Object Gateway CLI를 사용하여 AWS S3 네임 스페이스 버킷 추가

사전 요구 사항

- 실행 중인 OpenShift Container Storage Platform
- Multicloud Object Gateway에 액세스하고 2장, 애플리케이션을 사용하여 Multicloud Object Gateway에액세스
- Multicloud Object Gateway 명령줄 인터페이스를 다운로드합니다.

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

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들어 IBM Z 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

또는 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package 에 있는 OpenShift Container Storage RPM에서 mcg 패키지를 설치할 수도 있습니다.

참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1.

네임스페이스 저장소 리소스를 생성합니다. 네임 스페이스 저장소는 **Multicloud Object Gateway** 네임 스페이스 버킷의 데이터에 대한 읽기 또는 쓰기 대상으로 사용할 기본 스토리지를 나타냅니다. **MCG** 명령줄 인터페이스에서 다음 명령을 실행합니다.

```
noobaa namespacestore create aws-s3 <namespacestore > --access-key <AWS ACCESS KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name> -n openshift-storage
```

a.

<namespacestore> 를 네임 스페이스 저장소의 이름으로 바꿉니다.

b.

<AWS ACCESS KEY> 및 **<AWS SECRET ACCESS KEY>** 를 이 목적을 위해 생성한 **AWS** 액세스 키 **ID** 및 시크릿 액세스 키로 바꿉니다.

c.

<bucket-name> 을 기존 **AWS** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

2.

네임스페이스 버킷에 대한 네임스페이스 정책을 정의하는 네임스페이스 버킷 클래스를 생성합니다. 네임스페이스 정책에는 단일 또는 **multi** 의 유형이 필요합니다.

- 다음 명령을 실행하여 단일 유형의 네임스페이스 정책으로 네임 스페이스 버킷 클래스를 생성합니다.

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

<resource-name> 을 리소스를 제공할 이름으로 바꿉니다.

<my-bucket-class> 를 고유한 버킷 클래스 이름으로 바꿉니다.

<resource> 를 네임 스페이스 버킷의 읽기 및 쓰기 대상을 정의하는 단일 네임 스페이

스 저장소로 바꿉니다.

- 다음 명령을 실행하여 유형 **multi** 의 네임스페이스 정책으로 네임 스페이스 버킷 클래스를 생성합니다.

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

<resource-name> 을 리소스를 제공할 이름으로 바꿉니다.

<my-bucket-class> 를 고유한 버킷 클래스 이름으로 바꿉니다.

<write-resource> 를 네임 스페이스 버킷의 쓰기 대상을 정의할 단일 네임 스페이스 저장소로 바꿉니다.

<read-resources> 를 네임 스페이스 버킷의 읽기 대상을 정의하는 쉼표로 구분된 네임스페이스 저장소 목록으로 바꿉니다.

3. 다음 명령을 실행하여 **2**단계에 정의된 버킷 클래스를 사용하는 **OBC(**개체 버킷 클래스**)** 리소스를 사용하여 버킷을 생성합니다.

```
noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --bucketclass <custom-bucket-class>
```

a. **<bucket-name>** 을 선택한 버킷 이름으로 바꿉니다.

b. **<custom-bucket-class>** 를 **2**단계에서 생성된 버킷 클래스의 이름으로 바꿉니다.

Operator가 **OBC**를 프로비저닝하면 **Multicloud Object Gateway**에서 버킷이 생성되고 **Operator**는 **OBC**의 동일한 네임스페이스에서 **OBC**의 동일한 이름으로 **Secret** 및 **ConfigMap**을 생성합니다.

10.5.5.4. Multicloud Object Gateway CLI를 사용하여 IBM COS 네임 스페이스 버킷 추가

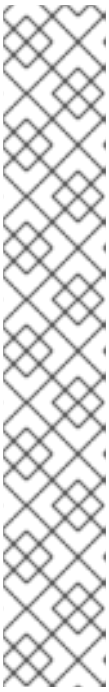
사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform**

- **Multicloud Object Gateway**에 액세스하고 **2**장, 애플리케이션을 사용하여 **Multicloud Object Gateway**에액세스

- **Multicloud Object Gateway** 명령줄 인터페이스를 다운로드합니다.

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



참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들면 다음과 같습니다.

- **IBM Power Systems**의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
```

- **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

또는 **https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package** 에 있는 **OpenShift Container Storage RPM**에서 **mcg** 패키지를 설치할 수도 있습니다.

참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1. 네임스페이스 저장소 리소스를 생성합니다. 네임 스페이스 저장소는 **Multicloud Object Gateway** 네임 스페이스 버킷의 데이터에 대한 읽기 또는 쓰기 대상으로 사용할 기본 스토리지를 나타냅니다. **MCG** 명령줄 인터페이스에서 다음 명령을 실행합니다.

```
noobaa namespacestore create ibm-cos <namespacestore> --endpoint <IBM COS
ENDPOINT> --access-key <IBM ACCESS KEY> --secret-key <IBM SECRET ACCESS
KEY> --target-bucket <bucket-name> -n openshift-storage
```

a. **<namespacestore>** 를 네임 스페이스 저장소의 이름으로 바꿉니다.

b. **<IBM ACCESS KEY>, <IBM SECRET ACCESS KEY>, <IBM COS ENDPOINT>** 를 **IBM** 액세스 키 **ID,** 시크릿 액세스 키 및 기존 **IBM** 버킷 위치에 해당하는 해당 지역 엔드포인트로 바꿉니다.

c. **<bucket-name>** 을 기존 **IBM** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

2. 네임스페이스 버킷에 대한 네임스페이스 정책을 정의하는 네임스페이스 버킷 클래스를 생성합니다. 네임스페이스 정책에는 단일 또는 **multi** 의 유형이 필요합니다.

- 다음 명령을 실행하여 단일 유형의 네임스페이스 정책으로 네임 스페이스 버킷 클래스를 생성합니다.

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

<resource-name> 을 리소스를 제공할 이름으로 바꿉니다.

<my-bucket-class> 를 고유한 버킷 클래스 이름으로 바꿉니다.

<resource> 를 네임 스페이스 버킷의 읽기 및 쓰기 대상을 정의하는 단일 네임 스페이스 저장소로 바꿉니다.

- 다음 명령을 실행하여 유형 **multi** 의 네임스페이스 정책으로 네임 스페이스 버킷 클래스를 생성합니다.

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

<resource-name> 을 리소스를 제공할 이름으로 바꿉니다.

<my-bucket-class> 를 고유한 버킷 클래스 이름으로 바꿉니다.

<write-resource> 를 네임 스페이스 버킷의 쓰기 대상을 정의할 단일 네임 스페이스 저장소로 바꿉니다.

<read-resources> 를 네임 스페이스 버킷의 읽기 대상을 정의하는 쉼표로 구분된 네임 스페이스 저장소 목록으로 바꿉니다.

3. 다음 명령을 실행하여 **2**단계에 정의된 버킷 클래스를 사용하는 **OBC(**개체 버킷 클래스**)** 리소스를 사용하여 버킷을 생성합니다.

```
noobaa obc create my-bucket-claim -n openshift-storage --app-namespace my-app --
bucketclass <custom-bucket-class>
```

a. **<bucket-name>** 을 선택한 버킷 이름으로 바꿉니다.

b. **<custom-bucket-class>** 를 **2**단계에서 생성된 버킷 클래스의 이름으로 바꿉니다.

Operator가 **OBC**를 프로비저닝하면 **Multicloud Object Gateway**에서 버킷이 생성되고 **Operator**는 **OBC**의 동일한 네임스페이스에서 **OBC**의 동일한 이름으로 **Secret** 및 **ConfigMap**을 생성합니다.

10.6. 하이브리드 및 멀티 클라우드 버킷의 데이터 미러링

MCG(Multicloud Object Gateway)는 클라우드 공급자와 클러스터 전체에서 데이터를 포괄하는 프로세스를 간소화합니다.

사전 요구 사항

- 먼저 **MCG**에서 사용할 수 있는 백업 스토리지를 추가해야 하며 **10.4**절. “하이브리드 또는 **Multicloud** 용 스토리지 리소스 추가” 참조하십시오.

그런 다음 데이터 관리 정책 미러링을 반영하는 버킷 클래스를 생성합니다.

절차

미러링 데이터를 설정하는 방법은 다음과 같습니다.

- 10.6.1절. “MCG 명령줄 인터페이스를 사용하여 데이터를 미러링하는 버킷 클래스 생성”
- 10.6.2절. “YAML을 사용하여 데이터를 미러링하도록 버킷 클래스 생성”
- 10.6.3절. “사용자 인터페이스를 사용하여 데이터를 미러링하도록 버킷 구성”

10.6.1. MCG 명령줄 인터페이스를 사용하여 데이터를 미러링하는 버킷 클래스 생성

1. MCG 명령줄 인터페이스에서 다음 명령을 실행하여 미러링 정책으로 버킷 클래스를 생성합니다.

```
$ noobaa bucketclass create placement-bucketclass mirror-to-aws --backingstores=azure-resource,aws-resource --placement Mirror
```

2. 새로 생성된 버킷 클래스를 새 버킷 클레임으로 설정하고 두 위치 간에 미러링될 새 버킷을 생성합니다.

```
$ noobaa obc create  mirrored-bucket --bucketclass=mirror-to-aws
```

10.6.2. YAML을 사용하여 데이터를 미러링하도록 버킷 클래스 생성

1. 다음 YAML을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: BucketClass
metadata:
  labels:
    app: noobaa
  name: <bucket-class-name>
  namespace: openshift-storage
spec:
  placementPolicy:
    tiers:
    - backingStores:
      - <backing-store-1>
      - <backing-store-2>
      placement: Mirror
```

2. 표준 **OBC(**오브젝트 버킷 클레임**)**에 다음 행을 추가합니다.

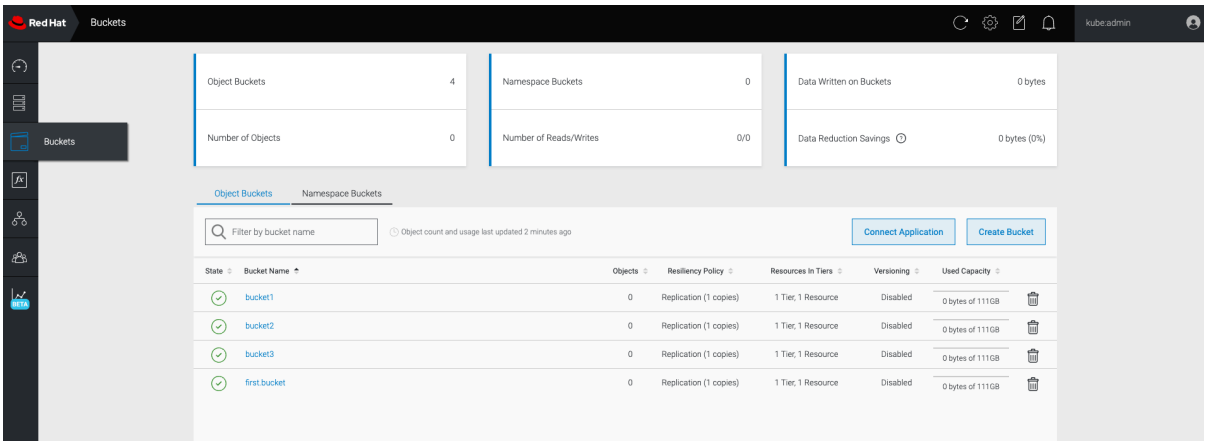
```
additionalConfig:
  bucketclass: mirror-to-aws
```

OBC에 대한 자세한 내용은 **10.8**절**. "**개체 버킷 클레임**"** 의 내용을 참조하십시오.

10.6.3. 사용자 인터페이스를 사용하여 데이터를 미러링하도록 버킷 구성

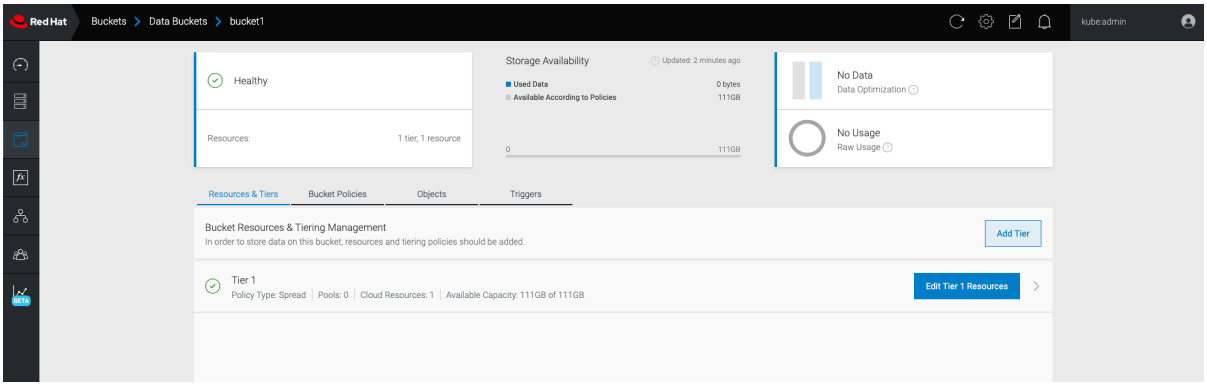
1. **OpenShift Storage** 콘솔에서 스토리지 → 개요 → 오브젝트 탭 → **Multicloud Object Gateway** 링크를 클릭합니다.

2. **NooBaa** 페이지의 왼쪽에 있는 버킷 아이콘을 클릭합니다. 버킷 목록이 표시됩니다.



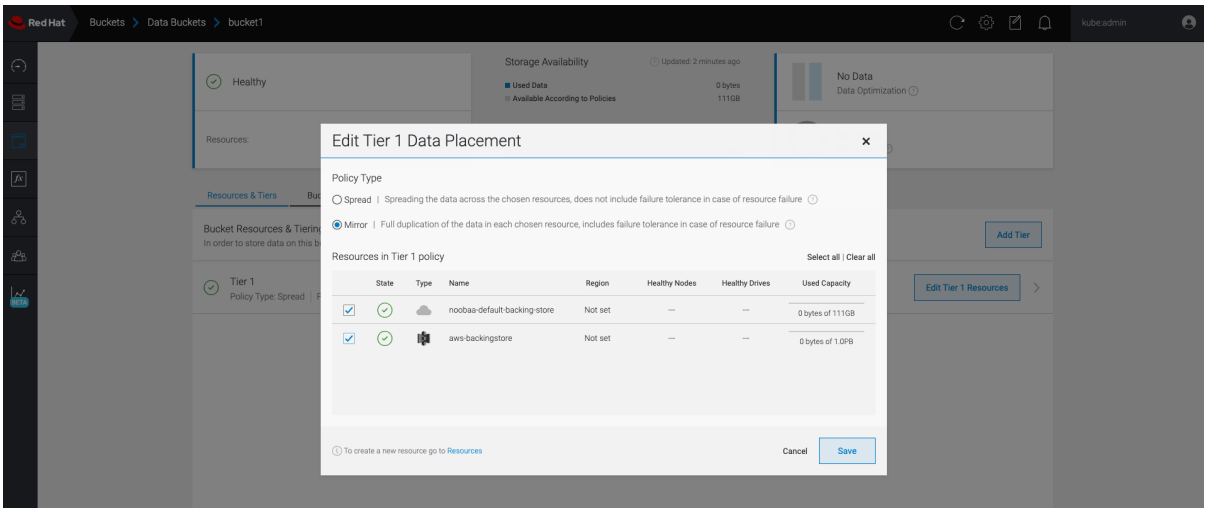
3. 업데이트할 버킷을 클릭합니다.

4. 계층 **1** 리소스 편집을 클릭합니다.



5.

Mirror 를 선택하고 이 버킷에 사용할 관련 리소스를 확인합니다. 다음 예제에서 **AWS**에 있는 **noobaa-default-backing-store** 와 **AWS**에 있는 **AWS-backingstore** 간의 데이터가 미러링됩니다.



6. 저장을 클릭합니다.

참고

NooBaa UI에서 생성된 리소스는 **OpenShift UI** 또는 **MCG CLI**에서 사용할 수 없습니다.

10.7. MULTICLOUD OBJECT GATEWAY의 버킷 정책

OpenShift Container Storage는 **AWS S3** 버킷 정책을 지원합니다. 버킷 정책을 사용하면 버킷 및 해당 객체에 대한 액세스 권한을 사용자에게 부여할 수 있습니다.

10.7.1. 버킷 정책 정보

버킷 정책은 **AWS S3** 버킷 및 오브젝트에 대한 권한을 부여할 수 있는 액세스 정책 옵션입니다. 버킷 정책은 **JSON** 기반 액세스 정책 언어를 사용합니다. 액세스 정책 언어에 대한 자세한 내용은 **AWS Access Policy Language Overview** 를 참조하십시오.

10.7.2. 버킷 정책 사용

사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform**

- **Multicloud Object Gateway**에 대한 액세스, 참조 10.2절. “애플리케이션을 사용하여 Multicloud Object Gateway에 액세스”

절차

Multicloud Object Gateway에서 버킷 정책을 사용하려면 다음을 수행합니다.

1. **JSON** 형식으로 버킷 정책을 생성합니다. 다음 예제를 참조하십시오.

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

액세스 권한과 관련하여 버킷 정책에 사용할 수 있는 많은 요소가 있습니다.

이러한 요소 및 액세스 권한을 제어하는 데 사용할 수 있는 방법에 대한 예제는 AWS Access Policy Language Overview 를 참조하십시오.

버킷 정책의 예를 보려면 AWS 버킷 정책 예제 를 참조하십시오.

S3 사용자를 생성하는 방법은 10.7.3절. “Multicloud Object Gateway에서 AWS S3 사용자 생성” 에서 확인할 수 있습니다.

2. **AWS S3** 클라이언트를 사용하여 **put-bucket-policy** 명령을 사용하여 버킷 정책을 **S3** 버킷에 적용합니다.

```
# aws --endpoint ENDPOINT --no-verify-ssl s3api put-bucket-policy --bucket MyBucket --
policy BucketPolicy
```

S3 엔드 포인트로 교체

MyBucket 을 버킷으로 교체하여 정책을 설정합니다.

