Red Hat OpenShift Data Foundation 4.10

Advanced Cluster Management를 사용하여 Metro-DR용 OpenShift Data Foundation 구성

DEVELOPER PRETION: Metro-DR 기능을 사용하여 OpenShift Data Foundation 설정에 대한 지침입니다. 이 솔루션은 개발자 프리뷰 기능이며 프로덕션 환경에서 실행할 수 없습니다.

Last Updated: 2023-09-20

Red Hat OpenShift Data Foundation 4.10 Advanced Cluster Management를 사용하여 Metro-DR용 OpenShift Data Foundation 구성

DEVELOPER PRETION: Metro-DR 기능을 사용하여 OpenShift Data Foundation 설정에 대한 지침입니다. 이 솔루션은 개발자 프리뷰 기능이며 프로덕션 환경에서 실행할 수 없습니다.

법적 공지

초록

이 솔루션 가이드의 의도는 고가용성 스토리지 인프라를 구현하기 위해 고급 클러스터 관리로 재해 복구를 위해 OpenShift Data Foundation을 배포하는 데 필요한 단계를 자세히 설명합니다. Configuring OpenShift Data Foundation for Metro-DR with Advanced Cluster Management is a Developer Preview feature and is subject to Developer Preview support limitations. Developer Preview releases are not intended to be run in production environments and are not supported through the Red Hat Customer Portal case management system. If you need assistance with Developer Preview features, reach out to the ocs-devpreview@redhat.com mailing list and a member of the Red Hat Development Team will assist you as quickly as possible based on their availability and work schedules.

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. 개선할 내용에 대해 알려주십시오. 피드백을 보내주시려면 다음을 확인하십시오.

- 특정 문구에 대한 간단한 의견 작성 방법은 다음과 같습니다.
  1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **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. **구성 요소** 섹션에서 **문서** 를 선택합니다.
  3. **설명** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
  4. **버그 제출**을 클릭합니다.

1장. 메트로 - DR 소개

재해 복구는 비즈니스 크리티컬 애플리케이션을 자연 또는 사람이 생성한 재해에서 복구 및 지속할 수 있는 기능입니다. 이는 주요 이벤트 중 비즈니스 운영의 연속성을 유지하기 위해 설계된 주요 조직의 전반적인 비즈니스 지속적인 전략의 구성 요소입니다.

Metro-DR 기능은 동일한 지역에 있는 사이트 간에 볼륨 영구 데이터 및 메타데이터 복제를 제공합니다. 퍼블릭 클라우드에서 이는 가용 영역 장애로부터 보호하는 것과 유사합니다. Metro-DR은 데이터 손실 없이 데이터 센터를 사용할 수 없는 동안 비즈니스 연속성을 보장합니다. 이는 일반적으로 복구 포인트 목표(RPO) 및 복구 시간 목표(RTO)로 표시됩니다.

- RPO는 백업 또는 영구 데이터의 스냅샷을 수행하는 빈도를 나타냅니다. 실제로 RPO는 손실되거나 중단 후 다시 입력해야 하는 데이터의 양을 나타냅니다. Metro-DR 솔루션은 데이터가 동기식으로 복제되기 때문에 RPO가 0임을 보장합니다.

- RTO는 비즈니스에서 허용할 수 있는 다운 타임입니다. RTO는 "비즈니스 중단에 대한 알림을 받은 후 시스템을 복구하는 데 얼마나 오랜 시간이 걸릴 수 있습니까?"

이 가이드의 목적은 Metro Disaster Recovery (Metro-DR) 단계 및 명령을 자세히 설명하는 것입니다. Red Hat OpenShift Container Platform 클러스터에서 다른 Red Hat OpenShift Container Platform 클러스터에서 다른 애플리케이션으로 애플리케이션을 장애 조치한 다음 동일한 애플리케이션을 원래 기본 클러스터로 장애 조치합니다. 이 경우 RHOCP 클러스터는 RHACM(Red Hat Advanced Cluster Management)을 사용하여 생성 또는 가져오고, ***RHOCP 클러스터 10ms RTT 대기 시간 미만의 제한 사항이*** 있습니다.

애플리케이션의 영구 스토리지는 이 스토리지 클러스터에 연결된 RHOCP 인스턴스와 함께 두 위치 간에 확장되는 외부 **Red Hat Ceph Storage**클러스터에서 제공합니다. 스토리지 모니터 서비스가 있는 중재자 노드는 사이트 중단 시 Red Hat Ceph Storage 클러스터에 대한 쿼럼을 설정하기 위해 third location (different location)에 필요합니다. 세 번째 위치의 대기 시간이 완화되었습니다. 이 요구 사항은 RHOCP 인스턴스에 연결된 스토리지 클러스터에서 최대 100ms RTT 대기 시간을 지원합니다.

1.1. 메트로-DR 솔루션의 구성 요소

Metro-DR은 Red Hat Advanced Cluster Management for Kubernetes, Red Hat Ceph Storage 및 OpenShift Data Foundation 구성 요소로 구성되어 OpenShift Container Platform 클러스터 전체에서 애플리케이션 및 데이터 이동성을 제공합니다.

Red Hat Advanced Cluster Security for Kubernetes

RHACM(Red Hat Advanced Cluster Management)은 여러 클러스터 및 애플리케이션 라이프사이클을 관리할 수 있는 기능을 제공합니다. 따라서 다중 클러스터 환경에서 컨트롤 플레인 역할을 합니다.

RHACM은 다음 두 부분으로 나뉩니다.

- RHACM Hub: 멀티 클러스터 컨트롤 플레인에서 실행되는 구성 요소

- 관리형 클러스터: 관리되는 클러스터에서 실행되는 구성 요소

이 제품에 대한 자세한 내용은 RHACM 설명서 및 RHACM "애플리케이션 관리" 설명서를 참조하십시오.

Red Hat Ceph Storage

Red Hat Ceph Storage는 대규모 확장이 가능한 오픈 소프트웨어 정의 스토리지 플랫폼으로서 가장 안정적인 Ceph 스토리지 시스템과 Ceph 관리 플랫폼, 배포 유틸리티 및 지원 서비스를 결합합니다. 이는 엔터프라이즈 데이터를 저장하는 비용을 크게 낮추고 조직이 급격한 데이터 증가를 관리하는 데 도움이 됩니다. 이 소프트웨어는 퍼블릭 또는 프라이빗 클라우드 배포를 위한 강력하고 최신 페타바이트 규모의 스토리지 플랫폼입니다.

OpenShift Data Foundation

OpenShift Data Foundation은 OpenShift Container Platform 클러스터에서 상태 저장 애플리케이션에 대한 스토리지를 프로비저닝하고 관리할 수 있는 기능을 제공합니다. OpenShift Data Foundation 구성 요소 스택의 Rook에서 라이프사이클이 관리되고 Ceph-CSI는 상태 저장 애플리케이션을 위한 영구 볼륨 프로비저닝 및 관리를 제공하는 스토리지 공급자로 Ceph에서 지원합니다.

OpenShift Data Foundation 스택은 영구 볼륨 클레임 미러링에 따라 관리할 수 있는 **csi-addons** 를 제공하여 향상되었습니다.

OpenShift DR

OpenShift DR은 RHACM을 사용하여 배포 및 관리되는 피어 OpenShift 클러스터 세트에 대한 상태 저장 애플리케이션의 재해 복구 오케스트레이터이며 영구 볼륨에서 애플리케이션 상태의 라이프 사이클을 오케스트레이션하는 클라우드 네이티브 인터페이스를 제공합니다. 여기에는 다음이 포함됩니다.

- OpenShift 클러스터에서 애플리케이션 상태 관계 보호
- 피어 클러스터로 애플리케이션 상태를 장애 조치
- 애플리케이션 상태를 이전에 배포한 클러스터로 재배치

OpenShift DR은 다음 두 가지 구성 요소로 나뉩니다.

- **OpenShift DR Hub Operator**: 애플리케이션에 대한 장애 조치 및 재배치를 관리하기 위해 허브 클러스터에 설치되었습니다.
- **OpenShift DR Cluster Operator**: 애플리케이션의 모든 PVC 라이프사이클을 관리하기 위해 각 관리 클러스터에 설치됨.

1.2. SCALE-DR 배포 워크플로

이 섹션에서는 두 가지 OpenShift Container Platform 클러스터에서 OpenShift Data Foundation 버전 4.10, RHCS 5 및 RHACM 최신 버전을 사용하여 Metro-DR 기능을 구성하고 배포하는 데 필요한 단계를 간략하게 설명합니다. 두 개의 관리형 클러스터 외에도 고급 클러스터 관리를 배포하려면 세 번째 OpenShift Container Platform 클러스터가 필요합니다.

인프라를 구성하려면 다음과 같은 순서로 다음 단계를 수행하십시오.

1. RHACM Operator 설치, 생성 또는 가져오기를 포함하는 각각의 메트로-DR 요구 사항을 RHACM 허브 및 네트워크 구성으로 충족해야 합니다. Metro-DR 활성화 요구 사항을 참조하십시오.
2. 중재자를 사용하여 Red Hat Ceph Storage stretch 클러스터 배포 요구 사항을 충족하는지 확인하십시오. Red Hat Ceph Storage 배포에 대한 요구 사항을 참조하십시오.
3. Red Hat Ceph Storage 확장 클러스터 모드를 구성합니다. 확장 모드 기능을 사용하여 두 개의 다른 데이터 센터에서 Ceph 클러스터를 활성화하는 방법에 대한 지침은 Red Hat Ceph Storage 확장 클러스터 구성을 참조하십시오.
4. 기본 및 보조 관리 클러스터에 OpenShift Data Foundation 4.10을 설치합니다. 관리형 클러스터에 OpenShift Data Foundation 설치를 참조하십시오.
5. Hub 클러스터에 Openshift DR Hub Operator를 설치합니다. Hub 클러스터에 OpenShift DR Hub Operator 설치를 참조하십시오.
6. 관리형 및 Hub 클러스터를 구성합니다. 관리형 및 hub 클러스터 구성을 참조하십시오.

7. 관리되는 클러스터에 워크로드를 배포, 장애 조치 및 재배치하는 데 사용되는 허브 클러스터에서 DRPolicy 리소스를 생성합니다. Hub 클러스터에서 재해 복구 정책 생성 을 참조하십시오.