BucketPolicy 를 버킷 정책 **JSON** 파일로 교체

자체 서명된 기본 인증서를 사용하는 경우 **--no-verify-ssl** 을 추가합니다.

예를 들면 다음과 같습니다.

```
# 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 명령에 대한 자세한 내용은 **put-bucket-policy** 에 대한 **AWS CLI** 명령 참조를 참조하십시오.

참고

principal 요소는 버킷과 같은 리소스에 대한 액세스가 허용되거나 거부된 사용자를 지정합니다. 현재는 **NooBaa** 계정만 주체로 사용할 수 있습니다. 개체 버킷 클레임의 경우 **NooBaa**는 자동으로 계정 **obc-account.<generated** 버킷 이름**>@noobaa.io** 를 생성합니다.

참고

버킷 정책 조건이 지원되지 않습니다.

10.7.3. Multicloud Object Gateway에서 AWS S3 사용자 생성

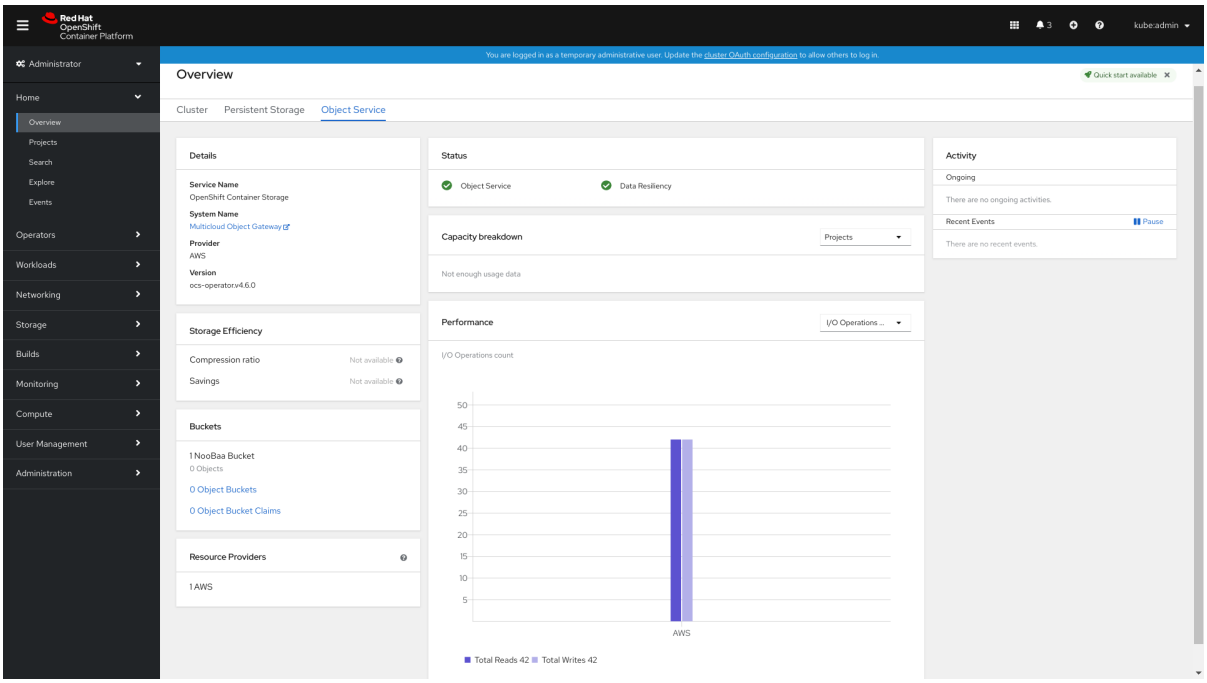
사전 요구 사항

- 실행 중인 **OpenShift Container Storage Platform**

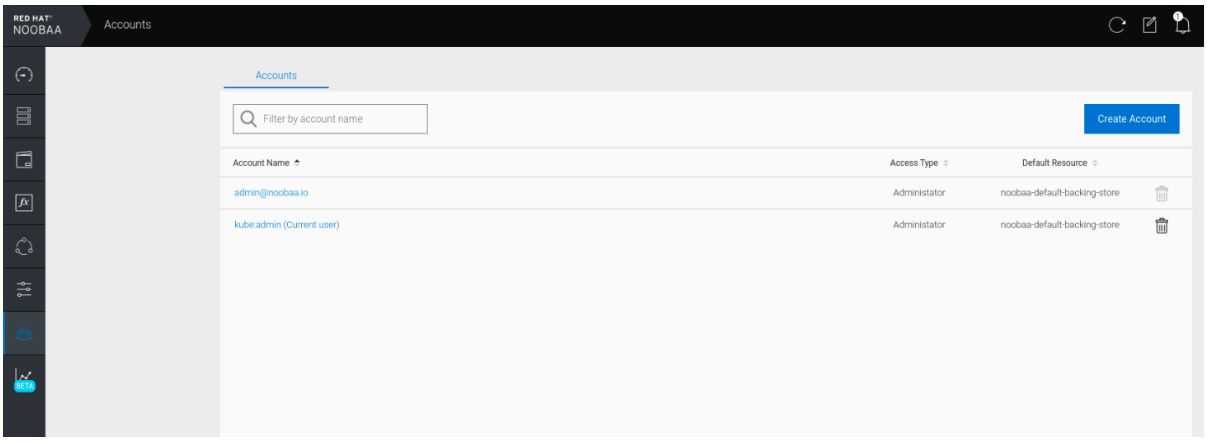
- **Multicloud Object Gateway**에 대한 액세스, 참조 **10.2**절**. "**애플리케이션을 사용하여 **Multicloud Object Gateway**에 액세스**"**

절차

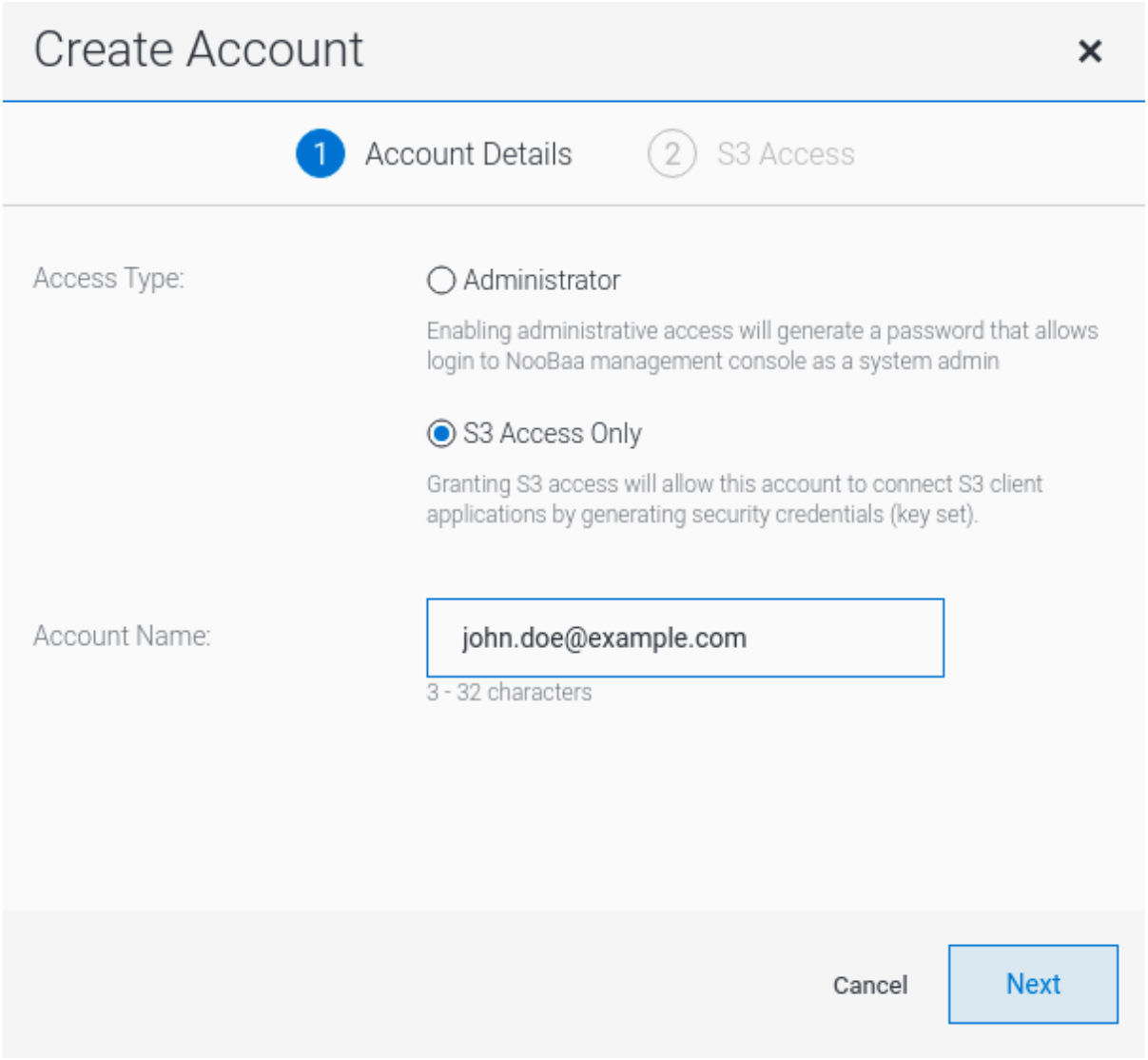
1. **OpenShift Storage** 콘솔에서 스토리지 → 개요 → 오브젝트 탭 → **Multicloud Object Gateway** 링크를 선택합니다.



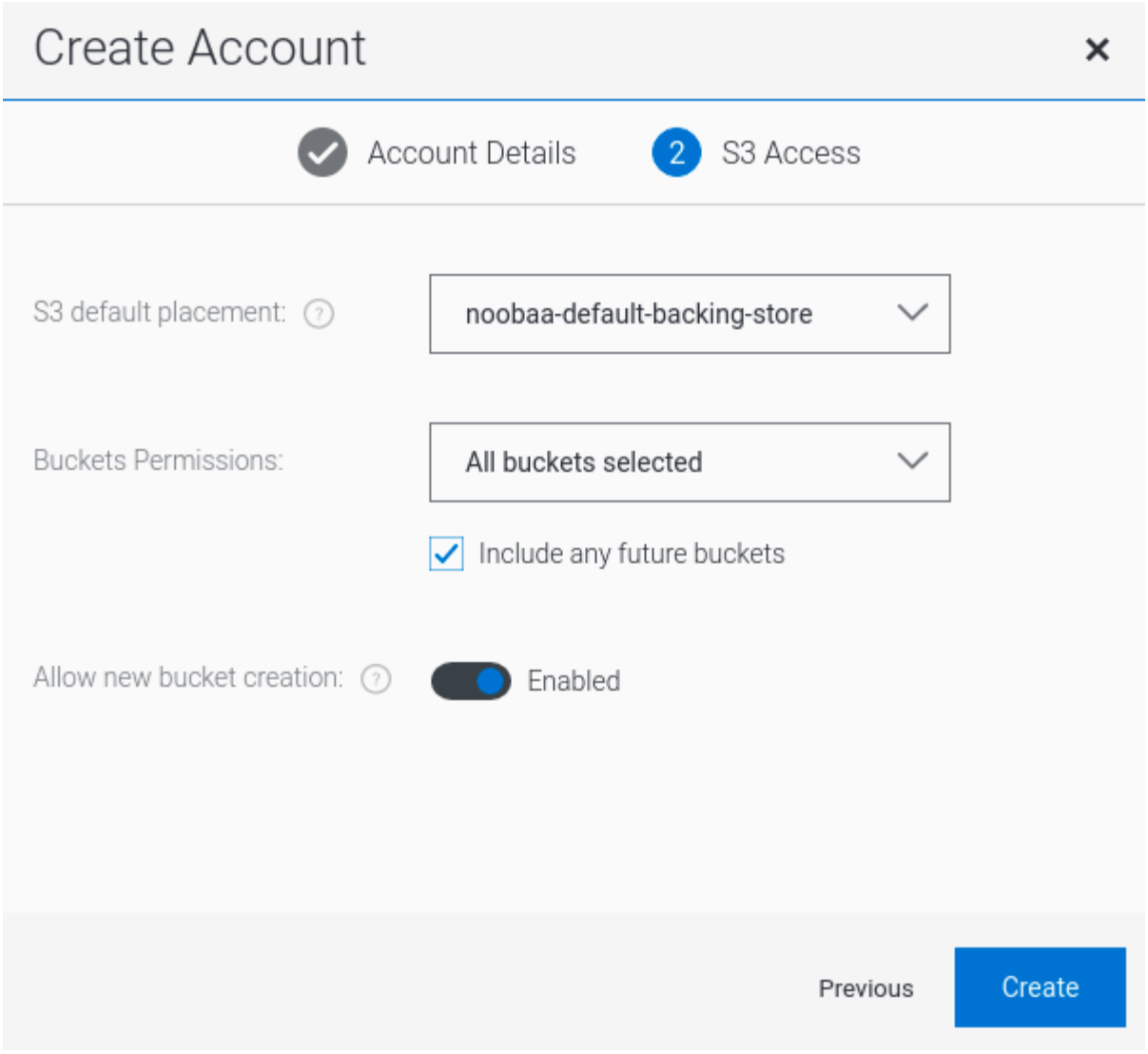
2. **Accounts** (계정) 탭에서 **Create Account**(계정 만들기)를 클릭합니다.



3. **S3** 액세스만 선택하고 계정 이름 (예**: john.doe@example.com**)을 입력합니다 **.** 다음을 클릭합니다.



4. **S3 기본 배치 (예: noobaa-default-backing-store)를 선택합니다. 버킷 권한을 선택합니다. 특정 버킷 또는 모든 버킷을 선택할 수 있습니다. 생성을 클릭합니다.**



10.8. 개체 버킷 클레임

개체 버킷 클레임은 워크로드에 대해 **S3** 호환 버킷 백엔드를 요청하는 데 사용할 수 있습니다.

다음 세 가지 방법으로 오브젝트 버킷 클레임을 생성할 수 있습니다.

- 10.8.1절. “동적 개체 버킷 클레임”
- 10.8.2절. “명령줄 인터페이스를 사용하여 개체 버킷 클레임 생성”
- 10.8.3절. “OpenShift 웹 콘솔을 사용하여 개체 버킷 클레임 생성”

개체 버킷 클레임은 새 액세스 키 및 시크릿 액세스 키를 포함하여 버킷에 대한 권한이 있는 **NooBaa**에

새 버킷 및 애플리케이션 계정을 생성합니다. 애플리케이션 계정은 단일 버킷에만 액세스할 수 있으며 기본적으로 새 버킷을 생성할 수 없습니다.

10.8.1. 동적 개체 버킷 클레임

영구 볼륨과 유사하게 애플리케이션의 YAML에 오브젝트 버킷 클레임의 세부 정보를 추가하고, 오브젝트 서비스 엔드포인트, 액세스 키 및 구성 맵 및 시크릿에서 사용할 수 있는 시크릿 액세스 키를 가져올 수 있습니다. 이 정보를 애플리케이션의 환경 변수로 동적으로 읽기가 쉽습니다.

절차

1. 애플리케이션 YAML에 다음 행을 추가합니다.

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: <obc-name>
spec:
  generateBucketName: <obc-bucket-name>
  storageClassName: openshift-storage.noobaa.io
```

이러한 행은 개체 버킷 클레임 자체입니다.

a. **<obc-name>** 을 고유한 개체 버킷 클레임 이름으로 바꿉니다.

b. **<obc-bucket-name>** 을 개체 버킷 클레임의 고유한 버킷 이름으로 바꿉니다.

2. YAML 파일에 더 많은 줄을 추가하여 개체 버킷 클레임의 사용을 자동화할 수 있습니다. 아래 예제는 데이터를 사용한 구성 맵과 인증 정보가 있는 시크릿인 버킷 클레임 결과 간의 매핑입니다. 이 특정 작업은 NooBaa의 개체 버킷을 클레임하여 버킷과 계정을 만듭니다.

```
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>의 모든 인스턴스를 오브젝트 버킷 클레임 이름으로 바꿉니다.**

b. **<your 애플리케이션 이미지>를 애플리케이션 이미지로 바꿉니다.**

3. **업데이트된 YAML 파일을 적용합니다.**

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

a. **<yaml.file> 을 YAML 파일의 이름으로 바꿉니다.**

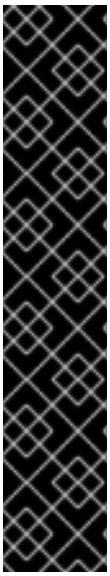
4. **새 구성 맵을 보려면 다음을 실행합니다.**

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

a. **obc-name 을 개체 버킷 클레임의 이름으로 바꿉니다.**

출력에 다음 환경 변수가 있을 수 있습니다.

- **BUCKET_HOST -** 애플리케이션에서 사용할 엔드 포인트
- **BUCKET_PORT -** 애플리케이션에 사용할 수 있는 포트
  - 포트는 **BUCKET_HOST** 와 관련이 있습니다. 예를 들어 **BUCKET_HOST** 가 **https://my.example.com** 이고 **BUCKET_PORT** 가 **443**이면 개체 서비스의 엔드포인트는 **https://my.example.com:443** 입니다.
- **BUCKET_NAME -** 요청되거나 생성된 버킷 이름
- **AWS_ACCESS_KEY_ID -** 인증 정보의 일부인 액세스 키
- **AWS_SECRET_ACCESS_KEY -** 인증 정보의 일부인 비밀 액세스 키



중요

AWS_ACCESS_KEY_ID 및 **AWS_SECRET_ACCESS_KEY** 를 검색합니다. 이름은 **AWS S3 API**와 호환되도록 사용됩니다. 특히 **MCG(Multicloud Object Gateway)** 버킷에서 **S3** 작업을 수행하는 동안 키를 지정해야 합니다. 키는 **Base64**로 인코딩됩니다. 키를 사용하기 전에 디코딩합니다.

```
# oc get secret <obc_name> -o yaml
```

<obc_name>

오브젝트 버킷 클레임의 이름을 지정합니다.

10.8.2. 명령줄 인터페이스를 사용하여 개체 버킷 클레임 생성

명령줄 인터페이스를 사용하여 오브젝트 버킷 클레임을 생성할 때 구성 맵과 시크릿이 함께 제공됩니다. 이 시크릿에는 애플리케이션에서 오브젝트 스토리지 서비스를 사용하는 데 필요한 모든 정보가 포함됩니다.

사전 요구 사항

- **MCG** 명령줄 인터페이스를 다운로드합니다.

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

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들면 다음과 같습니다.

- **IBM Power Systems**의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
```

- **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

절차

1. 명령줄 인터페이스를 사용하여 새 버킷 및 자격 증명의 세부 정보를 생성합니다. 다음 명령을 실행합니다.

```
# noobaa obc create <obc-name> -n openshift-storage
```

<obc-name> 을 고유한 개체 버킷 클레임 이름으로 바꿉니다(예: **myappobc**).

또한 **--app-namespace** 옵션을 사용하여 오브젝트 버킷 클레임 구성 맵과 시크릿이 생성되는 네임스페이스(예: **myapp-namespace**)를 지정할 수 있습니다.

출력 예:

```
INFO[0001]   Created: ObjectBucketClaim "test21obc"
```

MCG 명령줄-인터페이스에서 필요한 구성을 생성하고 **OpenShift**에 새 **OBC**에 대해 알립니다.

2. 다음 명령을 실행하여 개체 버킷 클레임을 확인합니다.

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

출력 예:

```
NAME       STORAGE-CLASS               PHASE   AGE
test21obc  openshift-storage.noobaa.io  Bound   38s
```

3. 다음 명령을 실행하여 새 오브젝트 버킷 클레임에 대한 **YAML** 파일을 확인합니다.

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

출력 예:

```
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** 네임스페이스 내에서 이 오브젝트 버킷 클레임을 사용하는 구성 맵과 시크릿을 찾을 수 있습니다. **CM** 및 시크릿의 이름은 개체 버킷 클레임과 동일합니다. 시크릿을 보려

면 다음을 수행합니다.

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

출력 예:

```
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
```

시크릿은 **S3** 액세스 자격 증명을 제공합니다.

5. 구성 맵을 보려면 다음을 수행합니다.

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

출력 예:

```
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
```

구성 맵에는 애플리케이션의 **S3** 끝점 정보가 포함되어 있습니다.

10.8.3. OpenShift 웹 콘솔을 사용하여 개체 버킷 클레임 생성

OpenShift 웹 콘솔을 사용하여 **OBC(**오브젝트 버킷 클레임**)**를 생성할 수 있습니다.

사전 요구 사항

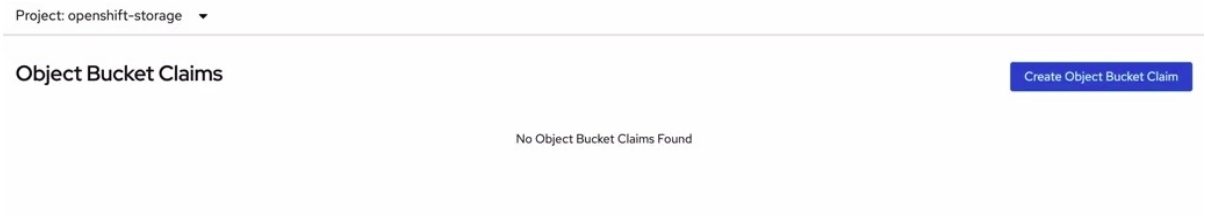
- **OpenShift** 웹 콘솔에 대한 관리 액세스.
- 애플리케이션이 **OBC**와 통신하려면 **configmap** 및 **secret**을 사용해야 합니다. 이에 대한 자세한 내용은 **10.8.1**절. “동적 개체 버킷 클레임” 의 내용을 참조하십시오.

절차

1. **OpenShift** 웹 콘솔에 로그인합니다.

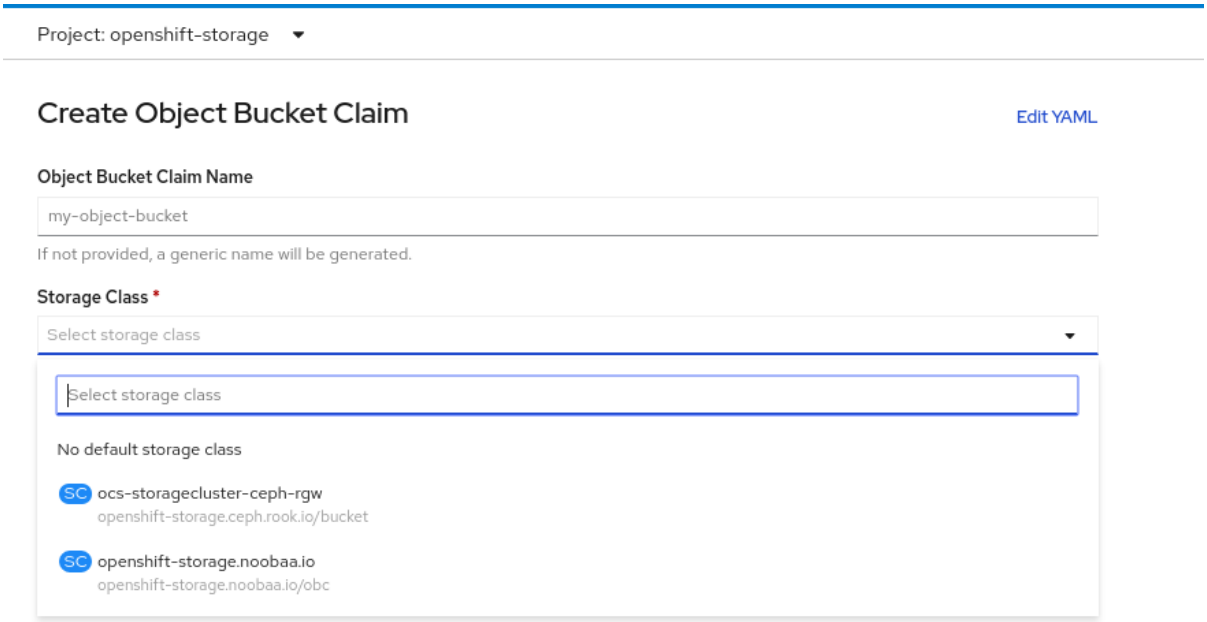
2. 왼쪽 네비게이션 바에서 스토리지 → 개체 버킷 클레임 을 클릭합니다.

3. 개체 버킷 클레임 생성을 클릭합니다.



4. 오브젝트 버킷 클레임의 이름을 입력하고 드롭다운 메뉴에서 배포(내부 또는 외부)를 기반으로 적절한 스토리지 클래스를 선택합니다.

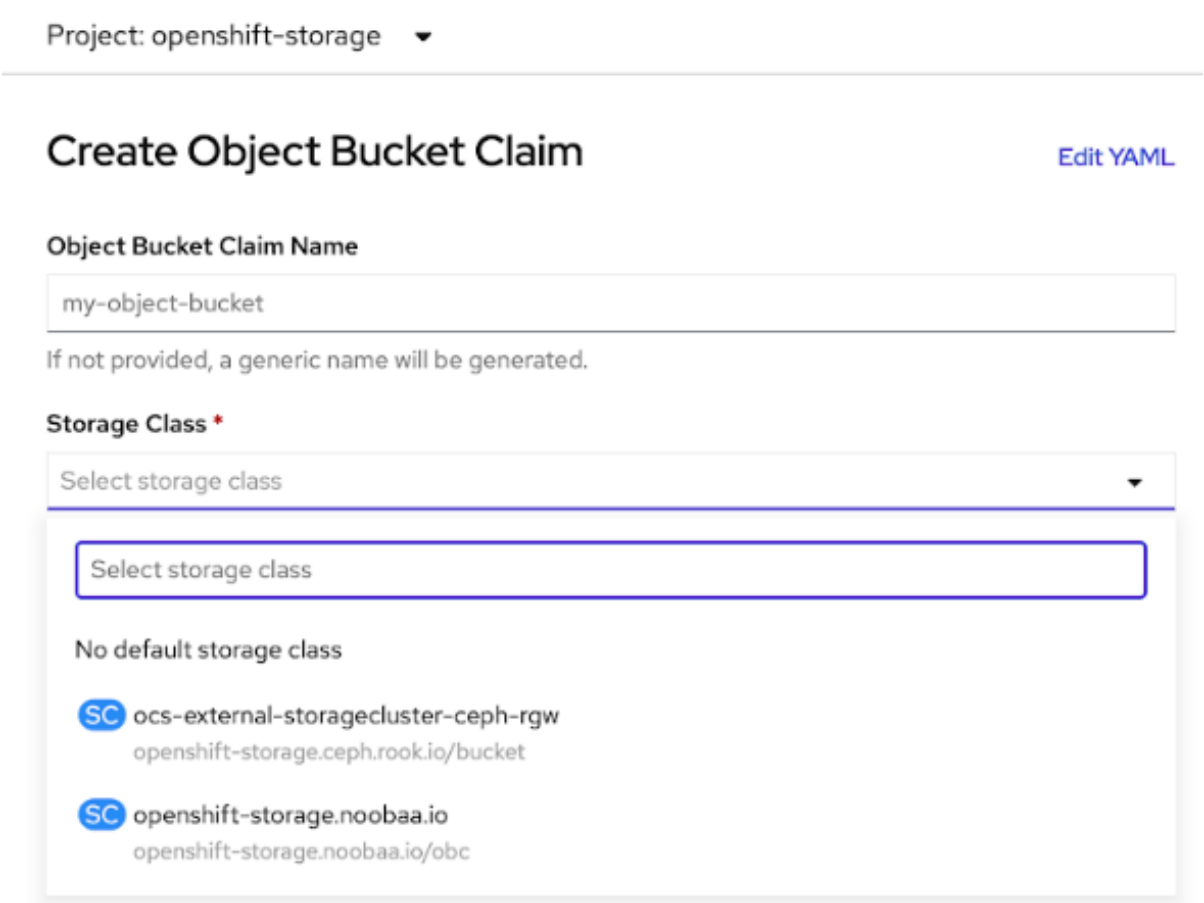
내부 모드



배포 후에 생성된 다음 스토리지 클래스를 사용할 수 있습니다.

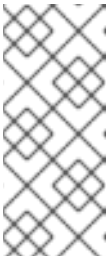
- **oCS-storagecluster-ceph-rgw** 는 **Ceph Object Gateway(RGW)**를 사용합니다.
- **openshift-storage.noobaa.io** 는 **Multicloud Object Gateway**를 사용합니다.

외부 모드



배포 후에 생성된 다음 스토리지 클래스를 사용할 수 있습니다.

- **OCS-external-storagecluster-ceph-rgw** 는 **Ceph RGW(**오브젝트 게이트웨이**)**를 사용합니다.
- **openshift-storage.noobaa.io** 는 **Multicloud Object Gateway**를 사용합니다.

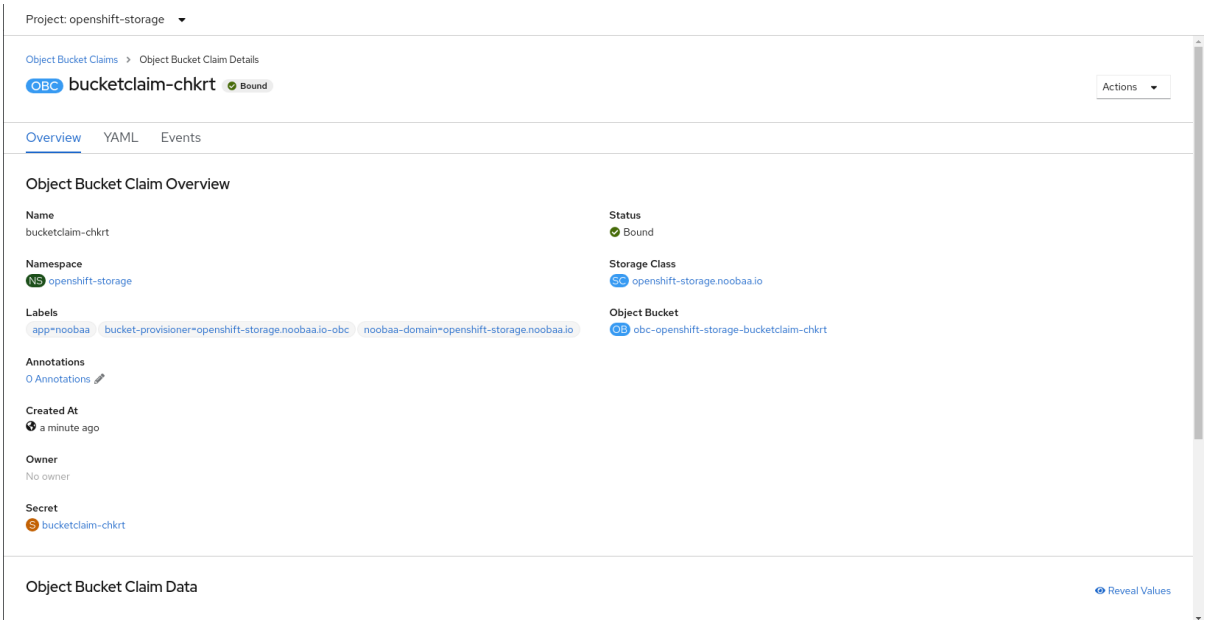


참고

RGW OBC 스토리지 클래스는 **OpenShift Container Storage** 버전 **4.5**의 새로운 설치에서만 사용할 수 있습니다. 이전 **OpenShift Container Storage** 릴리스에서 업그레이드된 클러스터에는 적용되지 않습니다.

5. 생성을 클릭합니다.

OBC를 생성하면 세부 정보 페이지로 리디렉션됩니다.



추가 리소스

- 10.8절. “개체 버킷 클레임”

10.8.4. 배포에 오브젝트 버킷 클레임 연결

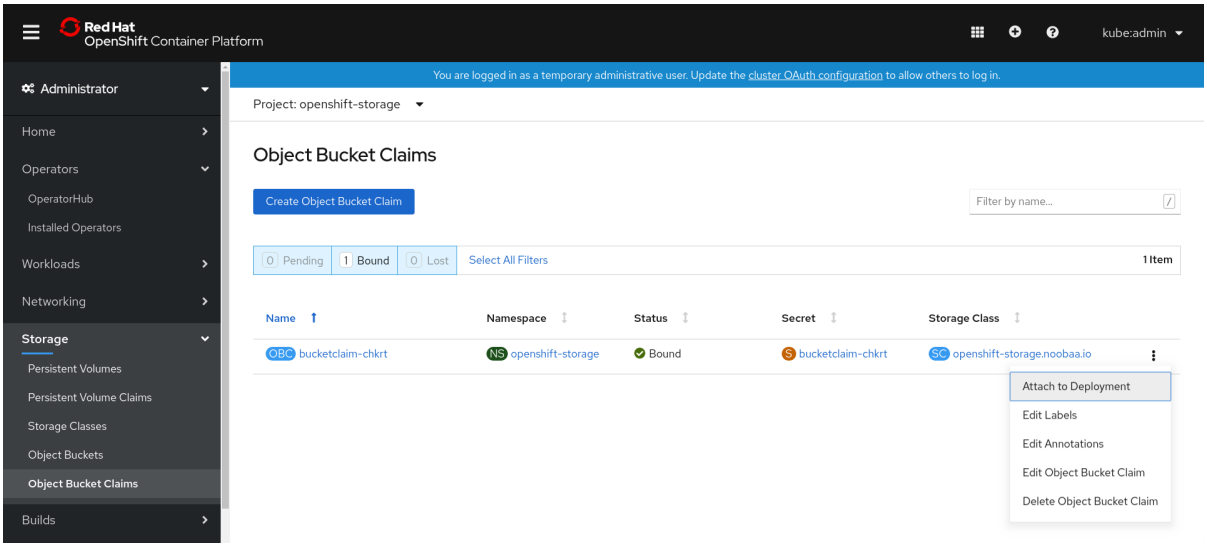
생성되고 나면 **OBC(**오브젝트 버킷 클레임**)**를 특정 배포에 연결할 수 있습니다.

사전 요구 사항

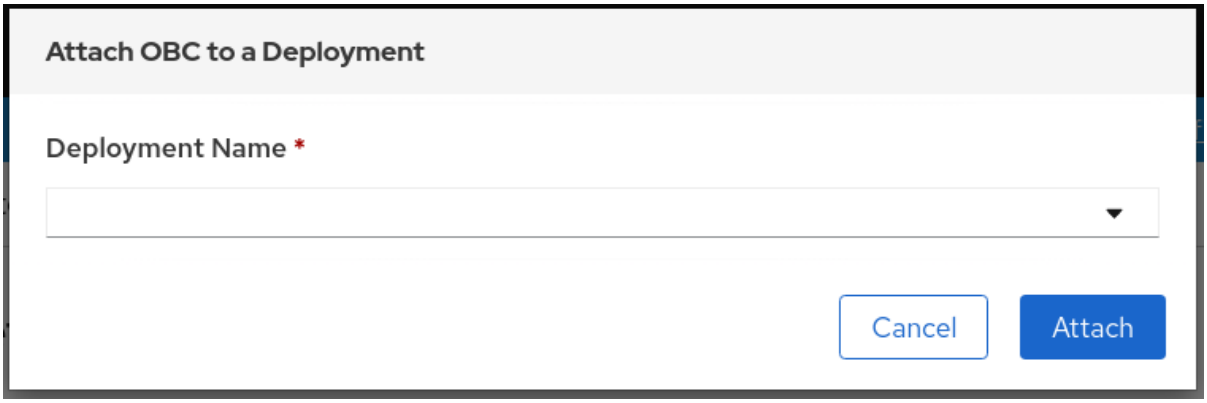
- **OpenShift** 웹 콘솔에 대한 관리 액세스.

절차

1. 왼쪽 네비게이션 바에서 스토리지 → 개체 버킷 클레임 을 클릭합니다.
2. 생성한 **OBC** 옆에 있는 작업 메뉴**(kube)**를 클릭합니다.
3. 드롭다운 메뉴에서 **Attach to Deployment** (배포에 연결)를 선택합니다.



4. **Deployment Name(배포 이름) 목록에서 원하는 배포를 선택한 다음 Attach:**



추가 리소스

- 10.8절. “개체 버킷 클레임”

10.8.5. OpenShift 웹 콘솔을 사용하여 오브젝트 버킷 보기

OpenShift 웹 콘솔을 사용하여 OBC(오브젝트 버킷 클레임)에 대해 생성된 개체 버킷의 세부 정보를 볼 수 있습니다.

사전 요구 사항

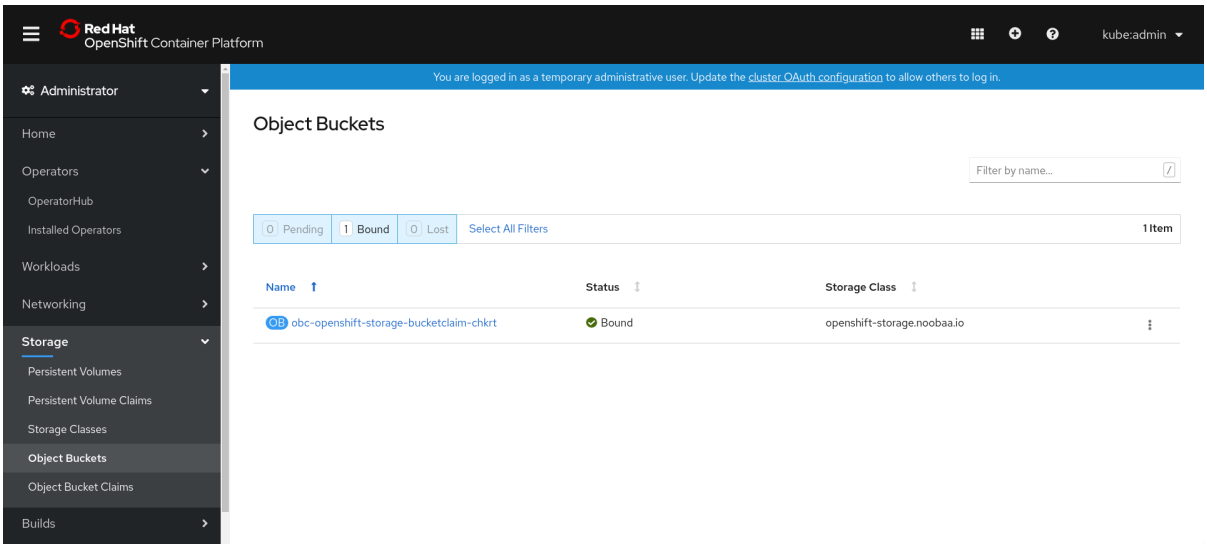
- **OpenShift 웹 콘솔에 대한 관리 액세스.**

절차

오브젝트 버킷 세부 정보를 보려면 다음을 수행합니다.

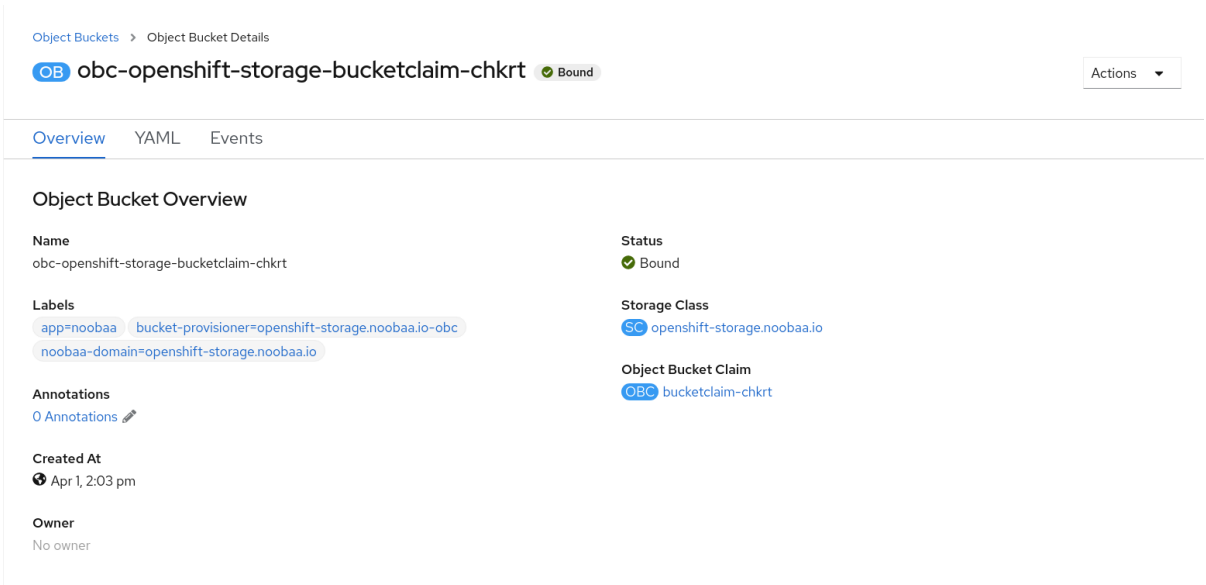
1. **OpenShift** 웹 콘솔에 로그인합니다.

2. 왼쪽 네비게이션 바에서 스토리지 → 개체 버킷을 클릭합니다.



특정 **OBC**의 세부 정보 페이지로 이동하고 리소스 링크를 클릭하여 해당 **OBC**의 개체 버킷을 볼 수도 있습니다.

3. 세부 정보를 볼 개체 버킷을 선택합니다. 오브젝트 버킷의 세부 정보 페이지로 이동합니다.



추가 리소스

- **10.8**절. “개체 버킷 클레임”

10.8.6. 개체 버킷 클레임 삭제

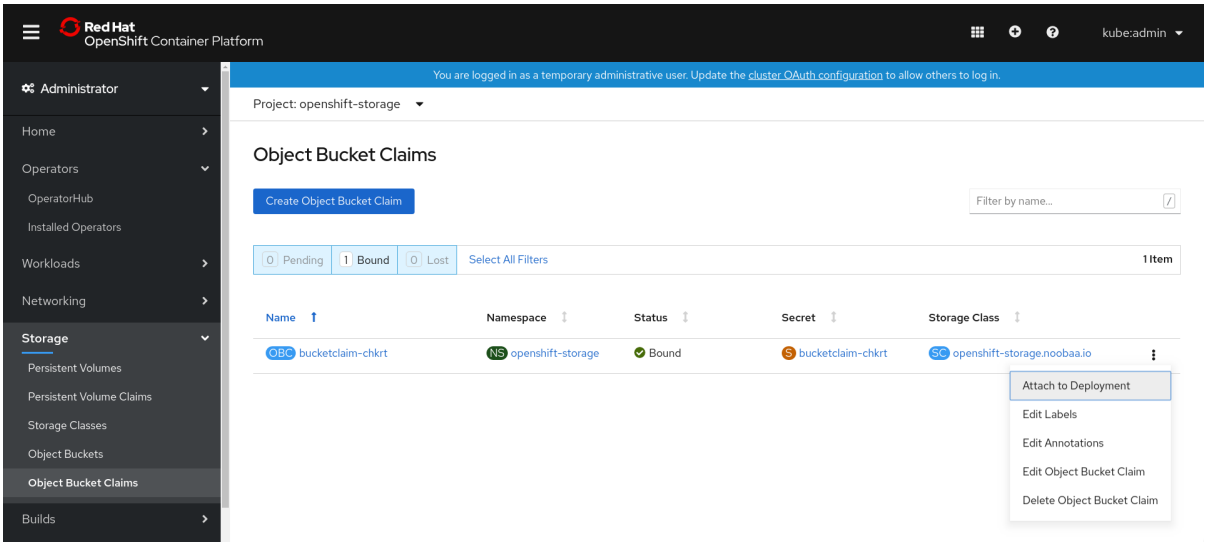
사전 요구 사항

- OpenShift 웹 콘솔에 대한 관리 액세스.

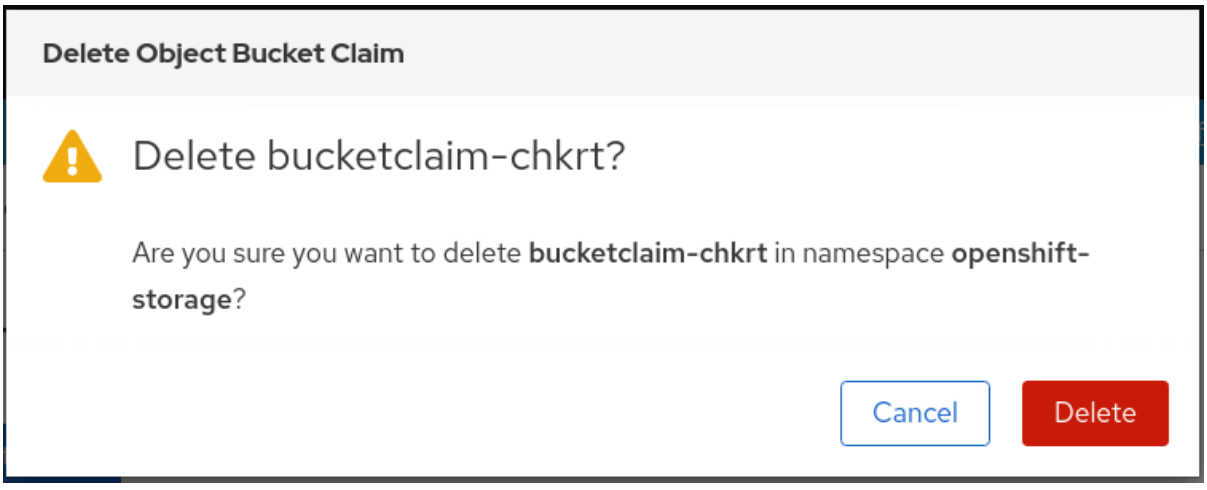
절차

1. 왼쪽 네비게이션 바에서 스토리지 → 개체 버킷 클레임 을 클릭합니다.

2. 삭제할 개체 버킷 클레임 옆에 있는 작업 메뉴(kube)를 클릭합니다.



3. 메뉴에서 개체 버킷 클레임 삭제 를 선택합니다.



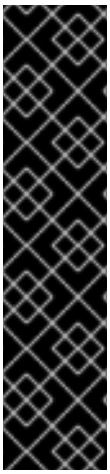
4. 삭제를 클릭합니다.

추가 리소스

- 10.8절. “개체 버킷 클레임”

10.9. 오브젝트 버킷의 캐싱 정책

캐시 버킷은 **hub** 대상 및 캐시 대상이 있는 네임스페이스 버킷입니다. **hub** 대상은 **S3** 호환 가능한 대형 오브젝트 스토리지 버킷입니다. 캐시 버킷은 로컬 **Multicloud Object Gateway** 버킷입니다. **AWS** 버킷 또는 **IBM COS** 버킷을 캐시하는 캐시 버킷을 생성할 수 있습니다.



중요

캐시 버킷은 기술 프리뷰 기능입니다. 기술 프리뷰 기능은 **Red Hat** 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않으며 기능적으로 완전하지 않을 수 있습니다. 따라서 프로덕션 환경에서 사용하는 것은 권장하지 않습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

자세한 내용은 기술 프리뷰 기능 지원 범위를 참조하십시오.

- **AWS S3**
- **IBM COS**

10.9.1. AWS 캐시 버킷 생성

사전 요구 사항

- **MCG(Multicloud Object Gateway)** 명령줄 인터페이스를 다운로드합니다.

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

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들어 **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

또는 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package 에 있는 OpenShift Container Storage RPM에서 mcg 패키지를 설치할 수도 있습니다.

참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1. 네임스페이스 저장소 리소스를 생성합니다. 네임 스페이스 저장소는 Multicloud Object Gateway 네임 스페이스 버킷의 데이터에 대한 읽기 또는 쓰기 대상으로 사용할 기본 스토리지를 나타냅니다. MCG 명령줄 인터페이스에서 다음 명령을 실행합니다.