8. OpenShift DR Cluster Operator의 자동 설치를 활성화하고 관리 클러스터에서 S3 시크릿을 자동으로 전송할 수 있습니다. 자세한 내용은 관리 클러스터에서 S3 시크릿 자동 전송 활성화 및 OpenShift DR 클러스터 Operator 자동 설치 활성화를 참조하십시오.

9. 장애 조치 및 재배치 테스트를 테스트하기 위해 RHACM 콘솔을 사용하여 샘플 애플리케이션을 생성합니다. 자세한 내용은 관리형 클러스터 간에 샘플 애플리케이션 생성,애플리케이션 장애 조치 및 애플리케이션 재배치를 참조하십시오.

2장. 메트로 - DR 활성화 요구 사항

Red Hat OpenShift Data Foundation에서 지원하는 재해 복구 기능을 사용하려면 재해 복구 솔루션을 성공적으로 구현하기 위해 다음과 같은 사전 요구 사항이 모두 필요합니다.

- 서브스크립션 요구 사항
  - 유효한 Red Hat OpenShift Data Foundation Advanced Entitlement
  - 유효한 Red Hat Advanced Cluster Management for Kubernetes 서브스크립션

  OpenShift Data Foundation의 서브스크립션이 작동하는 방법을 알아보려면 OpenShift Data Foundation 서브스크립션에 대한 기술 자료 문서 를 참조하십시오.

- 네트워크 연결이 가능한 세 개의 OpenShift 클러스터가 있어야 합니다.
  - Kubernetes(Advanced Cluster Management for Kubernetes) 및 OpenShift DR Hub 컨트롤러가 설치된 Hub **클러스터** 입니다.
  - OpenShift Data Foundation, OpenShift DR **클러스터 컨트롤러 및 애플리케이션이 설치된 기본 관리형** 클러스터입니다.
  - OpenShift Data Foundation, OpenShift DR **클러스터 컨트롤러 및 애플리케이션이 설치된 보조 관리형** 클러스터입니다.

- RHACM Operator 및 MultiClusterResourceOverride가 Hub 클러스터에 설치되어 있는지 확인합니다. 자세한 내용은 RHACM 설치 가이드 를 참조하십시오.
  - 배포가 완료되면 OpenShift 자격 증명을 사용하여 RHACM 콘솔에 로그인합니다.
  - Advanced Cluster Manager 콘솔에 대해 생성된 경로를 찾습니다.

    ```
    $ oc get route multicloud-console -n open-cluster-management -o jsonpath --
    template="https://{.spec.host}/multicloud/clusters{'\n'}"
    ```

    출력 예:

    ```
    https://multicloud-console.apps.perf3.example.com/multicloud/clusters
    ```

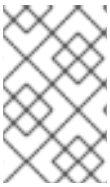
    OpenShift 인증 정보를 사용하여 로그인하면 로컬 클러스터가 표시됩니다.

- RHACM 콘솔을 사용하여 **기본 관리 클러스터** 와 **보조 관리 클러스터**를 가져오거나 만들어야 합니다. 환경에 적합한 옵션을 선택합니다. 관리형 클러스터를 성공적으로 생성하거나 가져온 후 콘솔에서 가져오거나 생성한 클러스터 목록을 확인할 수 있습니다.

3장. 중재자를 사용하여 RED HAT CEPH STORAGE 확장 클러스터를 배포하는 데 필요한 요구 사항

Red Hat Ceph Storage는 표준 경제적 서버 및 디스크에서 통합된 소프트웨어 정의 스토리지를 제공하는 오픈 소스 엔터프라이즈 플랫폼입니다. 블록, 오브젝트 및 파일 스토리지를 하나의 플랫폼으로 결합하면 Red Hat Ceph Storage가 모든 데이터를 효율적이고 자동으로 관리하므로 이를 사용하는 애플리케이션 및 워크로드에 집중할 수 있습니다.

이 섹션에서는 Red Hat Ceph Storage 배포에 대한 기본 개요를 제공합니다. 보다 복잡한 배포의 경우 RHCS 5의 공식 설명서 가이드를 참조하십시오.



참고

성능 저하된 경우 **min_size=1** 에서 실행되기 때문에 플래쉬 미디어만 지원됩니다. all-flash OSD에서만 스트레칭 모드를 사용합니다. 일체형 OSD를 사용하면 연결이 복원되면 복구에 필요한 시간을 최소화하여 데이터 손실 가능성을 최소화할 수 있습니다.

중요

파레저 코딩된 풀은 스트레칭 모드에서는 사용할 수 없습니다.

3.1. 하드웨어 요구 사항

Red Hat Ceph Storage 배포를 위한 최소 하드웨어 요구 사항에 대한 자세한 내용은 컨테이너화된 Ceph 권장 사항 최소 하드웨어 권장 사항을 참조하십시오.

표 3.1. Red Hat Ceph Storage 클러스터 배포를 위한 물리적 서버 위치 및 Ceph 구성 요소 레이아웃:

노드 이름	데이터 센터	Ceph 구성 요소
ceph1	DC1	OSD+MON+MGR
ceph2	DC1	OSD+MON
ceph3	DC1	OSD+MDS+RGW
ceph4	DC2	OSD+MON+MGR
ceph5	DC2	OSD+MON
ceph6	DC2	OSD+MDS+RGW
ceph7	DC3	MON

3.2. 소프트웨어 요구 사항

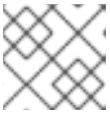
최신 **Red Hat Ceph Storage 5**소프트웨어 버전을 사용합니다.

Red Hat Ceph Storage에서 지원되는 운영 체제 버전에 대한 자세한 내용은 Red Hat Ceph Storage의 기술 자료 문서 : 지원되는 구성 을 참조하십시오.

3.3. 네트워크 설정 요구 사항

권장되는 Red Hat Ceph Storage 구성은 다음과 같습니다.

- 두 개의 개별 네트워크(공용 네트워크 1개와 사설 네트워크 1개)가 있어야 합니다.
- 모든 데이터센터에 대해 Ceph 프라이빗 및 퍼블릭 네트워크의 VLANS 및 서브넷을 지원하는 데이터 센터 3가지가 있어야 합니다.



참고

각 데이터 센터마다 다른 서브넷을 사용할 수 있습니다.

- Red Hat Ceph Storage Object Storage Devices (OSD)를 실행하는 두 데이터센터 사이의 대기 시간은 10ms RTT를 초과할 수 없습니다. 중재 **자 데이터센터** 의 경우 이는 다른 두 개의 OSD 데이터센터에 대해 100ms RTT의 높은 값으로 테스트되었습니다.

다음은 이 가이드에서 사용한 기본 네트워크 구성의 예입니다.

- **DC1: Ceph 공용/사설 네트워크:**10.0.40.0/24
- **DC2: Ceph 공용/사설 네트워크:**10.0.40.0/24
- **DC3: Ceph 공용/사설 네트워크:**10.0.40.0/24

필수 네트워크 환경에 대한 자세한 내용은 Ceph 네트워크 구성을 참조하십시오.

3.4. 노드 사전 배포 요구 사항

Red Hat Ceph Storage 클러스터를 설치하기 전에 다음 단계를 수행하여 필요한 모든 요구 사항을 충족합니다.

1. 모든 노드를 Red Hat Network 또는 Red Hat Satellite에 등록하고 유효한 풀에 등록합니다.

```
subscription-manager register
subscription-manager subscribe --pool=8a8XXXXXX9e0
```

2. 다음 리포지토리에 대해 Ceph 클러스터의 모든 노드에 대한 액세스를 활성화합니다.
   - **rhel-8-for-x86_64-baseos-rpms**
   - **rhel-8-for-x86_64-appstream-rpms**

```
subscription-manager repos --disable="*" --enable="rhel-8-for-x86_64-baseos-rpms" --
enable="rhel-8-for-x86_64-appstream-rpms"
```

3. 운영 체제 RPM을 최신 버전으로 업데이트하고 필요한 경우 재부팅합니다.

```
dnf update -y
reboot
```

4. 부트스트랩 노드가 될 클러스터에서 노드를 선택합니다. 이 예에서 **ceph1** 은 부트스트랩 노드입니다.

부트스트랩 노드 **ceph1** 에서만 **ansible-2.9-for-rhel-8-x86_64-rpms** 및 **rhceph-5-tools-for-rhel-8-x86_64-rpms** 리포지토리를 활성화합니다.

```
subscription-manager repos --enable="ansible-2.9-for-rhel-8-x86_64-rpms" --
enable="rhceph-5-tools-for-rhel-8-x86_64-rpms"
```

5. 모든 호스트의 베어/short 호스트 이름을 사용하여 호스트 이름을 구성합니다.

```
hostnamectl set-hostname <short_name>
```

6. **cephadm**을 사용하여 **Red Hat Ceph Storage**를 배포하기 위한 호스트 이름 구성을 확인합니다.

```
$ hostname
```

출력 예:

```
ceph1
```

7. **/etc/hosts** 파일을 수정하고 **DNS** 도메인 이름으로 **DOMAIN** 변수를 설정하여 **fqdn** 항목을 **127.0.0.1 IP**에 추가합니다.

```
DOMAIN="example.domain.com"

cat <<EOF >/etc/hosts
127.0.0.1 $(hostname).${DOMAIN} $(hostname) localhost localhost.localdomain localhost4
localhost4.localdomain4
::1       $(hostname).${DOMAIN} $(hostname) localhost6 localhost6.localdomain6
EOF
```

8. **hostname -f** 옵션을 사용하여 **fqdn** 이 있는 긴 호스트 이름을 확인합니다.

```
$ hostname -f
```

출력 예:

```
ceph1.example.domain.com
```

참고: 이러한 변경이 필요한 이유에 대해 자세 히 알아보려면 정규화된 도메인 이름과 베어 호스트 이름을 참조하십시오**.**

9. 부트스트랩 노드에서 다음 단계를 실행합니다. 이 예제에서 부트스트랩 노드는 **ceph1** 입니다.

a. **cephadm-ansible RPM** 패키지를 설치합니다.

```
$ sudo dnf install -y cephadm-ansible
```

중요

Ansible 플레이북을 실행하려면 **Red Hat Ceph Storage** 클러스터에 구성된 모든 노드에 **ssh** 암호 없이 액세스할 수 있어야 합니다. 구성된 사용자 **(예: deployment-user)**에 암호 없이 **sudo** 명령을 호출할 수 있는 루트 권한이 있는지 확인합니다.