```
noobaa namespacestore create aws-s3 <namespacestore> --access-key <AWS ACCESS KEY> --secret-key <AWS SECRET ACCESS KEY> --target-bucket <bucket-name>
```

a. <namespacestore> 를 네임스페이스 저장소의 이름으로 바꿉니다.

b. <AWS ACCESS KEY> 및 <AWS SECRET ACCESS KEY> 를 이 목적을 위해 생성한 AWS 액세스 키 ID 및 시크릿 액세스 키로 바꿉니다.

c. <bucket-name> 을 기존 AWS 버킷 이름으로 바꿉니다. 이 인수는 Multicloud Object Gateway에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

YAML을 적용하여 스토리지 리소스를 추가할 수도 있습니다. 먼저 인증 정보를 사용하여 보안을 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
type: Opaque
data:
  AWS_ACCESS_KEY_ID: <AWS ACCESS KEY ID ENCODED IN BASE64>
  AWS_SECRET_ACCESS_KEY: <AWS SECRET ACCESS KEY ENCODED IN BASE64>
```

Base64를 사용하여 자체 AWS 액세스 키 ID 및 시크릿 액세스 키를 제공하고 인코딩해야 하며 <AWS ACCESS KEY ID ENCODED in BASE64> 및 < AWS SECRET ACCESS

KEY ENCODED in BASE64> 대신 결과를 사용해야 합니다.

<namespacestore-secret-name> 을 고유한 이름으로 바꿉니다.

그런 다음 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <namespacestore>
  namespace: openshift-storage
spec:
  awsS3:
    secret:
      name: <namespacestore-secret-name>
      namespace: <namespace-secret>
    targetBucket: <target-bucket>
  type: aws-s3
```

d. **<namespacestore>** 를 고유한 이름으로 바꿉니다.

e. **<namespacestore-secret-name>** 을 이전 단계에서 생성한 보안으로 바꿉니다.

f. **<namespace-secret>** 을 이전 단계에서 보안을 생성하는 데 사용된 네임스페이스로 바꿉니다.

g. **<target-bucket>** 을 네임스페이스 저장소용으로 생성한 **AWS S3** 버킷으로 바꿉니다.

2. 다음 명령을 실행하여 버킷 클래스를 생성합니다.

```
noobaa bucketclass create namespace-bucketclass cache <my-cache-bucket-class> --
backingstores <backing-store> --hub-resource <namespacestore>
```

a. **<my-cache-bucket-class>** 를 고유한 버킷 클래스 이름으로 바꿉니다.

b. **<backing-store>** 를 관련 백업 저장소로 바꿉니다. 이 필드에 쉼표로 구분된 하나 이상의 백업 저장소를 나열할 수 있습니다.

c. **<namespacestore>** 를 이전 단계에서 생성한 네임스페이스 저장소로 바꿉니다.

3. 다음 명령을 실행하여 **2**단계에 정의된 버킷 클래스를 사용하는 개체 버킷 클레임 리소스를 사용하여 버킷을 생성합니다.

```
noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
```

a. **<my-bucket-claim>** 을 고유한 이름으로 바꿉니다.

b. **<custom-bucket-class>** 를 **2**단계에서 생성된 버킷 클래스의 이름으로 바꿉니다.

10.9.2. IBM COS 캐시 버킷 생성

사전 요구 사항

- **MCG(Multicloud Object Gateway)** 명령줄 인터페이스를 다운로드합니다.

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

참고

서브스크립션 관리자를 사용하여 리포지토리를 활성화하기 위한 적절한 아키텍처를 지정합니다. 예를 들면 다음과 같습니다.

- **IBM Power Systems**의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-ppc64le-rpms
```

- **IBM Z** 인프라의 경우 다음 명령을 사용합니다.

```
# subscription-manager repos --enable=rh-ocs-4-for-rhel-8-s390x-rpms
```

또는 https://access.redhat.com/downloads/content/547/ver=4/rhel---8/4/x86_64/package 에 있는 **OpenShift Container Storage RPM**에서 **mcg** 패키지를 설치할 수도 있습니다.

참고

아키텍처에 따라 올바른 제품 변형을 선택합니다.

절차

1. 네임스페이스 저장소 리소스를 생성합니다. 네임 스페이스 저장소는 **Multicloud Object Gateway** 네임 스페이스 버킷의 데이터에 대한 읽기 또는 쓰기 대상으로 사용할 기본 스토리지를 나타냅니다. **MCG** 명령줄 인터페이스에서 다음 명령을 실행합니다.

```
noobaa namespacestore create ibm-cos <namespacestore> --endpoint <IBM COS ENDPOINT> --access-key <IBM ACCESS KEY> --secret-key <IBM SECRET ACCESS KEY> --target-bucket <bucket-name>
```

a. **<namespacestore>** 를 네임 스페이스 저장소의 이름으로 바꿉니다.

b. **<IBM ACCESS KEY>, <IBM SECRET ACCESS KEY>, <IBM COS ENDPOINT>** 를 **IBM** 액세스 키 **ID,** 시크릿 액세스 키 및 기존 **IBM** 버킷 위치에 해당하는 해당 지역 엔드포인트로 바꿉니다.

c. **<bucket-name>** 을 기존 **IBM** 버킷 이름으로 바꿉니다. 이 인수는 **Multicloud Object Gateway**에 백업 저장소에 대상 버킷으로 사용할 버킷과 그 다음에는 데이터 스토리지 및 관리에 지시합니다.

YAML을 적용하여 스토리지 리소스를 추가할 수도 있습니다. 먼저 인증 정보를 사용하여 보안을 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <namespacestore-secret-name>
type: Opaque
data:
  IBM_COS_ACCESS_KEY_ID: <IBM COS ACCESS KEY ID ENCODED IN BASE64>
  IBM_COS_SECRET_ACCESS_KEY: <IBM COS SECRET ACCESS KEY ENCODED IN BASE64>
```

Base64를 사용하여 자체 **IBM COS** 액세스 키 **ID** 및 비밀 액세스 키를 제공하고 인코딩해야 하며 **<IBM COS COS** 키 **ID ENCODED IN BASE64>** 및 **< IBM COS SECRET ACCESS KEY ENCODED IN BASE64>'** 대신 결과를 사용해야 합니다.

<namespacestore-secret-name> 을 고유한 이름으로 바꿉니다.

그런 다음 다음 **YAML**을 적용합니다.

```
apiVersion: noobaa.io/v1alpha1
kind: NamespaceStore
metadata:
  finalizers:
  - noobaa.io/finalizer
  labels:
    app: noobaa
  name: <namespacestore>
  namespace: openshift-storage
spec:
  s3Compatible:
    endpoint: <IBM COS ENDPOINT>
    secret:
      name: <backingstore-secret-name>
      namespace: <namespace-secret>
    signatureVersion: v2
    targetBucket: <target-bucket>
  type: ibm-cos
```

d. **<namespacestore>** 를 고유한 이름으로 바꿉니다.

e. **<IBM COS ENDPOINT>** 를 적절한 **IBM COS** 엔드포인트로 바꿉니다.

f. **<backingstore-secret-name>** 을 이전 단계에서 생성한 보안으로 바꿉니다.

g. **<namespace-secret>** 을 이전 단계에서 보안을 생성하는 데 사용된 네임스페이스로 바꿉니다.

h. **<target-bucket>** 을 네임스페이스 저장소용으로 생성한 **AWS S3** 버킷으로 바꿉니다.

2. 다음 명령을 실행하여 버킷 클래스를 생성합니다.

```
noobaa bucketclass create namespace-bucketclass cache <my-bucket-class> --
backingstores <backing-store> --hubResource <namespacestore>
```

a. **<my-bucket-class>** 를 고유한 버킷 클래스 이름으로 바꿉니다.

b. **<backing-store>** 를 관련 백업 저장소로 바꿉니다. 이 필드에 쉼표로 구분된 하나 이상의 백업 저장소를 나열할 수 있습니다.

c. **<namespacestore>** 를 이전 단계에서 생성한 네임스페이스 저장소로 바꿉니다.

3. 다음 명령을 실행하여 **2**단계에 정의된 버킷 클래스를 사용하는 개체 버킷 클레임 리소스를 사용하여 버킷을 생성합니다.

```
noobaa obc create <my-bucket-claim> my-app --bucketclass <custom-bucket-class>
```

a. **<my-bucket-claim>** 을 고유한 이름으로 바꿉니다.

b. **<custom-bucket-class>** 를 **2**단계에서 생성된 버킷 클래스의 이름으로 바꿉니다.

10.10. 끝점을 추가하여 MULTICLOUD OBJECT GATEWAY 성능 확장

Multicloud Object Gateway 성능은 환경마다 다를 수 있습니다. 경우에 따라 특정 애플리케이션에는 **S3** 엔드포인트를 확장하여 쉽게 처리할 수 있는 더 빠른 성능이 필요한 경우도 있습니다.

Multicloud Object Gateway 리소스 풀은 기본적으로 두 가지 유형의 서비스를 제공하는 **NooBaa** 데몬 컨테이너 그룹입니다.

- 스토리지 서비스
- **S3** 끝점 서비스

10.10.1. Multicloud Object Gateway의 S3 끝점

S3 엔드포인트는 모든 **Multicloud Object Gateway**가 **Multicloud Object Gateway**에서 데이터 다이

제스트를 처리하는 데 기본적으로 제공하는 서비스입니다. 엔드포인트 서비스는 인라인 데이터 청크, 중복 제거, 압축 및 암호화를 처리하고 **Multicloud Object Gateway**의 데이터 배치 지침을 허용합니다.

10.10.2. 스토리지 노드를 사용한 확장

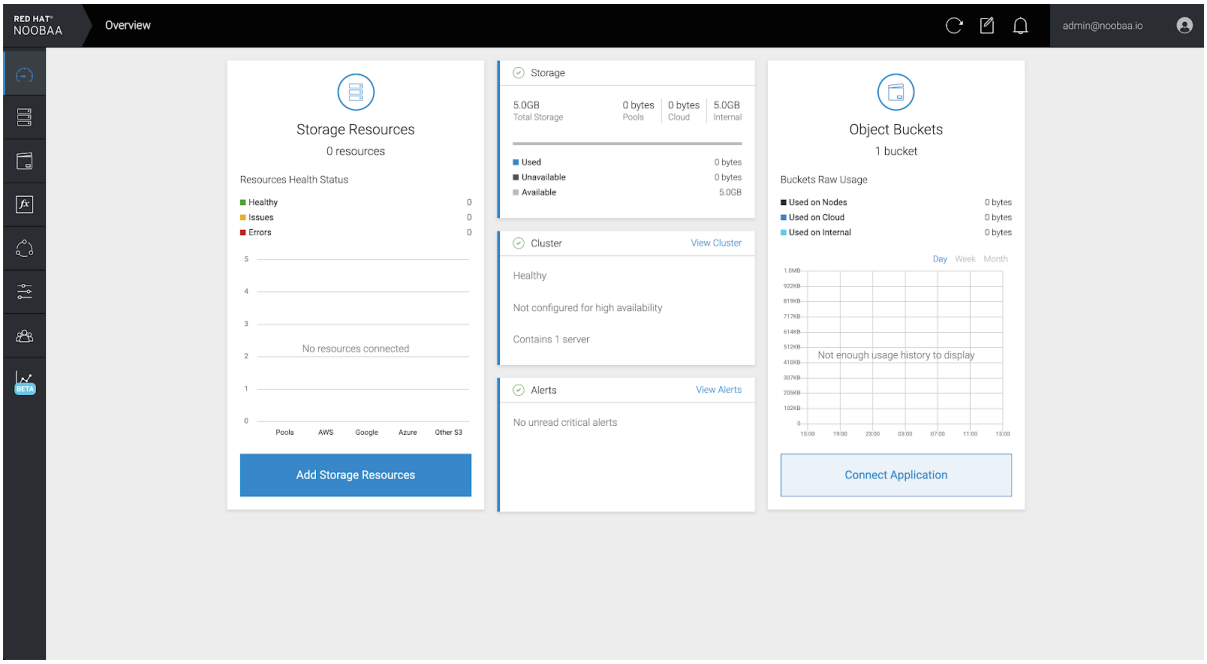
사전 요구 사항

- **Multicloud Object Gateway**에 액세스할 수 있는 **OpenShift Container Platform**에서 실행 중인 **OpenShift Container Storage** 클러스터입니다.

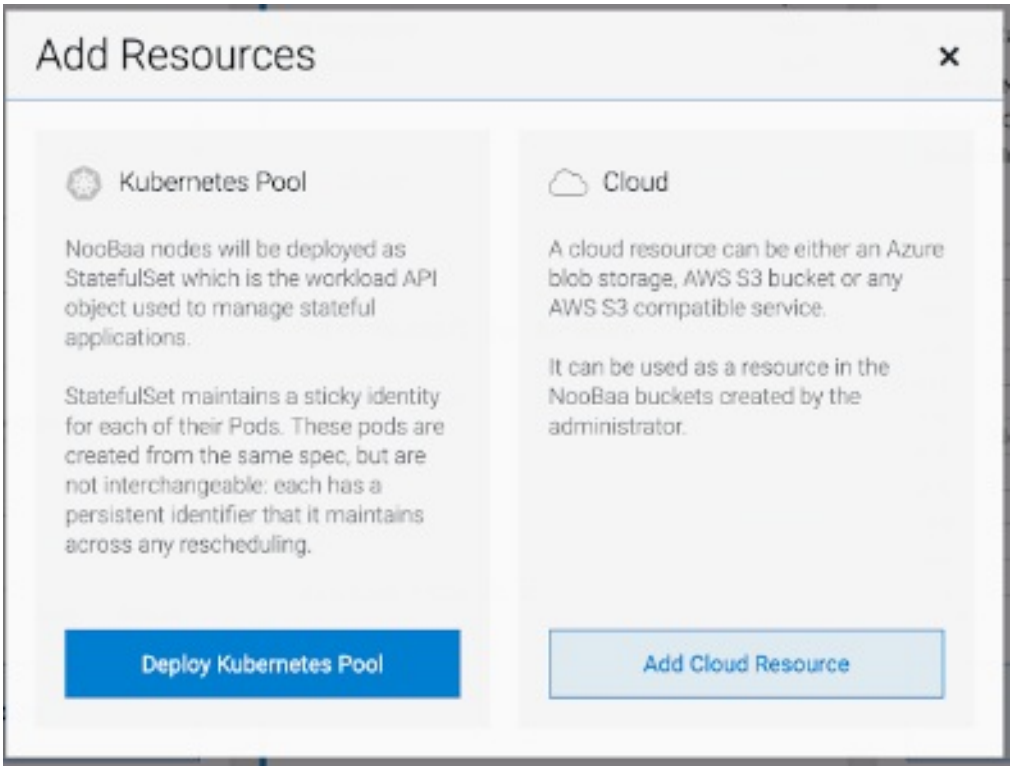
Multicloud Object Gateway의 스토리지 노드는 하나 이상의 영구 볼륨에 연결되어 로컬 오브젝트 서비스 데이터 스토리지에 사용되는 **NooBaa** 데몬 컨테이너입니다. **NooBaa** 데몬은 **Kubernetes** 노드에 배포할 수 있습니다. 이 작업은 **StatefulSet Pod**로 구성된 **Kubernetes** 풀을 생성하여 수행할 수 있습니다.

절차

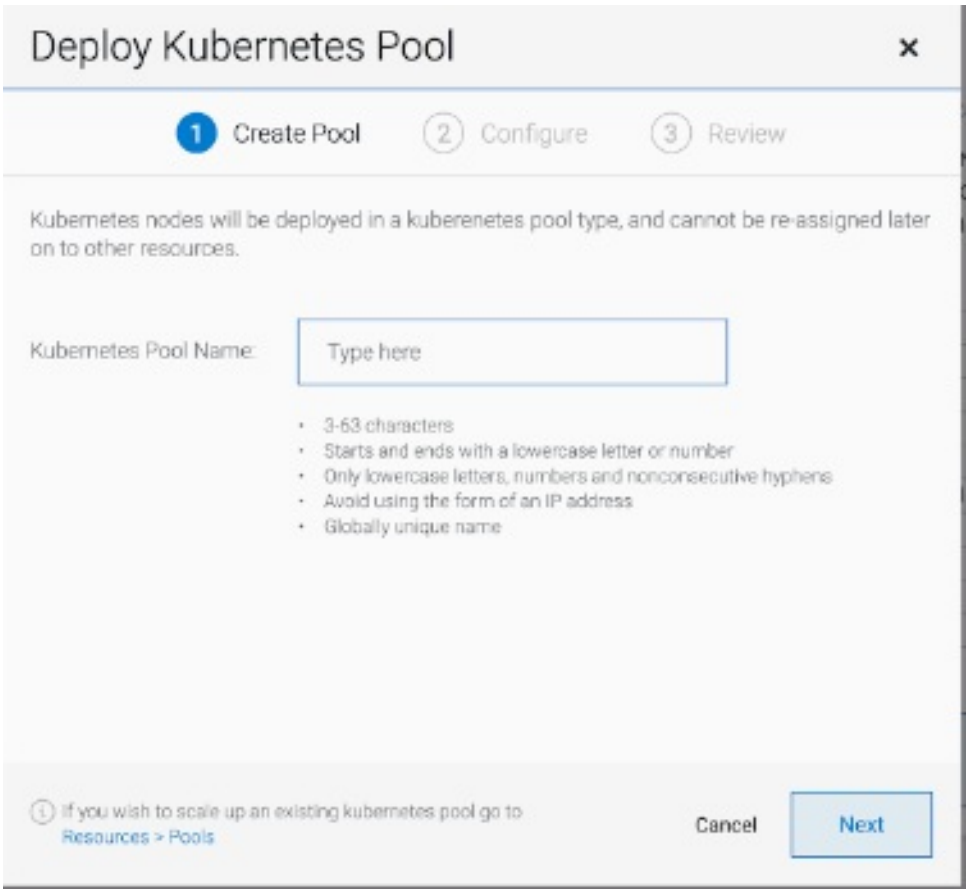
1. **Multicloud Object Gateway** 사용자 인터페이스의 **Overview** (개요) 페이지에서 **Add Storage Resources**(스토리지 리소스 추가)를 클릭합니다.



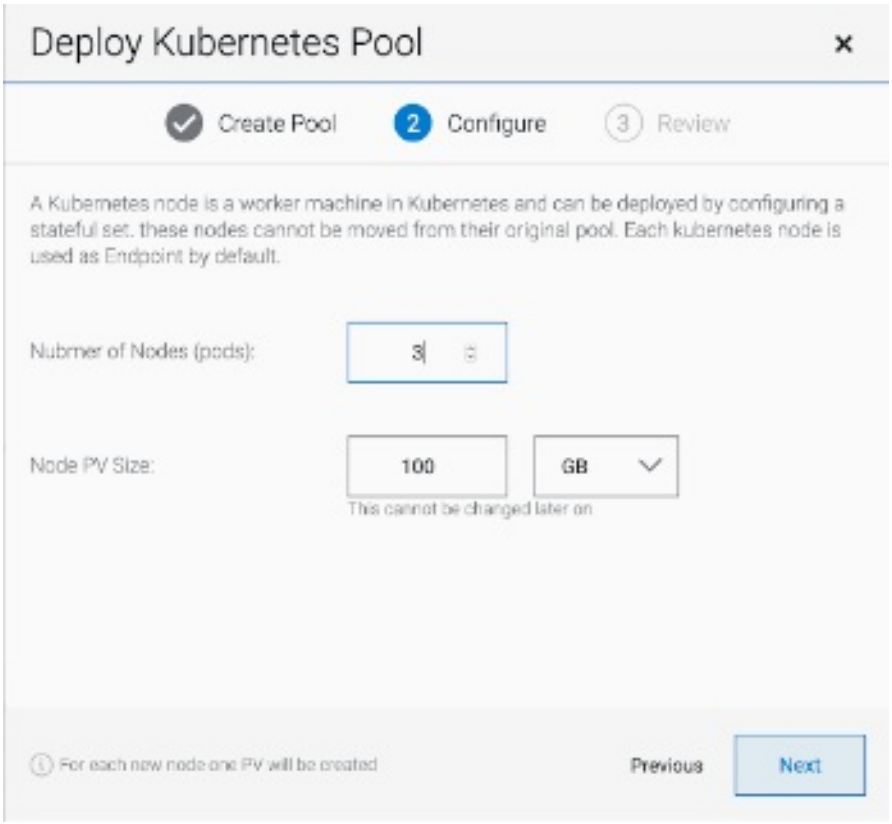
2. 창에서 **Deploy Kubernetes Pool** (**Kubernetes** 풀 배포)을 클릭합니다.



3. **Create Pool(풀 만들기) 단계에서 향후 설치된 노드의 대상 풀을 생성합니다.**

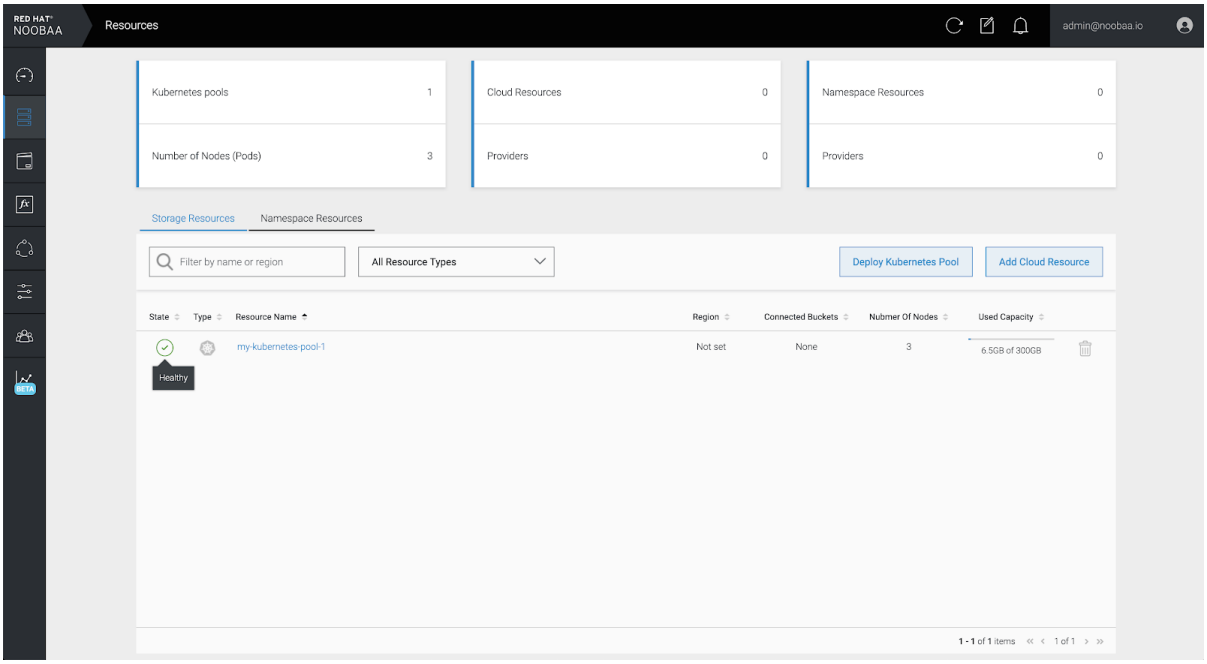


4. **구성 단계에서 요청된 Pod 수 및 각 PV 크기를 구성합니다. 새 포드마다 하나의 PV가 생성됩니다.**



5. 검토 단계에서 새 풀의 세부 정보를 찾고 사용할 배포 방법을 선택할 수 있습니다(로컬 또는 외부 배포). 로컬 배포를 선택하면 **Kubernetes** 노드가 클러스터 내에 배포됩니다. 외부 배포를 선택하면 외부에서 실행할 **YAML** 파일이 제공됩니다.

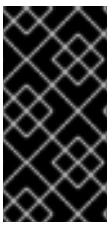
6. 모든 노드는 첫 번째 단계에서 선택한 풀에 할당되며 리소스 → 스토리지 리소스 → 리소스 이름:



10.11. MULTICLOUD OBJECT GATEWAY 엔드 포인트 자동 확장

MCG S3 서비스의 부하가 증가하거나 감소하면 **MCG** 개체 게이트웨이(**MCG**) 엔드포인트 수가 자동으로 확장됩니다. **{product-name-short}** 클러스터는 하나의 활성 **MCG** 끝점과 함께 배포됩니다. 각 **MCG** 끝점 **Pod**는 기본적으로 요청과 일치하는 제한이 있는 **1**개의 **CPU** 및 **2Gi** 메모리 요청으로 구성됩니다. 엔드포인트의 **CPU** 로드가 일관된 기간 동안 **80%**의 사용 임계값을 초과하면 첫 번째 엔드포인트의 부하를 줄이는 두 번째 엔드포인트가 배포됩니다. 두 끝점의 평균 **CPU** 로드가 일관된 기간 동안 **80%** 임계값 미만으로 떨어지면 엔드포인트 중 하나가 삭제됩니다. 이 기능을 사용하면 **MCG**의 성능 및 서비스 가능성이 향상됩니다.

11장. 영구 볼륨 클레임 관리



중요

OpenShift Container Storage에서 지원하는 PVC에는 PVC 확장이 지원되지 않습니다.

11.1. OPENSHIFT CONTAINER STORAGE를 사용하도록 애플리케이션 POD 구성

이 섹션의 지침에 따라 OpenShift Container Storage를 애플리케이션 포드의 스토리지로 구성합니다.

사전 요구 사항

- OpenShift 웹 콘솔에 대한 관리자 액세스 권한이 있습니다.
- OpenShift Container Storage Operator는 openshift-storage 네임스페이스에 설치되고 실행됩니다. OpenShift 웹 콘솔에서 Operator → 설치된 Operator 를 클릭하여 설치된 Operator 를 확인합니다.
- OpenShift Container Storage에서 제공하는 기본 스토리지 클래스는 사용할 수 있습니다. OpenShift 웹 콘솔에서 스토리지 → 스토리지 클래스를 클릭하여 기본 스토리지 클래스를 확인합니다.

절차

1. 애플리케이션에서 사용할 영구 볼륨 클레임(PVC)을 생성합니다.
   a. OpenShift 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭합니다.
   b. 애플리케이션 포드의 프로젝트를 설정합니다.
   c. 영구 볼륨 클레임 생성을 클릭합니다.
      i. OpenShift Container Storage에서 제공하는 스토리지 클래스를 지정합니다.

ii. **PVC 이름을 지정합니다(예: myclaim).**

iii. **필요한 액세스 모드를 선택합니다.**

iv. **Rados Block Device(RBD)의 경우 액세스 모드가 RWO(ReadWriteOnce)인 경우 필요한 볼륨 모드를 선택합니다. 기본 볼륨 모드는 Filesystem 입니다.**

v. **애플리케이션 요구 사항에 따라 크기를 지정합니다.**

vi. **생성을 클릭하고 PVC가 Bound 상태가 될 때까지 기다립니다.**

2. **새 PVC를 사용하도록 새 또는 기존 애플리케이션 포드를 구성합니다.**

- **새 애플리케이션 포드의 경우 다음 단계를 수행합니다.**

i. **워크로드 → 포드를클릭합니다.**

ii. **새 애플리케이션 포드 생성.**

iii. **spec: 섹션에서 volume: 섹션을 추가하여 애플리케이션 포드의 볼륨으로 새 PVC를 추가합니다.**