b. 사용자 지정 키를 사용하려면 선택한 사용자(예: **deployment-user) ssh** 구성 파일을 구성하여 **ssh**를 통해 노드 연결에 사용할 **id/key**를 지정합니다.

```
cat <<EOF > ~/.ssh/config
Host ceph*
   User deployment-user
   IdentityFile ~/.ssh/ceph.pem
EOF
```

c. **ansible** 인벤토리 빌드

```
cat <<EOF > /usr/share/cephadm-ansible/inventory
ceph1
ceph2
ceph3
ceph4
ceph5
ceph6
ceph7
[admin]
ceph1
EOF
```

참고

인벤토리 파일의 **[admin]** 그룹의 일부로 구성된 호스트는 **cephadm** 에서 **_admin** 으로 태그가 지정되어 부트스트랩 프로세스 중에 관리자 **ceph** 인증 키를 수신합니다.

d.

ansible 이 pre-flight 플레이북을 실행하기 전에 ping 모듈을 사용하여 모든 노드에 액세스할 수 있는지 확인합니다.

```
$ ansible -i /usr/share/cephadm-ansible/inventory -m ping all -b
```

출력 예:

```
ceph6 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph5 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph7 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
```

```
    },
    "changed": false,
    "ping": "pong"
}
```

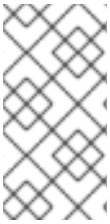
e. 다음 **ansible** 플레이북을 실행합니다.

```
$ ansible-playbook -i /usr/share/cephadm-ansible/inventory /usr/share/cephadm-ansible/cephadm-preflight.yml --extra-vars "ceph_origin=rhcs"
```

preflight Playbook Ansible Playbook은 **Red Hat Ceph Storage dnf** 리포지토리를 구성하고 부트스트랩을 위해 스토리지 클러스터를 준비합니다. **podman, lvm2, chronyd, cephadm**도 설치합니다. **cephadm-ansible** 및 **cephadm-preflight.yml** 의 기본 위치는 **/usr/share/cephadm-ansible** 입니다.

3.5. CEPHADM을 사용한 클러스터 부트스트랩 및 서비스 배포

cephadm 유틸리티는 **cephadm bootstrap** 명령이 실행되는 로컬 노드의 새로운 **Red Hat Ceph Storage** 클러스터에 대해 단일 **Ceph Monitor** 데몬과 **Ceph Manager** 데몬을 설치하고 시작합니다.



참고

부트스트랩 프로세스에 대한 자세한 내용은 새 스토리지 클러스터 부트스트랩 을 참조하십시오.

절차

1. 다음과 같이 **json** 파일을 사용하여 컨테이너 레지스트리에 대해 인증할 **json** 파일을 생성합니다.

```
$ cat <<EOF > /root/registry.json
{
 "url":"registry.redhat.io",
 "username":"User",
 "password":"Pass"
}
EOF
```

2. **RHCS** 클러스터에 노드를 추가하고 다음 표 **3.1**을 실행해야 하는 위치에 대한 특정 레이블을 설정하는 **cluster-spec.yaml** 을 만듭니다.

```
cat <<EOF > /root/cluster-spec.yaml
```

```
service_type: host
addr: 10.0.40.78  ## <XXX.XXX.XXX.XXX>
hostname: ceph1   ##  <ceph-hostname-1>
location:
  root: default
  datacenter: DC1
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: 10.0.40.35
hostname: ceph2
location:
  datacenter: DC1
labels:
  - osd
  - mon
---
service_type: host
addr: 10.0.40.24
hostname: ceph3
location:
  datacenter: DC1
labels:
  - osd
  - mds
  - rgw
---
service_type: host
addr: 10.0.40.185
hostname: ceph4
location:
  root: default
  datacenter: DC2
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: 10.0.40.88
hostname: ceph5
location:
  datacenter: DC2
labels:
  - osd
  - mon
---
service_type: host
addr: 10.0.40.66
hostname: ceph6
location:
  datacenter: DC2
labels:
```

```
  - osd
  - mds
  - rgw
---
service_type: host
addr: 10.0.40.221
hostname: ceph7
labels:
  - mon
---
service_type: mon
placement:
  label: "mon"
---
service_type: mds
service_id: fs_name
placement:
  label: "mds"
---
service_type: mgr
service_name: mgr
placement:
  label: "mgr"
---
service_type: osd
service_id: all-available-devices
service_name: osd.all-available-devices
placement:
  label: "osd"
spec:
  data_devices:
    all: true
---
service_type: rgw
service_id: objectgw
service_name: rgw.objectgw
placement:
  count: 2
  label: "rgw"
spec:
  rgw_frontend_port: 8080
EOF
```

3. **부트 스트랩 노드에서 구성된 RHCS 공용 네트워크를 사용하여 NIC의 IP를 검색합니다. 10.0.40.0 을 ceph 공용 네트워크에 정의한 서브넷이 사용된 후 다음 명령을 실행합니다.**

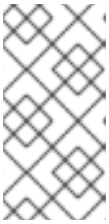
```
$ ip a | grep 10.0.40
```

출력 예:

```
10.0.40.78
```

4.

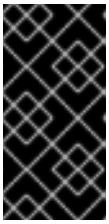
클러스터의 초기 모니터 노드가 될 노드에서 **Cephadm bootstrap** 명령을 **root** 사용자로 실행합니다. **IP_ADDRESS** 옵션은 **cephadm bootstrap** 명령을 실행하는 데 사용하는 노드의 **IP** 주소입니다.



참고

암호 없는 **SSH** 액세스를 위해 **root** 대신 다른 사용자를 구성한 경우 **--ssh-user=** 플래그를 **cepadm bootstrap** 명령과 함께 사용합니다.

```
$ cephadm  bootstrap --ssh-user=deployment-user --mon-ip 10.0.40.78 --apply-spec
/root/cluster-spec.yaml --registry-json /root/registry.json
```



중요

로컬 노드에서 **FQDN(**정규화된 도메인 이름)을 사용하는 경우 명령줄의 **cephadm** 부트스트랩 에 **--allow-fqdn-hostname** 옵션을 추가합니다.

부트스트랩이 완료되면 이전 **cephadm** 부트스트랩 명령의 다음 출력이 표시됩니다.

```
You can access the Ceph CLI with:

 sudo /usr/sbin/cephadm shell --fsid dd77f050-9afe-11ec-a56c-029f8148ea14 -c
/etc/ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring

Please consider enabling telemetry to help improve Ceph:

 ceph telemetry on

For more information see:

 https://docs.ceph.com/docs/pacific/mgr/telemetry/
```

5.

ceph1의 **Ceph CLI** 클라이언트를 사용하여 **Red Hat Ceph Storage** 클러스터 배포 상태를 확인합니다.

```
$ ceph -s
```

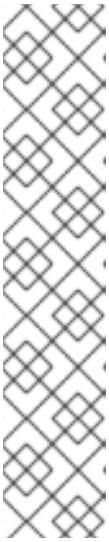
출력 예**:**

```
cluster:
  id:     3a801754-e01f-11ec-b7ab-005056838602
```

```
  health: HEALTH_OK

services:
  mon: 5 daemons, quorum ceph1,ceph2,ceph4,ceph5,ceph7 (age 4m)
  mgr: ceph1.khuuot(active, since 5m), standbys: ceph4.zotfsp
  osd: 12 osds: 12 up (since 3m), 12 in (since 4m)
  rgw: 2 daemons active (2 hosts, 1 zones)

data:
  pools:   5 pools, 107 pgs
  objects: 191 objects, 5.3 KiB
  usage:   105 MiB used, 600 GiB / 600 GiB avail
           105 active+clean
```



참고

모든 서비스를 시작하는 데 몇 분이 걸릴 수 있습니다.

osds가 구성되어 있지 않은 동안 글로벌 복구 이벤트를 얻는 것이 일반적입니다.

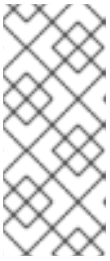
ceph orch ps 및 **ceph orch ls** 를 사용하여 서비스 상태를 추가로 확인할 수 있습니다.

6. 모든 노드가 **cephadm** 클러스터의 일부인지 확인합니다.

```
$ ceph orch host ls
```

출력 예:

```
HOST   ADDR         LABELS  STATUS
ceph1  10.0.40.78   _admin osd mon mgr
ceph2  10.0.40.35   osd mon
ceph3  10.0.40.24   osd mds rgw
ceph4  10.0.40.185  osd mon mgr
ceph5  10.0.40.88   osd mon
ceph6  10.0.40.66   osd mds rgw
ceph7  10.0.40.221  mon
```



참고

ceph1 이 **[admin]** 그룹의 일부로 **cephadm-ansible** 인벤토리에 구성되었으므로 호스트에서 **Ceph** 명령을 직접 실행할 수 있습니다. **Ceph** 관리자 키는 **cephadm** 부트스트랩 프로세스 중에 호스트에 복사되었습니다.

7. 데이터센터에서 **Ceph** 모니터 서비스의 현재 배치를 확인합니다.

```
$ ceph orch ps | grep mon | awk '{print $1 " " $2}'
```

출력 예:

```
mon.ceph1 ceph1
mon.ceph2 ceph2
mon.ceph4 ceph4
mon.ceph5 ceph5
mon.ceph7 ceph7
```

8. 데이터센터에서 **Ceph** 관리자 서비스의 현재 배치를 확인합니다.

```
$ ceph orch ps | grep mgr | awk '{print $1 " " $2}'
```

출력 예:

```
mgr.ceph2.ycgwyz ceph2
mgr.ceph5.kremtt ceph5
```

9. **ceph osd crush** 맵 레이아웃을 확인하여 각 호스트에 **OSD**가 하나씩 구성되어 있고 상태가 **UP** 인지 확인합니다. 또한 테이블 **3.1**에 지정된 대로 각 노드가 올바른 데이터 센터 버킷에 있는지 다시 확인하십시오.

```
$ ceph osd tree
```

출력 예:

```
ID   CLASS  WEIGHT   TYPE NAME          STATUS  REWEIGHT  PRI-AFF
-1         0.87900  root default
-16        0.43950      datacenter DC1
-11        0.14650          host ceph1
```

```
 2   ssd  0.14650          osd.2      up   1.00000  1.00000
-3       0.14650        host ceph2
 3   ssd  0.14650          osd.3      up   1.00000  1.00000
-13      0.14650        host ceph3
 4   ssd  0.14650          osd.4      up   1.00000  1.00000
-17      0.43950     datacenter DC2
-5       0.14650        host ceph4
 0   ssd  0.14650          osd.0      up   1.00000  1.00000
-9       0.14650        host ceph5
 1   ssd  0.14650          osd.1      up   1.00000  1.00000
-7       0.14650        host ceph6
 5   ssd  0.14650          osd.5      up   1.00000  1.00000
```

10. **새 RDB 블록 풀을 생성하고 활성화합니다.**

```
$ ceph osd pool create rbdpool 32 32
$ ceph osd pool application enable rbdpool rbd
```

참고

명령 끝에 있는 숫자 32는 이 풀에 할당된 PG 수입니다. PG의 수는 클러스터의 OSD 수, 풀의 %와 같은 여러 요인에 따라 다를 수 있습니다. 다음 계산기를 사용하여 필요한 PG 수를 확인할 수 있습니다. 풀 계산기당 Ceph PG(배치 그룹) 수를 확인할 수 있습니다.