```
volumes:
  - name: <volume_name>
    persistentVolumeClaim:
      claimName: <pvc_name>
```

예를 들면 다음과 같습니다.

```
volumes:
  - name: mypd
    persistentVolumeClaim:
      claimName: myclaim
```

- 기존 애플리케이션 **Pod**의 경우 다음 단계를 수행합니다.

  i. 워크로드 →배포 구성을 클릭합니다.

  ii. 애플리케이션 포드와 연결된 필수 배포 구성을 검색합니다.

  iii. 작업 메뉴 **(TI)** → 배포 구성 편집을 클릭합니다.

  iv. **spec:** 섹션에서 **volume:** 섹션에서 새 **PVC**를 애플리케이션 포드의 볼륨으로 추가하고 **Save(**저장 **)**를 클릭합니다.

```
volumes:
  - name: <volume_name>
    persistentVolumeClaim:
      claimName: <pvc_name>
```

예를 들면 다음과 같습니다.

```
volumes:
  - name: mypd
    persistentVolumeClaim:
      claimName: myclaim
```

3. 새 구성이 사용 중인지 확인합니다.

   a. 워크로드 → 포드를 클릭합니다.

   b. 애플리케이션 포드의 프로젝트를 설정합니다.

   c. 애플리케이션 포드가 **Running** (실행 중) 상태로 표시되는지 확인합니다.

   d. 애플리케이션 포드 이름을 클릭하여 포드 세부 정보를 확인합니다.

   e.

Volumes(볼륨) 섹션까지 아래로 스크롤하고 볼륨에 새 영구 볼륨 클레임과 일치하는 유형이 있는지 확인합니다(예: myclaim).

11.2. 영구 볼륨 클레임 요청 상태 보기

PVC 요청 상태를 보려면 다음 절차를 사용하십시오.

사전 요구 사항

- OpenShift Container Storage에 대한 관리자 액세스.

절차

1. OpenShift 웹 콘솔에 로그인합니다.
2. 스토리지 → 영구 볼륨 클레임을클릭합니다.
3. 필터 텍스트 상자를 사용하여 필요한 PVC 이름을 검색합니다. PVC 목록을 이름 또는 레이블로 필터링하여 목록을 좁힐 수도 있습니다.
4. 필요한 PVC에 해당하는 Status(상태) 열을 확인합니다.
5. 필요한 Name (이름)을 클릭하여 PVC 세부 정보를 확인합니다.

11.3. 영구 볼륨 클레임 요청 이벤트 검토

PVC(영구 볼륨 클레임) 요청 이벤트를 검토하고 해결하려면 이 절차를 사용하십시오.

사전 요구 사항

- OpenShift 웹 콘솔에 대한 관리자 액세스.

절차

1. OpenShift 웹 콘솔에 로그인합니다.
2. 스토리지 → 개요 → 블록 및 파일을 클릭합니다.
3. 인벤토리 카드를 찾아 오류가 있는 PVC 수를 확인합니다.
4. 스토리지 → 영구 볼륨 클레임을클릭합니다.
5. 필터 텍스트 상자를 사용하여 필요한 PVC를 검색합니다.
6. PVC 이름을 클릭하고 Events로 이동합니다.
7. 필요에 따라 또는 지시된 대로 이벤트를 처리합니다.

11.4. 동적 프로비저닝

11.4.1. 동적 프로비저닝 소개

StorageClass 리소스 객체는 요청 가능한 스토리지를 설명하고 분류할 뿐 만 아니라 필요에 따라 동적으로 프로비저닝된 스토리지에 대한 매개 변수를 전달하는 수단을 제공합니다. StorageClass 객체는 다른 수준의 스토리지 및 스토리지에 대한 액세스를 제어하기 위한 관리 메커니즘으로 사용될 수 있습니다. 클러스터 관리자(cluster-admin) 또는스토리지 관리자(스토리지 관리자)는 사용자가 기본 스토리지 볼륨 소스에 대한 지식이 없어도 요청할 수 있는 StorageClass 오브젝트를 정의하고 생성합니다.

OpenShift Container Platform 영구 볼륨 프레임 워크를 사용하면이 기능을 사용할 수 있으며 관리자는 영구 스토리지로 클러스터를 프로비저닝할 수 있습니다. 또한 이 프레임 워크를 통해 사용자는 기본 인프라에 대한 지식이 없어도 해당 리소스를 요청할 수 있습니다.

OpenShift Container Platform에서 많은 스토리지 유형을 영구 볼륨으로 사용할 수 있습니다. 관리자가 이를 모두 정적으로 프로비저닝할 수 있지만 일부 스토리지 유형은 기본 제공자 및 플러그인 API를 사용하여 동적으로 만들 수 있습니다.

11.4.2. OpenShift Container Storage의 동적 프로비저닝

Red Hat OpenShift Container Storage는 컨테이너 환경에 최적화된 소프트웨어 정의 스토리지입니다. OpenShift Container Platform에서 운영자로 실행되어 컨테이너에 대해 고도로 통합되고 단순화된 영구 스토리지 관리를 제공합니다.

OpenShift Container Storage는 다음을 비롯한 다양한 스토리지 유형을 지원합니다.

- 데이터베이스용 블록 스토리지
- 지속적인 통합, 메시징 및 데이터 집계를 위한 공유 파일 스토리지
- 아카이브, 백업 및 미디어 스토리지를 위한 오브젝트 스토리지

버전 4에서는 Red Hat Ceph Storage를 사용하여 영구 볼륨을 지원하는 파일, 블록 및 오브젝트 스토리지를 제공하고 Rook.io를 사용하여 영구 볼륨 및 클레임 프로비저닝을 관리하고 오케스트레이션합니다. NooBaa는 개체 스토리지를 제공하고, Multicloud Gateway를 사용하면 여러 클라우드 환경에서 개체 페더레이션(기술 프리뷰로 제공)이 가능합니다.

OpenShift Container Storage 4에서 RBD(RADOS Block Device) 및 Ceph File System(CephFS)용 Red Hat Ceph Storage CSI(Container Storage Interface) 드라이버는 동적 프로비저닝 요청을 처리합니다. PVC 요청이 동적으로 제공되면 CSI 드라이버에는 다음과 같은 옵션이 있습니다.

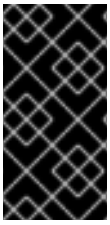
- 볼륨 모드 블록이있는 Ceph RBD를 기반으로 RWO(ReadWriteOnce) 및 RWX(ReadWriteMany) 액세스 권한이 있는 PVC를 만듭니다.
- 볼륨 모드 Filesystem이 있는 Ceph RBD를 기반으로 하는 RWO(ReadWriteOnce) 액세스 권한이 있는 PVC를 만듭니다.
- 볼륨 모드 Filesystem의 CephFS를 기반으로 하는 RWO(ReadWriteOnce) 및 RWX(ReadWriteMany) 액세스 권한으로 PVC를 생성합니다.

사용할 드라이버(RBD 또는 CephFS)는 storageclass.yaml 파일의 항목을 기반으로 합니다.

11.4.3. 사용 가능한 동적 프로비저닝 플러그인

OpenShift Container Platform은 다음과 같은 프로비저너 플러그인을 제공합니다. 이에는 클러스터의 구성된 제공자 API를 사용하여 새 스토리지 리소스의 동적 프로비저닝을 위한 일반 구현이 포함되어 있습니다.

스토리지 유형	프로비저너 플러그인 이름	참고
OpenStack Cinder	**kubernetes.io/cinder**	
AWS Elastic Block Store (EBS)	**kubernetes.io/aws-ebs**	다른 영역에서 여러 클러스터를 사용할 때 동적 프로비저닝의 경우 각 노드에 **Key=kubernetes.io/cluster/<cluster_name>,Value=<cluster_id>**로 태그를 지정합니다. 여기서 **<cluster_name>** 및 **<cluster_id>**는 클러스터마다 고유합니다.
AWS Elastic File System (EFS)		동적 프로비저닝은 프로비저너 플러그인이 아닌 EFS 프로비저너 Pod를 통해 수행됩니다.
Azure Disk	**kubernetes.io/azure-disk**	
Azure File	**kubernetes.io/azure-file**	**persistent-volume-binder** ServiceAccount에는 Azure 저장소 계정 및 키를 저장할 Secrets를 생성하고 검색할 수 있는 권한이 필요합니다.
GCE Persistent Disk (gcePD)	**kubernetes.io/gce-pd**	멀티 존 설정에서는 현재 클러스터에 노드가 없는 영역에서 PV가 생성되지 않도록 GCE 프로젝트 당 하나의 OpenShift Container Platform 클러스터를 실행하는 것이 좋습니다.
VMware vSphere	**kubernetes.io/vsphere-volume**	
Red Hat Virtualization	**csi.ovirt.org**	



중요

선택한 프로비저너 플러그인에는 관련 문서에 따라 클라우드, 호스트 또는 타사 공급자를 구성해야 합니다.

12장. 볼륨 스냅샷

볼륨 스냅샷은 특정 시점에서 클러스터의 스토리지 볼륨 상태입니다. 이러한 스냅샷은 매번 전체 복사를 하지 않아도 되므로 스토리지를 보다 효율적으로 사용할 수 있으며 애플리케이션 개발을 위한 구성 요소로 사용할 수 있습니다.

동일한 PVC(영구 볼륨 클레임)의 스냅샷을 여러 개 생성할 수 있습니다. CephFS의 PVC당 최대 100개의 스냅샷을 생성할 수 있습니다. RADOS 블록 장치(RBD)의 경우 PVC당 최대 512개의 스냅샷을 생성할 수 있습니다.

참고

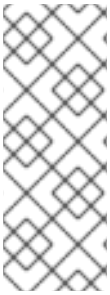
주기적으로 스냅샷 생성을 예약할 수 없습니다.

12.1. 볼륨 스냅샷 생성

PVC(Persistent Volume Claim) 페이지 또는 Volume Snapshots 페이지에서 볼륨 스냅샷을 생성할 수 있습니다.

사전 요구 사항

- 일관된 스냅샷의 경우 PVC는 Bound 상태여야 하며 사용하지 않아야 합니다. 스냅샷을 생성하기 전에 모든 IO를 중지해야 합니다.



참고

OpenShift Container Storage는 Pod가 이를 사용하는 경우에만 PVC의 볼륨 스냅샷에 대한 충돌 일관성을 제공합니다. 애플리케이션 일관성의 경우 일관된 스냅샷을 보장하거나 애플리케이션에서 제공하는 빠른 메커니즘을 사용하여 이를 보장하기 위해 실행 중인 포드를 먼저 제거해야 합니다.

절차

영구 볼륨 클레임 페이지에서

1. OpenShift 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭합니다.
2. 볼륨 스냅샷을 생성하려면 다음 중 하나를 수행하십시오.

- 원하는 **PVC** 옆의 작업 메뉴 **(PS) → Create Snapshot(**스냅샷 만들기**)** 을 클릭합니다.

- 스냅샷을 생성할 **PVC**를 클릭하고 작업 → 스냅샷 생성을 클릭합니다.

3. 볼륨 스냅샷의 **Name** 을 입력합니다.

4. 드롭다운 목록에서 **Snapshot Class(**스냅샷 클래스 **)**를 선택합니다.

5. 생성을 클릭합니다. 생성된 볼륨 스냅샷의 세부 정보 페이지로 리디렉션됩니다.

Volume Snapshots(볼륨 스냅샷**)** 페이지에서

1. **OpenShift** 웹 콘솔에서 스토리지 → 볼륨 스냅숏 을 클릭합니다.

2. **Volume Snapshots(**볼륨 스냅샷**)** 페이지에서 **Create Volume Snapshot (**볼륨 스냅샷 만들기**)**을 클릭합니다.

3. 드롭다운 목록에서 필요한 프로젝트를 선택합니다.

4. 드롭다운 목록에서 영구 볼륨 클레임 을 선택합니다.

5. 스냅샷의 **Name (**이름**)**을 입력합니다.

6. 드롭다운 목록에서 **Snapshot Class(**스냅샷 클래스 **)**를 선택합니다.

7. 생성을 클릭합니다. 생성된 볼륨 스냅샷의 세부 정보 페이지로 리디렉션됩니다.

검증 단계

- **PVC**의 세부 정보 페이지로 이동하고 **Volume Snapshots(**볼륨 스냅샷 **)** 탭을 클릭하여 볼륨

스냅샷 목록을 확인합니다. 새 볼륨 스냅샷이 나열되는지 확인합니다.

- OpenShift 웹 콘솔에서 스토리지 → 볼륨 스냅숏 을 클릭합니다. 새 볼륨 스냅샷이 나열되는지 확인합니다.
- 볼륨 스냅샷이 Ready 상태가 될 때까지 기다립니다.

12.2. 볼륨 스냅샷 복원

볼륨 스냅샷을 복원하면 새 PVC(영구 볼륨 클레임)가 생성됩니다. 복원된 PVC는 볼륨 스냅샷 및 상위 PVC와 독립적입니다.

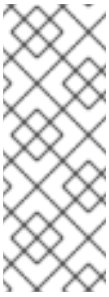
영구 볼륨 클레임 페이지 또는 볼륨 스냅샷 페이지에서 볼륨 스냅샷을 복원할 수 있습니다.

절차

영구 볼륨 클레임 페이지에서

상위 PVC가 있는 경우에만 영구 볼륨 클레임 페이지에서 볼륨 스냅샷을 복원할 수 있습니다.

1. OpenShift 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭합니다.
2. 볼륨 스냅샷이 있는 PVC 이름을 클릭하여 볼륨 스냅샷을 새 PVC로 복원합니다.
3. Volume Snapshots(볼륨 스냅샷) 탭에서 복원할 볼륨 스냅샷 옆에 있는 Action(작업 메뉴)을 클릭합니다.
4. Restore(복구)를 새 PVC로 클릭합니다.
5. 새 PVC의 이름을 입력합니다.
6. 스토리지 클래스 이름을 선택합니다.



참고

RBD(Rados Block Device)의 경우 상위 PVC와 동일한 풀을 사용하여 스토리지 클래스를 선택해야 합니다. 암호화가 활성화되지 않고 그 반대로 지원되지 않는 스토리지 클래스를 사용하여 암호화된 PVC의 스냅샷을 복원합니다.

7. 선택한 액세스 모드를 선택합니다.

중요

RX(ReadOnlyMany) 액세스 모드는 개발자 프리뷰 기능이며 개발자 프리뷰 지원 제한 사항이 있습니다. 개발자 미리보기 릴리스는 프로덕션 환경에서 실행되도록 설계되지 않으며 Red Hat 고객 포털 케이스 관리 시스템을 통해 지원되지 않습니다. ReadOnlyMany 기능에 대한 지원이 필요한 경우 ocs-devpreview@redhat.com 메일링 리스트와 Red Hat 개발 팀 구성원에게 연락하면 가용성 및 업무 일정에 따라 최대한 빨리 도움을 드릴 것입니다. ROX 액세스 모드를 사용하려면 복제본 생성 또는 새 읽기 전용 액세스 모드로 스냅샷 복원을 참조하십시오.

8. (선택 사항) RBD의 경우 Volume 모드를 선택합니다.

9. Restore (복구)를 클릭합니다. 새 PVC 세부 정보 페이지로 리디렉션됩니다.

Volume Snapshots(볼륨 스냅샷) 페이지에서

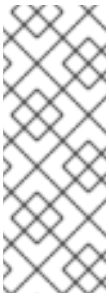
1. OpenShift 웹 콘솔에서 스토리지 → 볼륨 스냅숏 을 클릭합니다.

2. Volume Snapshots(볼륨 스냅샷) 탭에서 복원할 볼륨 스냅샷 옆에 있는 Action(작업 메뉴)을 클릭합니다.

3. Restore(복구)를 새 PVC로 클릭합니다.

4. 새 PVC의 이름을 입력합니다.

5. 스토리지 클래스 이름을 선택합니다.



참고

RBD(Rados Block Device)의 경우 상위 PVC와 동일한 풀을 사용하여 스토리지 클래스를 선택해야 합니다. 암호화가 활성화되지 않고 그 반대로 지원되지 않는 스토리지 클래스를 사용하여 암호화된 PVC의 스냅샷을 복원합니다.

6. 선택한 액세스 모드를 선택합니다.

중요

RX(ReadOnlyMany) 액세스 모드는 개발자 프리뷰 기능이며 개발자 프리뷰 지원 제한 사항이 있습니다. 개발자 미리보기 릴리스는 프로덕션 환경에서 실행되도록 설계되지 않으며 Red Hat 고객 포털 케이스 관리 시스템을 통해 지원되지 않습니다. ReadOnlyMany 기능에 대한 지원이 필요한 경우 ocs-devpreview@redhat.com 메일링 리스트와 Red Hat 개발 팀 구성원에게 연락하면 가용성 및 업무 일정에 따라 최대한 빨리 도움을 드릴 것입니다. ROX 액세스 모드를 사용하려면 복제본 생성 또는 새 읽기 전용 액세스 모드로 스냅샷 복원을 참조하십시오.

7. (선택 사항) RBD의 경우 Volume 모드를 선택합니다.

8. Restore (복구)를 클릭합니다. 새 PVC 세부 정보 페이지로 리디렉션됩니다.

검증 단계

- OpenShift 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭하고 새 PVC가 영구 볼륨 클레임 페이지에 나열되어 있는지 확인합니다.
- 새 PVC가 Bound 상태가 될 때까지 기다립니다.

12.3. 볼륨 스냅샷 삭제

사전 요구 사항

- 볼륨 스냅샷을 삭제하려면 특정 볼륨 스냅샷에 사용되는 볼륨 스냅샷 클래스가 있어야 합니다.

절차

영구 볼륨 클레임 페이지에서

1. OpenShift 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭합니다.
2. 삭제해야 하는 볼륨 스냅샷이 있는 PVC 이름을 클릭합니다.
3. Volume Snapshots(볼륨 스냅샷) 탭에서 원하는 볼륨 스냅샷 옆에 있는 Action menu(PS) → Delete Volume Snapshot(볼륨 스냅샷 삭제) 을 클릭합니다.

볼륨 스냅샷 페이지에서

1. OpenShift 웹 콘솔에서 스토리지 → 볼륨 스냅숏 을 클릭합니다.
2. Volume Snapshots(볼륨 스냅숏) 페이지의 원하는 볼륨 스냅샷 옆에 있는 Action(작업 메뉴) → Delete Volume Snapshot(볼륨 스냅샷 삭제) 을 클릭합니다.

검증 단계

- 삭제된 볼륨 스냅샷이 PVC 세부 정보 페이지의 Volume Snapshots(볼륨 스냅숏) 탭에 없는지 확인합니다.
- 스토리지 → 볼륨 스냅숏 을 클릭하고 삭제된 볼륨 스냅샷이 나열되지 않았는지 확인합니다.

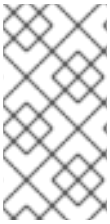
13장. 볼륨 복제

복제본은 표준 볼륨으로 사용되는 기존 스토리지 볼륨의 중복입니다. 볼륨 복제본을 생성하여 데이터의 시간 복사본을 만듭니다. PVC(영구 볼륨 클레임)는 다른 크기로 복제할 수 없습니다. CephFS 및 RADOS 블록 장치(RBD)에 대해 PVC당 최대 512개의 복제본을 생성할 수 있습니다.

13.1. 복제본 생성

사전 요구 사항

- 소스 PVC는 Bound 상태여야 하며 사용하지 않아야 합니다.

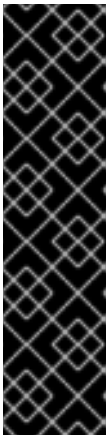


참고

Pod가 이를 사용하는 경우 PVC 복제본을 생성하지 마십시오. 그러면 PVC가 quiesced(일시 중지됨)되지 않기 때문에 데이터 손상이 발생할 수 있습니다.

절차

1. OpenShift 웹 콘솔에서 스토리지 → 영구 볼륨 클레임 을 클릭합니다.
2. 복제본을 생성하려면 다음 중 하나를 수행하십시오.
   - 원하는 PVC 옆의 작업 메뉴(PS) → Clone PVC (PVC 복제) 를 클릭합니다.
   - 복제할 PVC를 클릭하고 작업 → PVC 복제 를 클릭합니다.
3. 복제본의 이름을 입력합니다.
4. 선택한 액세스 모드를 선택합니다.



중요

RX(ReadOnlyMany) 액세스 모드는 개발자 프리뷰 기능이며 개발자 프리뷰 지원 제한 사항이 있습니다. 개발자 미리보기 릴리스는 프로덕션 환경에서 실행되도록 설계되지 않으며 Red Hat 고객 포털 케이스 관리 시스템을 통해 지원되지 않습니다. ReadOnlyMany 기능에 대한 지원이 필요한 경우 ocs-devpreview@redhat.com 메일링 리스트와 Red Hat 개발 팀 구성원에게 연락하면 가용성 및 업무 일정에 따라 최대한 빨리 도움을 드릴 것입니다. ROX 액세스 모드를 사용하려면 복제본 생성 또는 새 읽기 전용 액세스 모드로 스냅샷 복원을 참조하십시오.

5. Clone(복제)을 클릭합니다. 새 PVC 세부 정보 페이지로 리디렉션됩니다.

6. 복제된 PVC 상태가 Bound 가 될 때까지 기다립니다.

이제 Pod에서 복제된 PVC를 사용할 수 있습니다. 이 복제된 PVC는 dataSource PVC와 독립적입니다.

14장. 스토리지 노드 교체

다음 절차 중 하나를 선택하여 스토리지 노드를 교체할 수 있습니다.

- 14.1절. “Google Cloud 설치 관리자 프로비저닝 인프라에서 운영 노드 교체”
- 14.2절. “Google Cloud 설치 관리자 프로비저닝 인프라에서 실패한 노드 교체”

14.1. GOOGLE CLOUD 설치 관리자 프로비저닝 인프라에서 운영 노드 교체

Google Cloud 설치 관리자 프로비저닝 인프라(IPI)에서 운영 노드를 교체하려면 다음 절차를 사용하십시오.

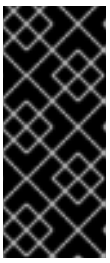
절차

1. OpenShift 웹 콘솔에 로그인하고 컴퓨팅 → 노드를 클릭합니다.
2. 교체해야 하는 노드를 확인합니다. Machine Name(시스템 이름)을 기록합니다.
3. 다음 명령을 사용하여 노드를 예약 불가능으로 표시합니다.

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

4. 다음 명령을 사용하여 노드를 드레이닝합니다.

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



중요

이 활동에는 최소 5-10분 이상 걸릴 수 있습니다. 이 기간 동안 생성된 Ceph 오류는 일시적이며 새 노드에 레이블이 지정되고 작동하는 경우 자동으로 해결됩니다.

5. 컴퓨팅 → 머신을 클릭합니다. 필요한 시스템을 검색합니다.

6. 필요한 시스템 외에도 작업 메뉴 (TI) → 머신 삭제를 클릭합니다.

7. Delete(삭제)를 클릭하여 머신 삭제를 확인합니다. 새 시스템이 자동으로 생성됩니다.

8. 새 시스템이 시작되고 Running 상태로 전환될 때까지 기다립니다.

중요

이 활동에는 최소 5-10분 이상 걸릴 수 있습니다.

9. 컴퓨팅 → 노드를 클릭하고 새 노드가 Ready 상태인지 확인합니다.

10. 다음 중 하나를 사용하여 OpenShift Container Storage 레이블을 새 노드에 적용합니다.

출처: 사용자 인터페이스

a. 새 노드의 경우 작업 메뉴 (PS) → 레이블 편집을클릭합니다.

b. cluster.ocs.openshift.io/openshift-storage 를 추가하고 Save(저장)를 클릭합니다.

명령줄 인터페이스의

- 다음 명령을 실행하여 OpenShift Container Storage 레이블을 새 노드에 적용합니다.

```
$ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
```

검증 단계

1. 다음 명령을 실행하고 새 노드가 출력에 있는지 확인합니다.

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

2. 워크로드 → **Pod** 를 클릭하여 새 노드의 다음 **Pod**가 **Running** 상태인지 확인합니다.

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

3. 기타 필요한 모든 **OpenShift Container Storage** 포드가 **Running** 상태인지 확인합니다.

4. 새 **OSD** 포드가 교체 노드에서 실행 중인지 확인합니다.

```
$ oc get pods -o wide -n openshift-storage| egrep -i new-node-name | egrep osd
```

5. (선택 사항) 클러스터에서 클러스터 전체 암호화가 활성화된 경우 새 **OSD** 장치가 암호화되었는지 확인합니다.

   이전 단계에서 식별된 각 새 노드에 대해 다음을 수행합니다.

   a. 디버그 포드를 만들고 선택한 호스트에 대해 **chroot** 환경을 엽니다.

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

   b. **"lsblk"**를 실행하고 **ocs-deviceset** 이름 옆의 **"crypt"** 키워드를 확인하십시오.

```
$ lsblk
```

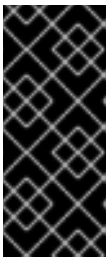
6. 확인 단계가 실패하는 경우 **Red Hat** 지원팀에 문의하십시오.

14.2. GOOGLE CLOUD 설치 관리자 프로비저닝 인프라에서 실패한 노드 교체

OpenShift Container Storage의 Google Cloud 설치 관리자 프로비저닝 인프라(IPI)에서 작동하지 않는 장애가 발생한 노드를 교체하려면 다음 절차를 수행하십시오.

절차

1. **OpenShift 웹 콘솔에 로그인하고 컴퓨팅 → 노드를 클릭합니다.**
2. **문제가 있는 노드를 식별하고 Machine Name(시스템 이름)을 클릭합니다.**
3. **작업 → 주석 편집을 클릭하고 추가를 클릭합니다.**
4. **machine.openshift.io/exclude-node-draining을 추가하고 저장을 클릭합니다.**
5. **작업 → 머신 삭제를 클릭하고 삭제 를 클릭합니다.**
6. **새 시스템이 자동으로 생성되고 새 머신이 시작될 때까지 기다립니다.**



중요

이 활동에는 최소 5-10분 이상 걸릴 수 있습니다. 이 기간 동안 생성된 Ceph 오류는 일시적이며 새 노드에 레이블이 지정되고 작동하는 경우 자동으로 해결됩니다.

7. **컴퓨팅 → 노드를 클릭하고 새 노드가 Ready 상태인지 확인합니다.**
8. **다음 중 하나를 사용하여 OpenShift Container Storage 레이블을 새 노드에 적용합니다.**

웹 사용자 인터페이스에서

a. **새 노드의 경우 작업 메뉴 (PS) → 레이블 편집을클릭합니다.**

b. **cluster.ocs.openshift.io/openshift-storage 를 추가하고 Save(저장)를 클릭합니다.**

명령줄 인터페이스에서

- 다음 명령을 실행하여 **OpenShift Container Storage** 레이블을 새 노드에 적용합니다.

```
$ oc label node <new_node_name> cluster.ocs.openshift.io/openshift-storage=""
```

9. **[선택 사항]: 실패한 Google Cloud 인스턴스가 자동으로 제거되지 않으면 Google Cloud 콘솔에서 인스턴스를 종료합니다.**

검증 단계

1. 다음 명령을 실행하고 새 노드가 출력에 있는지 확인합니다.

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

2. 워크로드 → **Pod** 를 클릭하여 새 노드의 다음 **Pod**가 **Running** 상태인지 확인합니다.

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

3. 기타 필요한 모든 **OpenShift Container Storage** 포드가 **Running** 상태인지 확인합니다.

4. 새 **OSD** 포드가 교체 노드에서 실행 중인지 확인합니다.

```
$ oc get pods -o wide -n openshift-storage| egrep -i new-node-name | egrep osd
```

5. **(선택 사항) 클러스터에서 클러스터 전체 암호화가 활성화된 경우 새 OSD 장치가 암호화되었는지 확인합니다.**

이전 단계에서 식별된 각 새 노드에 대해 다음을 수행합니다.

a. **디버그 포드를 만들고 선택한 호스트에 대해 chroot 환경을 엽니다.**

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

b. **"lsblk"를 실행하고 ocs-deviceset 이름 옆의 "crypt" 키워드를 확인하십시오.**

```
$ lsblk
```

6. **확인 단계가 실패하는 경우 Red Hat 지원팀에 문의하십시오.**

15장. 스토리지 장치 교체

15.1. GOOGLE CLOUD 설치 관리자 프로비저닝 인프라에서 운영 또는 실패한 스토리지 장치 교체

Google Cloud 설치 관리자 프로비저닝 인프라에서 동적으로 생성된 스토리지 클러스터에서 장치를 교체해야 하는 경우 스토리지 노드를 교체해야 합니다. 노드를 교체하는 방법에 대한 자세한 내용은 다음을 참조하십시오.

- Google Cloud 설치 관리자 프로비저닝 인프라에서 운영 노드 교체
- Google Cloud 설치 관리자 프로비저닝 인프라에서 실패한 노드를 교체합니다.

16장. UPDATING OPENSHIFT CONTAINER STORAGE

16.1. OPENSHIFT CONTAINER STORAGE 업데이트 프로세스 개요

4.7 및 4.8과 같은 마이너 릴리스 또는 4.8.0 및 4.8.1과 같은 배치 업데이트 간에 Red Hat OpenShift Container Storage 및 해당 구성 요소를 업그레이드할 수 있습니다.

특정 순서로 **OpenShift Container Storage**의 다른 부분을 업그레이드해야 합니다.

1. **OpenShift Container Platform**의 클러스터 업데이트 설명서에 따라 **OpenShift Container Platform**을 업데이트합니다.

2. **OpenShift Container Storage** 업데이트.

   a. 업데이트를 위해 연결이 끊긴 환경을 준비하려면 제한된 네트워크에서 **Operator Lifecycle Manager**를 사용하여 **Openshift Container Storage** 및 **Local Storage Operator**를 업데이트할 수 있도록 **Operator** 가이드를 참조하십시오.

   b. 설정에 적절한 프로세스를 사용하여 **OpenShift Container Storage Operator**를 업데이트합니다.

   - 내부 모드에서 **OpenShift Container Storage** 업데이트

업데이트 고려 사항

시작하기 전에 다음 중요한 고려 사항을 검토하십시오.

- **Red Hat OpenShift Container Storage**와 동일한 버전의 **Red Hat OpenShift Container Platform**을 사용하는 것이 좋습니다.

  OpenShift Container Platform 및 **OpenShift Container Storage**의 지원되는 조합에 대한 자세한 내용은 **Interoperability Matrix** 를 참조하십시오.

- **Local Storage Operator**는 **Local Storage Operator** 버전이 **Red Hat OpenShift Container Platform** 버전과 일치하는 경우에만 완전히 지원됩니다.

16.2. 연결이 끊긴 환경에서 업데이트 준비

Red Hat OpenShift Container Storage 환경이 인터넷에 직접 연결되지 않은 경우 기본 **Operator Hub** 및 이미지 레지스트리에 대한 대안으로 **OLM(Operator Lifecycle Manager)**을 제공하기 위해 일부 추가 구성이 필요합니다.

자세한 내용은 **OpenShift Container Platform** 설명서를 참조하십시오. **Operator** 카탈로그 이미지 업데이트.

연결이 끊긴 업데이트를 위해 클러스터를 구성하려면 다음을 수행합니다.

1. 대체 레지스트리에 대한 인증을 구성합니다.
2. **Red Hat** 운영 프로그램 카탈로그를 빌드하고 미러링 합니다.
3. **Operator imageContentSourcePolicy** 생성
4. **redhat-operator catalogsource** 업데이트

이러한 단계가 완료되면 정상적으로 업데이트를 계속합니다.

16.2.1. 미러 레지스트리 인증 세부 정보 추가

사전 요구 사항

- 연결이 끊긴 기존 클러스터에서 **OpenShift Container Platform 4.3** 이상을 사용하는지 확인합니다.
- **oc** 클라이언트 버전이 **4.4** 이상인지 확인합니다.
- 미러 레지스트리를 사용하여 미러 호스트를 준비합니다. 자세한 내용은 미러 호스트 준비를 참조하십시오.

절차

1. cluster -admin 역할을 사용하여 OpenShift Container Platform 클러스터에 로그인합니다.

2. auth.json 파일을 찾습니다.

   이 파일은 podman 또는 docker를 사용하여 레지스트리에 로그인할 때 생성됩니다. 다음 위치 중 하나에 있습니다.

   - ~/.docker/auth.json
   - /run/user/<UID>/containers/auth.json
   - /var/run/containers/<UID>/auth.json

3. 고유한 Red Hat 레지스트리 가져오기 시크릿을 가져와 auth.json 파일에 붙여넣습니다. 다음과 같이 나타납니다.

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

4. 설정에 대한 적절한 세부 정보를 사용하여 환경 변수를 내보냅니다.

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

5. **podman** 을 사용하여 미러 레지스트리에 로그인하고 자격 증명을 **${AUTH_FILE}** 에 저장합니다.

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

그러면 미러 레지스트리가 **auth.json** 파일에 추가됩니다.

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

16.2.2. Red Hat Operator 카탈로그 빌드 및 미러링

Red Hat 레지스트리에 액세스할 수 있는 호스트에서 다음 프로세스를 수행하여 해당 레지스트리의 미러를 생성합니다.

사전 요구 사항

- 클러스터 관리자로 다음 명령을 실행합니다.
- **redhat-operator** 카탈로그를 미러링하는 데 몇 시간이 걸릴 수 있으며 미러 호스트에서 사용 가능한 디스크 공간이 필요합니다.

절차

1. **redhat-operators** 의 카탈로그를 빌드합니다.

   대상 **OpenShift Container Platform** 클러스터 주 버전 및 부 버전과 일치하는 태그를 사용하여 **--from**을 **ose-operator-registry** 기본 이미지로 설정합니다.