11. **RBD 풀이 생성되었는지 확인합니다.**

```
$ ceph osd lspools | grep rbdpool
```

출력 예:

```
3 rbdpool
```

12. **MDS 서비스가 활성 상태이며 각 데이터 센터에 하나의 서비스가 있는지 확인합니다.**

```
$ ceph orch ps | grep mds
```

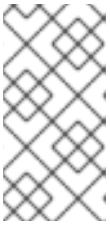
출력 예:

```
mds.cephfs.ceph3.cjpbqo    ceph3              running (17m)   117s ago  17m    16.1M      -
16.2.9
```

```
mds.cephfs.ceph6.lqmgqt  ceph6          running (17m)  117s ago  17m  16.1M  -
16.2.9
```

13. **CephFS 볼륨을 생성합니다.**

```
$ ceph fs volume create cephfs
```



참고

ceph fs volume create 명령은 필요한 데이터 및 메타 **CephFS** 풀도 생성합니다. 자세한 내용은 **Ceph** 파일 시스템 구성 및 마운트 를 참조하십시오.

14. **Ceph** 상태를 확인하여 **MDS** 데몬이 배포된 방법을 확인합니다. **ceph6** 이 이 파일 시스템의 기본 **MDS**인 상태가 활성이고 **ceph3** 이 보조 **MDS**인지 확인합니다.

```
$ ceph fs status
```

출력 예:

```
cephfs - 0 clients
======
RANK  STATE         MDS           ACTIVITY     DNS    INOS   DIRS   CAPS
 0    active  cephfs.ceph6.ggjywj  Reqs:    0 /s    10     13     12      0
       POOL         TYPE     USED  AVAIL
cephfs.cephfs.meta  metadata  96.0k   284G
cephfs.cephfs.data    data       0    284G
   STANDBY MDS
cephfs.ceph3.ogcqkl
```

15. **RGW** 서비스가 활성화되어 있는지 확인합니다.

```
$ ceph orch ps | grep rgw
```

출력 예:

```
rgw.objectgw.ceph3.kkmxgb  ceph3  *:8080       running (7m)     3m ago   7m    52.7M      -
16.2.9
rgw.objectgw.ceph6.xmnpah  ceph6  *:8080       running (7m)     3m ago   7m    53.3M      -
16.2.9
```

4장. RED HAT CEPH STORAGE 확장 클러스터 구성

cephadm 을 사용하여 **Red Hat Ceph Storage** 클러스터를 완전히 배포하면 다음 절차를 사용하여 확장 클러스터 모드를 설정합니다. 새로운 스트레칭 모드는 **2** 사이트 케이스를 처리하도록 설계되었습니다.

절차

1. **ceph mon dump** 명령을 사용하여 모니터에서 사용하는 현재 선택 전략을 확인합니다. 기본적으로 **ceph** 클러스터에서 연결은 클래식으로 설정됩니다.

```
ceph mon dump | grep election_strategy
```

출력 예:

```
dumped monmap epoch 9
election_strategy: 1
```

2. 모니터 선택을 연결로 변경합니다.

```
ceph mon set election_strategy connectivity
```

3. 이전 **ceph mon dump** 명령을 다시 실행하여 **election_strategy** 값을 확인합니다.

```
$ ceph mon dump | grep election_strategy
```

출력 예:

```
dumped monmap epoch 10
election_strategy: 3
```

다양한 선택 전략에 대한 자세한 내용은 **Configuring Monitor election strategy** 을 참조하십시오.

4. 모든 **Ceph** 모니터의 위치를 설정합니다.

```
ceph mon set_location ceph1 datacenter=DC1
ceph mon set_location ceph2 datacenter=DC1
```

```
ceph mon set_location ceph4 datacenter=DC2
ceph mon set_location ceph5 datacenter=DC2
ceph mon set_location ceph7 datacenter=DC3
```

5. 각 모니터에 적절한 위치가 있는지 확인합니다.

```
$ ceph mon dump
```

출력 예:

```
epoch 17
fsid dd77f050-9afe-11ec-a56c-029f8148ea14
last_changed 2022-03-04T07:17:26.913330+0000
created 2022-03-03T14:33:22.957190+0000
min_mon_release 16 (pacific)
election_strategy: 3
0: [v2:10.0.143.78:3300/0,v1:10.0.143.78:6789/0] mon.ceph1; crush_location
{datacenter=DC1}
1: [v2:10.0.155.185:3300/0,v1:10.0.155.185:6789/0] mon.ceph4; crush_location
{datacenter=DC2}
2: [v2:10.0.139.88:3300/0,v1:10.0.139.88:6789/0] mon.ceph5; crush_location
{datacenter=DC2}
3: [v2:10.0.150.221:3300/0,v1:10.0.150.221:6789/0] mon.ceph7; crush_location
{datacenter=DC3}
4: [v2:10.0.155.35:3300/0,v1:10.0.155.35:6789/0] mon.ceph2; crush_location
{datacenter=DC1}
```

6. **crushtool** 명령을 사용하도록 **ceph-base RPM** 패키지를 설치하여 이 **OSD crush** 토폴로지를 사용하는 **NetNamespace** 규칙을 만듭니다.

```
$ dnf -y install ceph-base
```

CRUSH 규칙셋에 대한 자세한 내용은 **Ceph CRUSH** 규칙 세트를 참조하십시오.

7. 클러스터에서 컴파일된 **CRUSH** 맵을 가져옵니다.

```
$ ceph osd getcrushmap > /etc/ceph/crushmap.bin
```

8. **CRUSH** 맵을 컴파일하고 이를 편집할 수 있도록 텍스트 파일로 변환합니다.

```
$ crushtool -d /etc/ceph/crushmap.bin -o /etc/ceph/crushmap.txt
```

9.

파일 끝에 텍스트 파일 /etc/ceph/crushmap.txt 를 편집하여 CRUSH 맵에 다음 규칙을 추가합니다.

```
$ vim /etc/ceph/crushmap.txt
```

```
rule stretch_rule {
        id 1
        type replicated
        min_size 1
        max_size 10
        step take DC1
        step chooseleaf firstn 2 type host
        step emit
        step take DC2
        step chooseleaf firstn 2 type host
        step emit
}

# end crush map
```

참고

규칙 ID 는 고유해야 합니다. 이 예제에서는 id 0과 함께 한 번만 더 크러쉬 규칙을 사용하므로 id 1을 사용합니다. 배포에 더 많은 규칙이 생성된 경우 다음 사용 가능한 ID를 사용합니다.

선언된 CRUSH 규칙에는 다음 정보가 포함됩니다.

- **규칙 이름:**
  - **설명: 규칙을 식별하는 고유한 전체 이름입니다.**
  - **value: stretch_rule**
- **id:**
  - **설명: 규칙을 확인하기 위한 고유한 정수입니다.**

- 값: 1

- 유형:
  - 설명: 복제된 스토리지 드라이브 또는 삭제로 코드된 규칙에 대해 설명합니다.
  - 값: replicated
- min_size:
  - 설명: 풀이 이 수보다 복제본 수를 줄이는 경우 CRUSH는 이 규칙을 선택하지 않습니다.
  - 값: 1
- max_size:
  - 설명: 풀이 이 수보다 더 많은 복제본을 만드는 경우 CRUSH는 이 규칙을 선택하지 않습니다.
  - 값: 10
- DC1을 선택합니다.
  - 설명: 버킷 이름(DC1)을 가져와서 트리를 반복하기 시작합니다.
- Step chooseleaf firstn 2 type host
  - 설명: 지정된 유형의 버킷 수를 선택합니다. 이 경우 DC1에 있는 두 개의 서로 다른 호스트입니다.

- **Step emit**
  - 설명: 현재 값을 출력하고 스택을 선점합니다. 일반적으로 규칙의 끝에 사용되지만 동일한 규칙의 다른 트리에서 선택하는 데 사용할 수도 있습니다.
- **DC2 단계**
  - 설명: 버킷 이름(DC2)을 가져와서 트리를 반복하기 시작합니다.
- **Step chooseleaf firstn 2 type host**
  - 설명: 지정된 유형의 버킷 수를 선택합니다. 이 경우 DC2에 있는 두 개의 서로 다른 호스트입니다.
- **Step emit**
  - 설명: 현재 값을 출력하고 스택을 선점합니다. 일반적으로 규칙의 끝에 사용되지만 동일한 규칙의 다른 트리에서 선택하는 데 사용할 수도 있습니다.

10. /etc/ceph/crushmap.txt 파일에서 새 CRUSH 맵을 컴파일하고 /etc/ceph/crushmap2.bin 이라는 바이너리 파일로 변환합니다.

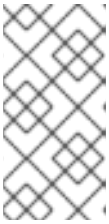
```
$ crushtool -c /etc/ceph/crushmap.txt -o /etc/ceph/crushmap2.bin
```

11. 클러스터에 다시 생성한 새 crushmap을 삽입합니다.

```
$ ceph osd setcrushmap -i /etc/ceph/crushmap2.bin
```

출력 예:

```
17
```



참고

숫자 **17**은 카운터이며 크러쉬 맵에 대한 변경 사항에 따라 **(18,19** 등**)** 증가합니다**.**

12.

생성된 확장 규칙을 이제 사용할 수 있는지 확인합니다**.**

```
ceph osd crush rule ls
```

출력 예**:**

```
replicated_rule
stretch_rule
```

13.

확장 클러스터 모드를 활성화합니다**.**

```
$ ceph mon enable_stretch_mode ceph7 stretch_rule datacenter
```

이 예에서 **ceph7** 은 **arbiter** 노드이고**, stretch_rule** 은 이전 단계에서 생성한 크러쉬 규칙이며 데이터 센터는 분할 버킷입니다**.**

14.

모든 풀이 **Ceph** 클러스터에서 생성한 **stretch_rule CRUSH** 규칙을 사용하고 있는지 확인합니다**.**

```
$ for pool in $(rados lspools);do echo -n "Pool: ${pool}; ";ceph osd pool get ${pool}
crush_rule;done
```

출력 예**:**

```
Pool: device_health_metrics; crush_rule: stretch_rule
Pool: cephfs.cephfs.meta; crush_rule: stretch_rule
Pool: cephfs.cephfs.data; crush_rule: stretch_rule
Pool: .rgw.root; crush_rule: stretch_rule
Pool: default.rgw.log; crush_rule: stretch_rule
Pool: default.rgw.control; crush_rule: stretch_rule
Pool: default.rgw.meta; crush_rule: stretch_rule
Pool: rbdpool; crush_rule: stretch_rule
```

이는 작동하는 **Red Hat Ceph Storage**가 **arbiter** 모드로 확장되는 클러스터를 사용할 수 있음

을 나타냅니다.

5장. 관리형 클러스터에 OPENSHIFT DATA FOUNDATION 설치

두 **OpenShift Container Platform** 클러스터 간에 스토리지 복제를 구성하려면 다음과 같이 각 관리 클러스터에 **OpenShift Data Foundation**을 먼저 설치해야 합니다.

1. 각 관리형 클러스터에 최신 **OpenShift Data Foundation**을 설치합니다.

2. **Operator**를 설치한 후 외부 스토리지 플랫폼과 함께 **Connect** 옵션을 사용하여 **StorageSystem**을 만듭니다.

   자세한 내용은 **Deploying OpenShift Data foundation in external mode** 를 참조하십시오.

3. **OpenShift Data foundation**의 성공적인 배포를 검증합니다.

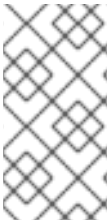
   a. 다음 명령을 사용하여 각 관리 클러스터에서 다음을 수행합니다.

   ```
   $ oc get storagecluster -n openshift-storage ocs-external-storagecluster -o
   jsonpath='{.status.phase}{"\n"}'
   ```

   b. **MCG(Multicloud Gateway)**의 경우:

   ```
   $ oc get noobaa -n openshift-storage noobaa -o jsonpath='{.status.phase}{"\n"}'
   ```

상태 결과가 **Primary** 관리 클러스터 및 보조 관리 클러스터 의 쿼리 모두에 **Ready** 인 경우 다음 단계를 계속합니다.



참고

OpenShift Data Foundation의 성공적인 설치는 **OpenShift Container Platform** 웹 콘솔에서 **Storage** (스토리지)로 이동한 다음 **Data Foundation.**

6장. HUB 클러스터에 OPENSHIFT DR HUB OPERATOR 설치

절차

1. **Hub** 클러스터에서 **OperatorHub**로 이동하여 **OpenShift DR Hub Operator** 에 대한 검색 필터를 사용합니다.

2. 화면 지시에 따라 **openshift-dr-system** 프로젝트에 **Operator**를 설치합니다.

3. 다음 명령을 사용하여 **Operator Pod**가 **Running** 상태인지 확인합니다.

```
$ oc get pods -n openshift-dr-system
```

출력 예:

```
NAME                                    READY   STATUS    RESTARTS   AGE
ramen-hub-operator-898c5989b-96k65   2/2     Running   0          4m14s
```

7장. 관리형 및 HUB 클러스터 구성

7.1. S3 끝점 간 SSL 액세스 구성

보안 전송 프로토콜을 사용하여 MCG 오브젝트 버킷의 대체 클러스터에 메타데이터를 저장할 수 있도록 s3 끝점 간 네트워크(SSL) 액세스를 구성합니다. 또한 Hub 클러스터 는 개체 버킷에 대한 액세스를 확인해야 합니다.

참고

환경에 대해 서명된 유효한 인증서 세트를 사용하여 모든 OpenShift 클러스터를 배포하는 경우 이 섹션을 건너뛸 수 있습니다.

절차

1. 기본 관리 클러스터의 수신 인증서를 추출하고 출력을 primary.crt 에 저장합니다.

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > primary.crt
```

2. Secondary 관리 클러스터의 수신 인증서를 추출하고 출력을 secondary.crt 에 저장합니다.

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > secondary.crt
```

3. 기본 관리 클러스터 ,Secondary 관리 클러스터 , Hub 클러스터의 파일 이름 cm-clusters-crt.yaml 로 원격 클러스터 의 인증서 번들을 보관할 새 ConfigMap 을 만듭니다.

참고

이 예제 파일에 표시된 대로 각 클러스터에 대해 3개 이상의 인증서가 있을 수 있습니다. 또한 이전에 생성된 primary.crt 및 secondary.crt 파일에서 복사하여 붙여넣은 후 인증서 콘텐츠의 들여쓰기가 올바른지 확인합니다.

```
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from primary.crt here>
    -----END CERTIFICATE-----
```

```
    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from primary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 primary.crt here>
    -----END CERTIFICATE----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 from secondary.crt here>
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: openshift-config
```

4. **기본 관리 클러스터, 보조 관리 클러스터 , Hub 클러스터에 ConfigMap 파일을 만듭니다.**

```
$ oc create -f cm-clusters-crt.yaml
```

출력 예:

```
configmap/user-ca-bundle created
```

중요

Hub 클러스터에서 DRPolicy 리소스를 사용하여 오브젝트 버킷에 대한 액세스를 확인하려면 Hub 클러스터에서 동일한 ConfigMap cm-clusters-crt.yaml 도 생성해야 합니다.

5. **기본 관리 클러스터,보조 관리 클러스터, Hub 클러스터에서 기본 프록시 리소스를 패치합니다.**

```
$ oc patch proxy cluster --type=merge  --patch='{"spec":{"trustedCA":{"name":"user-ca-bundle"}}}'
```

출력 예:

```
proxy.config.openshift.io/cluster patched
```

7.2. 오브젝트 버킷 및 S3STOREPROFILE 생성

OpenShift DR에서는 관리형 클러스터에서 워크로드 관련 클러스터 데이터를 저장하고 장애 조치(failover) 또는 재배치 작업 중 워크로드 복구를 오케스트레이션하기 위해 S3 저장소를 필요로 합니다. 이러한 지침은 MCG(Multicloud Object Gateway)를 사용하여 필요한 오브젝트 버킷을 생성하는 데 적용할 수 있습니다. MCG는 이미 OpenShift Data Foundation을 설치함으로써 설치되어 있어야 합니다.

절차

1. 기본 및 보조 관리 클러스터에서 영구 볼륨 메타데이터를 저장하는 데 사용할 MCG 개체 버킷 또는 OBC를 생성합니다.

   a. 다음 YAML 파일을 파일 이름 odrbucket.yaml 에 복사합니다.

   ```
   apiVersion: objectbucket.io/v1alpha1
   kind: ObjectBucketClaim
   metadata:
     name: odrbucket
     namespace: openshift-storage
   spec:
     generateBucketName: "odrbucket"
     storageClassName: openshift-storage.noobaa.io
   ```

   b. 기본 관리 클러스터와 보조 관리 클러스터 모두에 MCG 버킷 odrbucket 을 만듭니다.

   ```
   $ oc create -f odrbucket.yaml
   ```

   출력 예:

   ```
   objectbucketclaim.objectbucket.io/odrbucket created
   ```

2. 다음 명령을 사용하여 각 관리 클러스터의 odrbucket OBC 액세스 키를 *base-64 인코딩* 값으로 추출합니다.

   ```
   $ oc get secret odrbucket -n openshift-storage -o jsonpath='{.data.AWS_ACCESS_KEY_ID}
   {"\n"}'
   ```

-

출력 예:

```
cFpIYTZWN1NhemJjbEUyWlpwN1E=
```

3. 다음 명령을 사용하여 각 관리 클러스터의 **odrbucket OBC** 시크릿 키를 ***base-64 인코딩*** 값으로 추출합니다.

```
$ oc get secret odrbucket -n openshift-storage -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}{"\n"}'
```

출력 예:

```
V1hUSnMzZUoxMHRRTXdGMU9jQXRmUlAyMmd5bGwwYjNvMHprZVhtNw==
```

중요

기본 관리 클러스터 및 보조 관리 클러스터에서 **odrbucket OBC** 에 대해 액세스 키와 시크릿 키를 검색해야 합니다.

7.3. MULTICLOUD OBJECT GATEWAY 오브젝트 버킷에 대한 S3 시크릿 생성

이제 이전 섹션의 오브젝트 버킷에 필요한 정보가 추출되었으므로 **Hub** 클러스터에 생성된 새로운 시크릿 이 있어야 합니다. 이러한 새 보안에서는 **Hub** 클러스터의 두 관리 클러스터 모두에 대해 **MCG** 오브젝트 버킷 액세스 키와 시크릿 키를 저장합니다.

절차

1. 기본 관리 클러스터 의 다음 **S3** 시크릿 **YAML** 형식을 파일 이름 **odr-s3secret-column.yaml** 에 복사합니다.

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: <primary cluster base-64 encoded access key>
  AWS_SECRET_ACCESS_KEY: <primary cluster base-64 encoded secret access key>
kind: Secret
metadata:
  name: odr-s3secret-primary
  namespace: openshift-dr-system
```

2. **Hub** 클러스터에 이 시크릿을 생성합니다.

```
$ oc create -f odr-s3secret-primary.yaml
```

출력 예:

```
secret/odr-s3secret-primary created
```

3. 보조 관리 클러스터에 대한 다음 **S3** 시크릿 **YAML** 형식을 파일 이름 **odr-s3secret-secondary.yaml** 에 복사합니다.

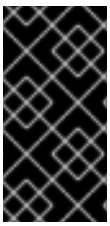
```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: <secondary cluster base-64 encoded access key>
  AWS_SECRET_ACCESS_KEY: <secondary cluster base-64 encoded secret access key>
kind: Secret
metadata:
  name: odr-s3secret-secondary
  namespace: openshift-dr-system
```

4. **Hub** 클러스터에 이 시크릿을 생성합니다.