```
$ oc adm catalog build --appregistry-org redhat-operators \
  --from=registry.redhat.io/openshift4/ose-operator-registry:v4.7 \
  --to=${MIRROR_REGISTRY_DNS}/olm/redhat-operators:v2 \
  --registry-config=${AUTH_FILE} \
  --filter-by-os="linux/amd64" --insecure
```

   참고

   IBM Power Systems 및 **IBM Z** 인프라의 경우 **filter-by-os** 의 값을 **linux/ppc64le** 및 **linux/s390x** 로 지정합니다.

2. **redhat-operators** 의 카탈로그를 미러링합니다.

   이 작업은 오래 걸리며 **1-5**시간이 걸릴 수 있습니다. 미러 호스트에 **100GB**의 사용 가능한 디스크 공간이 있는지 확인합니다.

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

16.2.3. Operator imageContentSourcePolicy생성

oc adm catalog mirror 명령이 완료되면 **imageContentSourcePolicy.yaml** 파일이 생성됩니다. 이 파일의 출력 디렉터리는 일반적으로 **./[catalog** 이미지 이름**]-manifests**입니다. 누락된 항목을 **.yaml** 파일에 추가하고 클러스터에 적용하려면 다음 절차를 사용하십시오.

절차

1. 이 파일의 콘텐츠에서 다음과 같이 표시된 미러 매핑을 확인합니다.

```
spec:
  repositoryDigestMirrors:
    - mirrors:
```

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

2. 누락된 항목을 **imageContentSourcePolicy.yaml** 파일 끝에 추가합니다.

3. **imageContentSourcePolicy.yaml** 파일을 클러스터에 적용합니다.

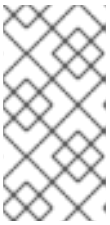
```
$ oc apply -f ./[output dir]/imageContentSourcePolicy.yaml
```

Image Content Source Policy가 업데이트되면 클러스터의 모든 노드(마스터, 인프라, 작업자)를 업데이트하고 재부팅해야 합니다. 이 프로세스는 **Machine Config Pool** 운영자를 통해 자동으로 처리되며, **OpenShift** 클러스터의 노드 수에 따라 정확한 경과 시간이 다를 수 있지만 최대 **30**분이 걸립니다. **oc get mcp** 명령 또는 **oc get node** 명령을 사용하여 업데이트 프로세스를 모니터링할 수 있습니다.

16.2.4. redhat-operator CatalogSource업데이트

절차

1. **Red Hat** 운영자의 카탈로그 이미지를 참조하는 **CatalogSource** 오브젝트를 다시 생성합니다.



참고

올바른 버전(즉, **v2**)을 사용하여 올바른 카탈로그 소스를 미러링했는지 확인합니다.

<your_registry> 를 미러 레지스트리 **URL**로 교체해야 한다는 점을 기억하여 **redhat-operator-catalogsource.yaml** 파일에 다음을 저장합니다.

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
```

```
  name: redhat-operators
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  icon:
    base64data:
PHN2ZyBpZD0iTGF5ZXJfMSIgZGF0YS1uYW1lPSJMYXllciAxIiB4bWxucz0iaHR0cDovL3d3dy
53My5vcmcvMjAwMC9zdmciIHZpZXdCb3g9IjAgMCAxOTIgMTQ1Ij48ZGVmcz48c3R5bGU+L
mNscy0xe2ZpbGw6I2UwMDt9PC9zdHlsZT48L2RlZnM+PHRpdGxlPlJlZEhhdC1Mb2dvLUhhd
C1Db2xvcjwvdGl0bGU+PHBhdGggZD0iTTE1Ny43Nyw2Mi42MWExNCwxNCwwLDAsMSwuM
zEsMy40MmMwLDE0Ljg4LTE4LjEsMTcuNDYtMzAuNjEsMTcuNDZDNzguODMsODMuNDksN
DIuNTMsNTMuMjYsNDIuNTMsNDRhNi40Myw2LjQzLDAsMCwxLC4yMi0xLjk0bC0zLjY2LDku
MDZhMTguNDUsMTguNDUsMCwwLDAtMS41MSw3LjMzYzAsMTguMTEsNDEsNDUuNDgs
ODcuNzQsNDUuNDgsMjAuNjksMCwzNi40My03Ljc2LDM2LjQzLTIxLjc3LDAtMS4wOCwwLTE
uOTQtMS43My0xMC4xM1oiLz48cGF0aCBjbGFzcz0iY2xzLTEiIGQ9Ik0xMjcuNDcsODMuNDlj
MTIuNTEsMCwzMC42MS0yLjU4LDMwLjYxLTE3LjQ2YTE0LDE0LDAsMCwwLS4zMS0zLjQyb
C03LjQ1LTMyLjM2Yy0xLjcyLTcuMTItMy4yMy0xMC4zNS0xNS43My0xNi42QzEyNC44OSw4Lj
Y5LDEwMy43Ni41LDk3LjUxLjUsOTEuNjkuNSw5MCw4LDgzLjA2LDhjLTYuNjgsMC0xMS42N
C01LjYtMTcuODktNS42LTYsMC05LjkxLDQuMDktMTIuOTMsMTIuNSwwLDAtOC40MSwyMy
43Mi05LjQ5LDI3LjE2QTYuNDMsNi40MywwLDAsMCw0Mi41Myw0NGMwLDkuMjIsMzYuMywz
OS40NSw4NC45NCwzOS40NU0xNjAsNzIuMDdjMS43Myw4LjE5LDEuNzMsOS4wNSwxLjczLjczL
DEwLjEzLDAsMTQtMTUuNzQsMjEuNzctMzYuNDMsMjEuNzdDNzguNTQsMTA0LDM3LjU4L
Dc2LjYsMzcuNTgsNTguNDlhMTguNDUsMTguNDUsMCwwLDEsMS41MS03LjMzQzIyLjI3LDU
yLC41LDU1LC41LDc0LjIyYzAsMzEuNDgsNzQuNTksNzAuMjgsMTMzLjY1LDcwLjI4LDQ1LjI4L
DAsNTYuNy0yMC40OCw1Ni43LTM2LjY1LDAtMTIuNzItMTEtMjcuMTYtMzAuODMtMzUuNzgi
Lz48L3N2Zz4=
    mediatype: image/svg+xml
  image: <your_registry>/olm/redhat-operators:v2
  displayName: Redhat Operators Catalog
  publisher: Red Hat
```

2. **redhat-operator-catalogsource.yaml 파일을 사용하여 카탈로그 소스를 생성합니다.**

```
$ oc apply -f redhat-operator-catalogsource.yaml
```

3. **새 redhat-operator 포드가 실행 중인지 확인합니다.**

```
$ oc get pod -n openshift-marketplace | grep redhat-operators
```

16.2.5. 계속 업데이트

대체 카탈로그 소스를 구성한 후에는 적절한 업데이트 프로세스를 계속 진행할 수 있습니다.

- 내부 모드에서 OpenShift Container Storage 업데이트

16.3. 내부 모드에서 OPENSHIFT CONTAINER STORAGE 업데이트

다음 절차에 따라 내부 모드에 배포된 **OpenShift Container Storage** 클러스터를 업데이트합니다.

16.3.1. 내부 모드에서 OpenShift Container Storage Operator 자동 업데이트 활성화

OpenShift Container Platform에서 **OpenShift Container Storage Operator**를 업데이트하기 위해 자동 업데이트 승인을 활성화하려면 다음 절차를 사용하십시오.

사전 요구 사항

- 상태 카드의 블록 및 파일 아래에 *스토리지 클러스터* 및 *데이터 복구에 녹색* 표시가 있는지 확인합니다.
- 상태 카드의 **Object** 에서 ***Object Service*** 및 ***Data Resiliency*** 가 **Ready** 상태(**Green tick**) 상태인지 확인합니다.
- **OpenShift Container Platform** 클러스터를 안정적인 버전 **4.8.X**의 최신 릴리스로 업데이트하고 클러스터 업데이트를 참조하십시오.
- **Red Hat OpenShift Container Storage** 채널을 **stable-4.7**에서 **stable -4.8** 로 전환합니다. 채널에 대한 자세한 내용은 **OpenShift Container Storage** 업그레이드 채널 및 릴리스를 참조하십시오.

참고

마이너 버전(예: **4.7**에서 **4.8**로 업데이트)과 **4.8**의 배치 업데이트(예: **4.8.0**에서 **4.8.1**) 간에 업데이트하는 경우에만 채널을 전환해야 합니다.

- 운영자 포드를 포함한 모든 **OpenShift Container Storage Pod**가 **openshift-storage** 네임스페이스 의 **Running** 상태인지 확인합니다.

  Pod 상태를 보려면 **OpenShift** 웹 콘솔의 왼쪽 창에서 워크로드 → **Pod** 를 클릭합니다. 프로젝트 드롭다운 목록에서 **openshift-storage** 를 선택합니다.

- 업데이트 시간이 클러스터에서 실행되는 **OSD** 수에 따라 다르므로 **Openshift Container Storage** 업데이트 프로세스를 완료할 충분한 시간이 있는지 확인합니다.

절차

1. **OpenShift 웹 콘솔에 로그인합니다.**

2. **Operators → 설치된 Operator**를 클릭합니다.

3. **openshift-storage** 프로젝트를 선택합니다.

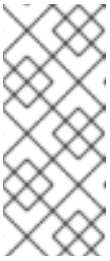
4. **OpenShift Container Storage** 운영자 이름을 클릭합니다.

5. **Subscription** (서브스크립션) 탭을 클릭하고 **Approval**(승인) 아래의 링크를 클릭합니다 .

6. **Automatic (default)**(자동(기본값)) 을 선택하고 **Save**(저장)를 클릭합니다.

7. 업그레이드 상태에 따라 다음 중 하나를 수행하십시오.

- **Upgrade Status**(업그레이드 상태) *표시에 승인이 필요합니다* .



참고

업그레이드 상태 표시 에는 새 **OpenShift Container Storage** 버전이 이미 채널에서 이미 감지되어 있고 업데이트 시 **Manual** 에서 **Automatic** 로 승인 전략이 변경된 경우 승인 상태가 필요합니다.

a. **Install Plan** (계획 설치) 링크를 클릭합니다.

b. **InstallPlan** 세부 정보 페이지에서 설치 계획 프리뷰 를 클릭합니다.

c. 설치 계획을 검토하고 승인을 클릭합니다.

d. **Status**(상태)가 **Unknown** (알 수 없음)에서 **Created** (생성) 로 변경될 때까지 기

다립니다.

e. **Operators** → 설치된 **Operator**를 클릭합니다.

f. **openshift-storage** 프로젝트를 선택합니다.

g. **Status**(상태) 가 **Up to date**로 변경될 때까지 기다립니다.

- 업그레이드 상태에 승인이 *필요하지* 않습니다.

a. 업데이트가 시작될 때까지 기다립니다. 이 작업은 최대 **20**분이 걸릴 수 있습니다.

b. **Operators** → 설치된 **Operator**를 클릭합니다.

c. **openshift-storage** 프로젝트를 선택합니다.

d. **Status**(상태) 가 **Up to date**로 변경될 때까지 기다립니다.

검증 단계

1. **OpenShift** 웹 콘솔에서 스토리지 → 개요 → 오브젝트 탭으로 이동합니다.

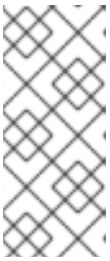
- 상태 카드에서 ***Object Service*** 및 ***Data Resiliency*** 가 **Ready** 상태(**Green tick**) 상태인지 확인합니다.

2. **OpenShift** 웹 콘솔에서 스토리지 → 개요 → 블록 및 파일 탭으로 이동합니다.

- 상태 카드에서 ***Storage Cluster*** 및 ***Data Resiliency*** 에 녹색 눈금 표시가 있는지 확인합니다.

3. **Operator** → 설치된 **Operator** → **OpenShift Container Storage Operator**를 클릭합니다. 스

토리지 클러스터에서 클러스터 서비스 상태가 **Ready** 인지 확인합니다.



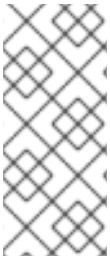
참고

OpenShift Container Storage 버전 **4.7**에서 **4.8**로 업데이트되면 버전 필드에 **4.7**이 계속 표시됩니다. **ocs-operator** 가 이 필드에 표시된 문자열을 업데이트하지 않기 때문입니다.

4. 운영자 포드를 포함한 모든 **OpenShift Container Storage Pod**가 **openshift-storage** 네임스페이스 의 **Running** 상태인지 확인합니다.

Pod 상태를 보려면 워크로드 → **Pod** 를 클릭합니다. 프로젝트 드롭다운 목록에서 **openshift-storage** 를 선택합니다.

5. 확인 단계가 실패하는 경우 **Red Hat** 지원팀에 문의하십시오.



참고

유연한 확장 기능은 **Red Hat OpenShift Container Storage 4.7**의 새로운 배포에서만 사용할 수 있습니다. **4.7** 버전으로 업그레이드된 스토리지 클러스터는 유연한 확장을 지원하지 않습니다.

추가 리소스

OpenShift Container Storage를 업데이트하는 동안 문제가 발생하는 경우 *문제 해결 가이드의 일반적으로 필요한 로그를 참조하십시오*.

16.3.2. 내부 모드에서 OpenShift Container Storage Operator 수동 업데이트

설치 계획에 대한 수동 승인을 제공하여 **OpenShift Container Storage Operator**를 업데이트하려면 다음 절차를 사용하십시오.

사전 요구 사항

- 상태 카드의 블록 및 파일 아래에 *스토리지 클러스터* 및 *데이터 복구에 녹색* 표시가 있는지 확인합니다.

상태 카드의 **Object** 에서 ***Object Service*** 및 ***Data Resiliency*** 가 **Ready** 상태(**Green tick**) 상태인지 확인합니다.

- **OpenShift Container Platform** 클러스터를 안정적인 버전 **4.8.X**의 최신 릴리스로 업데이트하고 클러스터 업데이트를 참조하십시오.

- **Red Hat OpenShift Container Storage** 채널을 **stable-4.7**에서 **stable -4.8** 로 전환합니다. 채널에 대한 자세한 내용은 **OpenShift Container Storage** 업그레이드 채널 및 릴리스를 참조하십시오.

참고

마이너 버전(예: **4.7**에서 **4.8**로 업데이트)과 **4.8**의 배치 업데이트(예: **4.8.0**에서 **4.8.1**) 간에 업데이트하는 경우에만 채널을 전환해야 합니다.

- 운영자 포드를 포함한 모든 **OpenShift Container Storage Pod**가 **openshift-storage** 네임스페이스 의 **Running** 상태인지 확인합니다.

Pod 상태를 보려면 **OpenShift** 웹 콘솔의 왼쪽 창에서 워크로드 → **Pod** 를 클릭합니다. 프로젝트 드롭다운 목록에서 **openshift-storage** 를 선택합니다.

- 업데이트 시간이 클러스터에서 실행되는 **OSD** 수에 따라 다르므로 **Openshift Container Storage** 업데이트 프로세스를 완료할 충분한 시간이 있는지 확인합니다.

절차

1. **OpenShift** 웹 콘솔에 로그인합니다.

2. **Operators** → 설치된 **Operator**를 클릭합니다.

3. **openshift-storage** 프로젝트를 선택합니다.

4. **OpenShift Container Storage** 운영자 이름을 클릭합니다.

5. **Subscription** (서브스크립션) 탭을 클릭하고 **Approval**(승인) 아래의 링크를 클릭합니다 .

6. **Manual** 을 선택하고 **Save**(저장)를 클릭합니다.

7. **Upgrade Status**(업그레이드 상태)가 **Upgrading** 으로 변경될 때까지 기다립니다.

8. 업그레이드 상태에 승인이 필요한 경우 **requires approval** (승인 필요)를 클릭합니다.

9. **InstallPlan** 세부 정보 페이지에서 설치 계획 프리뷰 를 클릭합니다.

10. 설치 계획을 검토하고 승인을 클릭합니다.

11. **Status**(상태)가 **Unknown** (알 수 없음)에서 **Created** (생성) 로 변경될 때까지 기다립니다.

12. **Operators** → 설치된 **Operator**를 클릭합니다.

13. **openshift-storage** 프로젝트를 선택합니다.

14. **Status**(상태) 가 **Up to date**로 변경될 때까지 기다립니다.

검증 단계

1. **OpenShift** 웹 콘솔에서 스토리지 → 개요 → 오브젝트 탭으로 이동합니다.

- 상태 카드에서 ***Object Service*** 및 ***Data Resiliency*** 가 **Ready** 상태(**Green tick**) 상태인지 확인합니다.

2. **OpenShift** 웹 콘솔에서 스토리지 → 개요 → 블록 및 파일 탭으로 이동합니다.

- 상태 카드에서 ***Storage Cluster*** 및 ***Data Resiliency*** 에 녹색 눈금 표시가 있는지 확인

합니다.

3. **Operator** → 설치된 **Operator** → **OpenShift Container Storage Operator**를 클릭합니다. 스토리지 클러스터에서 클러스터 서비스 상태가 **Ready** 인지 확인합니다.

참고

OpenShift Container Storage 버전 **4.7**에서 **4.8**로 업데이트되면 버전 필드에 **4.7**이 계속 표시됩니다. **ocs-operator** 가 이 필드에 표시된 문자열을 업데이트하지 않기 때문입니다.

4. 운영자 포드를 포함한 모든 **OpenShift Container Storage Pod**가 **openshift-storage** 네임스페이스 의 **Running** 상태인지 확인합니다.

Pod 상태를 보려면 **OpenShift** 웹 콘솔의 왼쪽 창에서 워크로드 → **Pod** 를 클릭합니다. 프로젝트 드롭다운 목록에서 **openshift-storage** 를 선택합니다.

5. 확인 단계가 실패하는 경우 **Red Hat** 지원팀에 문의하십시오.

추가 리소스

OpenShift Container Storage를 업데이트하는 동안 문제가 발생하는 경우 *문제 해결 가이드의 일반적으로 필요한 로그를 참조하십시오.*