```
$ oc create -f odr-s3secret-secondary.yaml
```

출력 예:

```
secret/odr-s3secret-secondary created
```



중요

액세스 키 및 시크릿 키 값은 **base-64**로 인코딩 되어야 합니다. 키의 인코딩된 값은 이전 섹션에서 검색되었습니다.

7.4. OPENSHIFT DR HUB OPERATOR S3STOREPROFILES 구성

MCG의 **s3 compatibleibleEndpoint** 또는 경로를 찾으려면 **Primary** 관리 클러스터 및 **Secondary** 관리 클러스터에서 다음 명령을 실행합니다.

절차

a.

다음 명령을 사용하여 각 관리 클러스터에서 외부 **S 3 endpoint s3 CompatibilityibleEndpoint** 또는 **route for MCG**를 검색합니다.

```
$ oc get route s3 -n openshift-storage -o jsonpath --template="https://{.spec.host}{'\n'}"
```

출력 예:

```
https://s3-openshift-storage.apps.perf1.example.com
```

중요

기본 관리 클러스터 및 보조 관리 클러스터 모두에 대해 각각 **s3 CompatibilityibleEndpoint** 경로 또는 **s3-openshift-storage.apps. <baseDomain >** 및 **s3-openshift-storage.apps.<secondary clusterID>. <baseDomain >**을 검색해야 합니다.

b.

odrbucket OBC 정확한 버킷 이름을 검색합니다.

```
$ oc get configmap odrbucket -n openshift-storage -o jsonpath='{.data.BUCKET_NAME}
{"\n"}'
```

출력 예:

```
odrbucket-2f2d44e4-59cb-4577-b303-7219be809dcd
```

중요

고유한 **s3Bucket** 이름 ***odrbucket-<your value1 >*** 및 ***odrbucket-<your value2 >***를 각각 기본 관리 클러스터 와 보조 관리 클러스터에서 검색해야 합니다.

c.

새 콘텐츠를 추가하도록 **Hub** 클러스터에서 **ConfigMap ramen-hub-operator-config** 를 수정합니다.

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

d.

s3StoreProfiles 에서 시작하는 다음 새 콘텐츠를 Hub 클러스터 의 ConfigMap에만 추가합니다.

```
[...]
data:
  ramen_manager_config.yaml: |
    apiVersion: ramendr.openshift.io/v1alpha1
    kind: RamenConfig
[...]
    ramenControllerType: "dr-hub"
    ### Start of new content to be added
    s3StoreProfiles:
    - s3ProfileName: s3-primary
      s3CompatibleEndpoint: https://s3-openshift-storage.apps.<primary clusterID>.
<baseDomain>
      s3Region: primary
      s3Bucket: odrbucket-<your value1>
      s3SecretRef:
        name: odr-s3secret-primary
        namespace: openshift-dr-system
    - s3ProfileName: s3-secondary
      s3CompatibleEndpoint: https://s3-openshift-storage.apps.<secondary clusterID>.
<baseDomain>
      s3Region: secondary
      s3Bucket: odrbucket-<your value2>
      s3SecretRef:
        name: odr-s3secret-secondary
        namespace: openshift-dr-system
[...]
```

8장. HUB 클러스터에서 재해 복구 정책 생성

OpenShift DR은 RHACM 허브 클러스터에서 Dis broken recovery Policy(DRPolicy) 리소스(클러스터 범위)를 사용하여 관리형 클러스터에 워크로드를 배포, 장애 조치, 재배치합니다.

사전 요구 사항

- 두 클러스터 세트가 있는지 확인합니다.
- 정책의 각 클러스터에 OpenShift DR 클러스터 및 허브 Operator의 ConfigMap을 사용하여 구성된 S3 프로필 이름이 할당되어 있는지 확인합니다.

절차

1. Hub 클러스터에서 openshift-dr-system 프로젝트에서 설치된 Operator로 이동하여 OpenShift DR Hub Operator 를 클릭합니다. DRPolicy 및 DRPlacementControl 의 사용 가능한 API 두 개가 표시됩니다.
2. DRPolicy에 대한 인스턴스 생성 을 클릭하고 YAML 보기를 클릭합니다.
3. <cluster1> 및 <cluster2>를 RHACM 에서 관리 클러스터의 올바른 이름으로 교체한 후 drpolicy.yaml 파일 이름에 다음 YAML을 저장합니다. <string_value>를 임의의 값(예: metro)으로 바꿉니다.

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPolicy
metadata:
  name: odr-policy
spec:
  drClusterSet:
  - name: <cluster1>
    region: <string_value>
    s3ProfileName: s3-primary
    clusterFence: Unfenced
  - name: <cluster2>
    region: <string_value>
    s3ProfileName: s3-secondary
    clusterFence: Unfenced
```

참고

DRPolicy는 클러스터 범위 리소스이므로 이 리소스를 생성하기 위해 네임스페이스를 지정할 필요가 없습니다.

4. 고유한 **drpolicy.yaml** 파일의 콘텐츠를 **YAML** 보기에 복사합니다. 원본 콘텐츠를 완전히 교체해야 합니다.

5. **YAML** 보기 화면에서 생성 을 클릭합니다.

6. **DRPolicy** 가 성공적으로 생성되고, 이전에 생성된 **Secrets**를 사용하여 **MCG** 오브젝트 버킷에 액세스할 수 있는지 확인하려면 **Hub** 클러스터에서 다음 명령을 실행합니다.

```
$ oc get drpolicy odr-policy -n openshift-dr-system -o jsonpath='{.status.conditions[].reason}
{"\n"}'
```

출력 예:

```
Succeeded
```

9장. OPENSHIFT DR 클러스터 OPERATOR 자동 설치 활성화

DRPolicy가 성공적으로 생성되면 **OpenShift DR Cluster Operator** 를 **openshift-dr-system** 네임스페이스의 기본 관리 클러스터와 보조 관리 클러스터에 설치할 수 있습니다.

절차

1. **Hub** 클러스터에서 **ConfigMag ramen-hub-operator-config** 를 편집하고 다음과 같이 **deploymentAutomationEnabled=false** 값을 **true** 로 수정합니다.

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

```
apiVersion: v1
data:
  ramen_manager_config.yaml: |
  [...]
    drClusterOperator:
      deploymentAutomationEnabled: true  ## <-- Change value to "true" if it is set to "false"
      channelName: stable-4.10
      packageName: odr-cluster-operator
      namespaceName: openshift-dr-system
      catalogSourceName: redhat-operators
      catalogSourceNamespaceName: openshift-marketplace
      clusterServiceVersionName: odr-cluster-operator.v4.10.0
[...]
```

2. 기본 관리 클러스터에서 설치가 성공적으로 수행되었으며 보조 관리 클러스터에서 다음 명령을 수행하는지 확인합니다.

```
$ oc get csv,pod -n openshift-dr-system
```

출력 예:

```
NAME                                                        DISPLAY                       VERSION
REPLACES   PHASE
clusterserviceversion.operators.coreos.com/odr-cluster-operator.v4.10.0   Openshift DR
Cluster Operator   4.10.0              Succeeded

NAME                                       READY   STATUS    RESTARTS   AGE
pod/ramen-dr-cluster-operator-5564f9d669-f6lbc   2/2     Running   0          5m32s
```

각 관리 클러스터에서 **OperatorHub**로 이동하여 **OpenShift DR Cluster Operator** 가 설치되었는지 확인할 수도 있습니다.

10장. S3SECRETS를 관리된 클러스터로 자동 전송 활성화

s3Secrets를 필수 **OpenShift DR** 클러스터 구성 요소로 자동 전송할 수 있도록 하려면 다음 절차를 따르십시오**. OpenShift DR** 구성 맵의 **s3Profiles**에 액세스하는 데 필요한 **s3Secrets**로 **OpenShift DR** 클러스터 네임스페이스를 업데이트합니다**.**

절차

1. 다음과 같이 **Hub** 클러스터에서 **ConfigMag ramen-hub-operator-config** 를 편집하여 **s3SecretDistributionEnabled=true** 를 추가합니다**.**

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

```
apiVersion: v1
data:
  ramen_manager_config.yaml: |
    apiVersion: ramendr.openshift.io/v1alpha1
    drClusterOperator:
      deploymentAutomationEnabled: true
      s3SecretDistributionEnabled: true  ## <-- Add to enable automatic transfer of s3secrets
      catalogSourceName: redhat-operators
      catalogSourceNamespaceName: openshift-marketplace
      channelName: stable-4.10
      clusterServiceVersionName: odr-cluster-operator.v4.10.0
      namespaceName: openshift-dr-system
      packageName: odr-cluster-operator
[...]
```

2. 두 관리 클러스터에서 이 명령을 실행하여 시크릿 전송에 성공했는지 확인합니다**.**

```
$ oc get secrets -n openshift-dr-system | grep Opaque
```

출력 예**:**

```
8b3fb9ed90f66808d988c7edfa76eba35647092   Opaque       2      11m
af5f82f21f8f77faf3de2553e223b535002e480   Opaque       2      11m
```

11장. 샘플 애플리케이션 생성

기본 관리 클러스터에서 보조 관리 클러스터로 장애 조치를 테스트하고 다시 간단한 애플리케이션이 필요합니다. 예제로 **busybox** 라는 샘플 애플리케이션을 사용합니다.

절차

1. **busybox** 샘플 애플리케이션의 **Hub** 클러스터에 네임스페이스 또는 프로젝트를 생성합니다.

```
$ oc new-project busybox-sample
```

참고

busybox-sample 이외의 다른 프로젝트 이름을 원하는 경우 사용할 수 있습니다. **Advanced Cluster Manager** 콘솔을 통해 샘플 애플리케이션을 배포하여 이 단계에서 생성된 것과 동일한 프로젝트 이름을 사용해야 합니다.

2. **DRPlacementControl** 리소스 생성

DRPlacementControl은 **OpenShift DR Hub Operator**가 **Hub** 클러스터에 설치된 후 사용 가능한 **API**입니다. **DRPolicy**에 속하는 클러스터 전체의 데이터 가용성에 따라 배치 결정을 오케스트레이션하는 **Advanced Cluster Manager PlacementRule** 조정기입니다.

a. **Hub** 클러스터에서 **busybox-sample** 프로젝트에서 **Installed Operators**로 이동하여 **OpenShift DR Hub Operator** 를 클릭합니다. 사용 가능한 **API** 두 개, **DRPolicy** 및 **DRPlacementControl**이 표시됩니다.

b. **DRPlacementControl** 에 대한 인스턴스를 생성한 다음 **YAML** 보기로 이동합니다. **busybox-sample** 프로젝트가 선택되어 있는지 확인합니다.

c. **Advanced Cluster Manager**의 관리 클러스터의 **< *cluster1* >**을 올바른 이름으로 교체한 후 파일 이름 **busybox-drpc.yaml** 에 다음 **YAML**을 복사하고 저장합니다.

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPlacementControl
metadata:
  labels:
    app: busybox-sample
  name: busybox-drpc
```

```
spec:
  drPolicyRef:
    name: odr-policy
  placementRef:
    kind: PlacementRule
    name: busybox-placement
  preferredCluster: <cluster1>
  pvcSelector:
    matchLabels:
      appname: busybox
```

d. 고유한 **busybox-drpc.yaml** 파일의 콘텐츠를 **YAML** 보기(원래 콘텐츠 교체)로 복사합니다.

e. **YAML** 보기 화면에서 생성 을 클릭합니다.

다음 **CLI** 명령을 사용하여 이 리소스를 생성할 수도 있습니다.

```
$ oc create -f busybox-drpc.yaml -n busybox-sample
```

출력 예:

```
drplacementcontrol.ramendr.openshift.io/busybox-drpc created
```

중요

이 리소스는 **busybox-sample** 네임 스페이스(또는 이전에 만든 네임스페이스)에서 생성해야 합니다.

3. 리소스 템플릿을 배포할 수 있는 대상 클러스터를 정의하는 배치 규칙 리소스를 생성합니다. 배치 규칙을 사용하여 애플리케이션의 다중 클러스터 배포를 용이하게 합니다.

a. 다음 **YAML**을 복사하여 파일 이름 **busybox-placementrule.yaml** 에 저장합니다.

```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  labels:
    app: busybox-sample
  name: busybox-placement
spec:
```

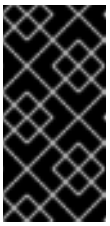
```
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterReplicas: 1
  schedulerName: ramen
```

b. **busybox-sample** 애플리케이션에 대한 배치 규칙 리소스를 생성합니다.

```
$ oc create -f busybox-placementrule.yaml -n busybox-sample
```

출력 예:

```
placementrule.apps.open-cluster-management.io/busybox-placement created
```



중요

이 리소스는 **busybox-sample** 네임 스페이스(또는 이전에 만든 네임스페이스)에서 생성해야 합니다.

4. **RHACM** 콘솔을 사용하여 샘플 애플리케이션 생성

a. 아직 로그인하지 않은 경우 **OpenShift** 인증 정보를 사용하여 **RHACM** 콘솔에 로그인합니다.

```
$ oc get route multicloud-console -n open-cluster-management -o jsonpath --
template="https://{.spec.host}/multicloud/applications{'\n'}"
```

출력 예:

```
https://multicloud-console.apps.perf3.example.com/multicloud/applications
```

b. **Applications** (애플리케이션)로 이동하여 **Create application** 을 클릭합니다.

c. **type**을 **Subscription** 으로 선택합니다.

d. 애플리케이션 이름 (예: **busybox**) 및 네임스페이스 (예: **busybox-sample**)를 입력합니다.

e. **Repository location for resources** 섹션에서 **Repository type Git** 을 선택합니다.

f. 샘플 애플리케이션의 **Git** 리포지토리 **URL, github Branch** 및 **Path** 를 입력합니다. 여기서 리소스 **busybox Pod** 및 **PVC**가 생성됩니다.

샘플 애플리케이션 리포지토리를 **https://github.com/RamenDR/ocm-ramen-samples** 로, 분기 가 기본 이고 경로 는 **busybox-odr-metro** 입니다.

g. 양식을 아래로 스크롤하여 배포할 클러스터 선택 섹션까지 아래로 스크롤하고 기존 배치 구성 선택을 클릭합니다.

h. 드롭다운 목록에서 기존 배치 규칙 (예: **busybox-placement**)을 선택합니다.

i. 저장을 클릭합니다.

후속 화면에서 아래쪽으로 스크롤합니다.**On the follow-on screen scroll to the bottom.** 애플리케이션 토폴로지에 모든 그린 확인 표시가 있는지 확인해야 합니다.

참고

자세한 내용을 보려면 토폴로지 요소 중 하나를 클릭하면 토폴로지 보기 오른쪽에 창이 표시됩니다.

5. 샘플 애플리케이션 배포 및 복제 검증.

이제 **busybox** 애플리케이션이 기본 클러스터(**DRPlacementControl**에 지정)에 배포되었으므로 배포를 검증할 수 있습니다.

a. **RHACM**에서 **busybox** 를 배포한 관리형 클러스터에 로그인합니다.

```
$ oc get pods,pvc -n busybox-sample
```

출력 예:

```
NAME        READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          6m

NAME                                 STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS                AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-a56c138a-a1a9-4465-927f-
af02afbbff37   1Gi        RWO            ocs-storagecluster-ceph-rbd   6m
```

b. **busybox PVC**에 대한 복제 리소스도 생성되었는지 확인합니다.

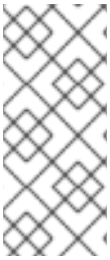
```
$ oc get volumereplicationgroup -n busybox-sample
```

출력 예:

```
NAME                                          AGE
volumereplicationgroup.ramendr.openshift.io/busybox-drpc   6m
```

11.1. 샘플 애플리케이션 삭제

RHACM 콘솔을 사용하여 샘플 애플리케이션 **busybox** 를 삭제할 수 있습니다.



참고

샘플 애플리케이션을 삭제하는 지침은 장애 조치(**failover**) 및 실패(**relocate**) 테스트가 완료되고 애플리케이션이 **RHACM** 및 관리되는 클러스터에서 제거할 준비가 될 때까지 실행되지 않아야 합니다.

절차

1. **RHACM** 콘솔에서 **Applications** (애플리케이션) .
2. 삭제할 샘플 애플리케이션을 검색합니다(예: **busybox**).
3. 삭제하려는 애플리케이션 옆에 있는 **Action Menu**(작업 메뉴) 를 클릭합니다.

4. 애플리케이션 삭제를 클릭합니다.

Delete application(애플리케이션 삭제)을 선택하면 애플리케이션 관련 리소스도 삭제해야 하는지 묻는 새 화면이 표시됩니다.

5. 애플리케이션 관련 리소스 제거 확인란을 선택하여 서브스크립션 및 PlacementRule을 삭제합니다.

6. 삭제를 클릭합니다. 이렇게 하면 기본 관리 클러스터(또는 애플리케이션이 실행 중인 클러스터)에서 **busybox** 애플리케이션이 삭제됩니다.

7. RHACM 콘솔을 사용하여 삭제된 리소스 외에도 DRPlacementControl 도 busybox 애플리케이션을 삭제한 직후 삭제해야 합니다.

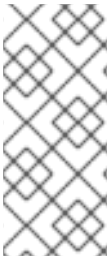
   a. Hub 클러스터의 OpenShift 웹 콘솔에 로그인하고 busybox-sample 프로젝트에 대해 설치된 Operator로 이동합니다.

   b. OpenShift DR Hub Operator 를 클릭한 다음 DRPlacementControl 탭을 클릭합니다.

   c. 삭제하려는 busybox 애플리케이션 DRPlacementControl 옆에 있는 Action Menu(및 작업 메뉴) 를 클릭합니다.

   d. DRPlacementControl 삭제를 클릭합니다.

   e. 삭제를 클릭합니다.



참고

이 프로세스를 사용하여 DRPlacementControl 리소스가 있는 모든 애플리케이션을 삭제할 수 있습니다. CLI를 사용하여 애플리케이션 네임스페이스에서 DRPlacementControl 리소스를 삭제할 수도 있습니다.

12장. 관리형 클러스터 간 애플리케이션 장애 조치

이 섹션에서는 **busybox** 샘플 애플리케이션을 장애 조치하는 방법에 대한 지침을 제공합니다. **Metro-DR**의 장애 조치 방법은 애플리케이션을 기반으로 합니다. 이러한 방식으로 보호되도록 하는 각 애플리케이션은 **DR** 테스트용 샘플 애플리케이션 생성 섹션에 표시된 대로 애플리케이션 네임스페이스에 생성된 해당 **DRPlacementControl** 리소스 및 **PlacementRule** 리소스가 있어야 합니다.

절차

1. **NetworkFence** 리소스를 생성하고 펜싱을 활성화합니다.

네트워크 펜싱 작업을 수행할 **CIDR** 블록 또는 **IP** 주소 목록을 지정합니다. 이 경우 외부 **RHCS** 클러스터를 사용하여 펜싱해야 하는 클러스터의 모든 **OpenShift** 노드의 **EXTERNAL-IP**입니다.

a. 이 명령을 실행하여 기본 관리 클러스터 의 **IP** 주소를 가져옵니다.

```
$ oc get nodes -o jsonpath='{range .items[*]}{.status.addresses[?
(@.type=="ExternalIP")].address}{"\n"}{end}'
```

출력 예:

```
10.70.56.118
10.70.56.193
10.70.56.154
10.70.56.242
10.70.56.136
10.70.56.99
```

참고

사이트 중단이 발생하기 전에 모든 **OpenShift** 노드의 현재 **IP** 주소를 수집합니다. 가장 좋은 방법은 **NetworkFence YAML** 파일을 생성하고 재해 복구 이벤트를 위해 사용 가능하고 최신 상태로 유지하는 것입니다.

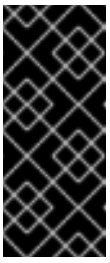
모든 노드의 **IP** 주소는 아래 표시된 대로 **NetworkFence** 예제 리소스에 추가됩니다. 이 예는 **6**개의 노드이지만 클러스터에 더 많은 노드가 있을 수 있습니다.

```
apiVersion: csiaddons.openshift.io/v1alpha1
kind: NetworkFence
metadata:
```

```
  name: network-fence-<cluster1>
spec:
  driver: openshift-storage.rbd.csi.ceph.com
  cidrs:
    - <IP_Address1>/32
    - <IP_Address2>/32
    - <IP_Address3>/32
    - <IP_Address4>/32
    - <IP_Address5>/32
    - <IP_Address6>/32
    [...]
  secret:
    name: rook-csi-rbd-provisioner
    namespace: openshift-storage
  parameters:
    clusterID: openshift-storage
```

b.

위의 **YAML** 파일 예제의 경우 **IP** 주소를 수정하고 기본 관리 *클러스터의 경우 **RHACM** 에 있는 클러스터 이름으로 올바른* ***<cluster1>*** 을 제공합니다. 이 파일을 파일 이름 **network-fence-<cluster1>.yaml** 에 저장합니다.



중요

NetworkFence 는 장애 조치 전에 애플리케이션이 현재 실행 중인 반대의 관리형 클러스터에서 생성해야 합니다. 이 경우 이 클러스터는 보조 관리 클러스터 입니다.

```
$ oc create -f network-fence-<cluster1>.yaml
```

출력 예:

```
networkfences.csiaddons.openshift.io/network-fence-ocp4perf1 created
```

중요

NetworkFence 가 생성되면 애플리케이션에서 **OpenShift Data Foundation** 스토리지로의 모든 통신이 실패하고 일부 **Pod**는 이제 펜싱되는 클러스터의 비정상적인 상태(예: **CreateContainerError, CrashLoopBackOff**)에 있습니다.

c.

NetworkFence 가 생성된 위치와 동일한 클러스터에서 상태가 성공인지 확인합니다. **<cluster1>**이 올바르도록 수정합니다.

```
export NETWORKFENCE=network-fence-<cluster1>
oc get networkfences.csiaddons.openshift.io/$NETWORKFENCE -n openshift-dr-system
-o jsonpath='{.status.result}{"\n"}'
```

출력 예:

```
Succeeded
```

2. **펜싱된 클러스터의 DRPolicy 를 수정합니다.**

a. **Hub 클러스터에서 DRPolicy 를 편집하고 < *cluster1* >(예: ocp4perf1)을 Unfenced 에서 ManuallyFenced 로 변경합니다.**

```
$ oc edit drpolicy odr-policy
```

출력 예:

```
[...]
spec:
  drClusterSet:
  - clusterFence: ManuallyFenced  ## <-- Modify from Unfenced to ManuallyFenced
    name: ocp4perf1
    region: metro
    s3ProfileName: s3-primary
  - clusterFence: Unfenced
    name: ocp4perf2
    region: metro
    s3ProfileName: s3-secondary
[...]
```

출력 예:

```
drpolicy.ramendr.openshift.io/odr-policy edited
```

b. **Hub 클러스터에서 DRPolicy 상태가 기본 관리형 클러스터 의 Fenced 로 변경되었는지 확인합니다.**

```
$ oc get drpolicies.ramendr.openshift.io odr-policy -o yaml | grep -A 6 drClusters
```

출력 예:

```
drClusters:
  ocp4perf1:
    status: Fenced
    string: ocp4perf1
  ocp4perf2:
    status: Unfenced
    string: ocp4perf2
```

3. **장애조치로 DRPlacementControl 수정**

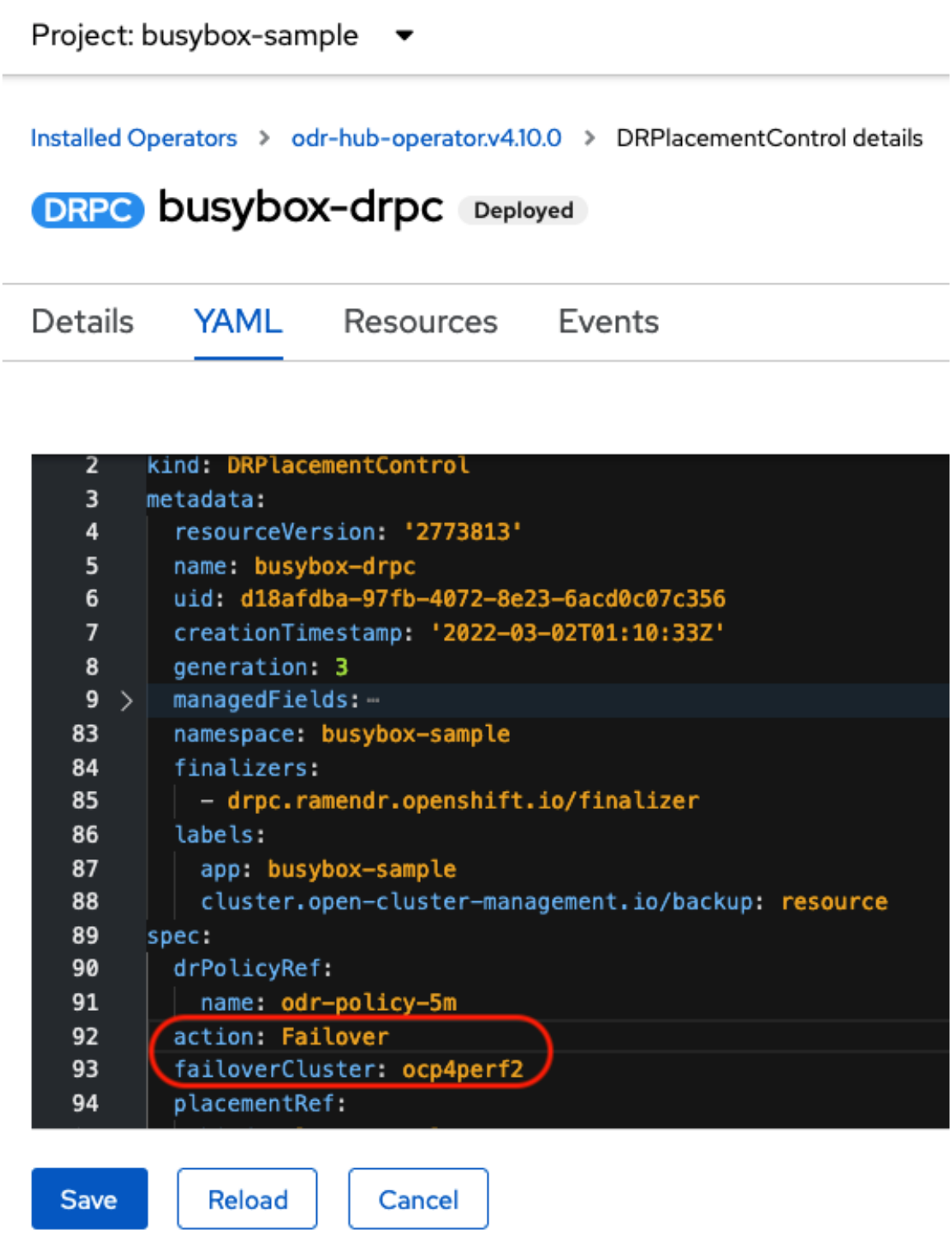
a. **Hub 클러스터에서 Installed Operators로 이동한 다음 Openshift DR Hub Operator 를 클릭합니다.**

b. **DRPlacementControl 탭을 클릭합니다.**

c. **DRPC busybox-drpc 를 클릭한 다음 YAML 보기를 클릭합니다.**

d. **아래 스크린샷과 같이 작업 및 장애 조치 클러스터 세부 정보를 추가합니다. 장애 조치 클러스터는 Secondary 관리 클러스터의 ACM 클러스터 이름입니다.**

DRPlacementControl 추가 장애 조치



e. 저장을 클릭합니다.

4. 이제 애플리케이션이 **YAML** 파일에 지정된 장애 조치 클러스터 **ocp4perf2** 인 **Secondary** 관리 클러스터에서 실행 중인지 확인합니다.

```
$ oc get pods,pvc -n busybox-sample
```

출력 예:

```
NAME        READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          35s

NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS                  AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-79f2a74d-6e2c-48fb-9ed9-666b74cfa1bb
5Gi        RWO            ocs-storagecluster-ceph-rbd   35s
```

5. **busybox** 가 기본 관리 클러스터에서 더 이상 실행되지 않는지 확인합니다.

```
$ oc get pods,pvc -n busybox-sample
```

출력 예:

```
No resources found in busybox-sample namespace.
```

중요

릴리스 노트의 **Known issues** 섹션에 설명된 대로 메트로-DR 알려진 문제에 대해 주의하십시오.

13장. 관리 클러스터 간에 애플리케이션 재배치

재배치 작업은 장애 조치와 매우 유사합니다. 재배치는 애플리케이션을 기반으로 하며 **DRPlacementControl** 을 사용하여 재배치를 트리거합니다. 장애 조치의 주요 차이점은 애플리케이션이 **failoverCluster**에서 축소되므로 **NetworkFence** 를 생성할 필요가 없다는 것입니다.

절차

1. **NetworkFence** 리소스를 제거하고 펜싱을 비활성화합니다.

장애 복구 또는 재배치 조치를 성공하려면 기본 관리 클러스터의 **NetworkFence** 를 삭제해야 합니다.

a. 보조 관리 클러스터에서 이 명령을 실행하고 이전 섹션에서 생성한 **NetworkFence YAML** 파일 이름에 맞게 **<** ***cluster1*** **>**을 수정합니다.

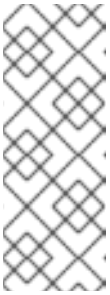
```
$ oc delete -f network-fence-<cluster1>.yaml
```

출력 예**:**

```
networkfence.csiaddons.openshift.io "network-fence-ocp4perf1" deleted
```

b. **Fenced** 인 **OpenShift Container Platform** 노드를 재부팅합니다.

이 단계는 이전 펜싱 클러스터의 일부 애플리케이션 **Pod(**이 경우 기본 관리 클러스터**)**가 비정상 상태**(**예**: CreateContainerError, CrashLoopBackOff)**이기 때문에 필요합니다. 이 문제는 한 번에 하나씩 모든 작업자 **OpenShift** 노드를 재부팅하여 가장 쉽게 해결할 수 있습니다.



참고

OpenShift 웹 콘솔 대시보드 및 개요 페이지를 사용하여 애플리케이션 상태 및 외부 스토리지를 평가할 수도 있습니다. 자세한 **OpenShift Data Foundation** 대시보드는 **Storage** → **Data Foundation** 으로 이동하여 찾을 수 있습니다.

c. 모든 **OpenShift** 노드가 재부팅되고 **Ready** 상태가 된 후 기본 관리 클러스터에서 이 명령을 실행하여 모든 **Pod**가 정상 상태인지 확인합니다. 이 쿼리의 출력은 **Pod**가 **0**이어야 합

니다.

```
$ oc get pods -A | egrep -v 'Running|Completed'
```

출력 예:

```
NAMESPACE                                 NAME
READY   STATUS     RESTARTS      AGE
```

중요

심각한 스토리지 통신으로 인해 **Pod**가 비정상 상태에 있는 경우 계속하기 전에 문제를 해결하고 해결합니다. 스토리지 클러스터는 **OpenShift** 외부에 있으므로 **OpenShift** 애플리케이션이 정상 상태가 되는 사이트 중단 후에도 제대로 복구되어야 합니다.

2. **DRPolicy** 를 **Unfenced** 상태로 수정합니다.

ODR HUB Operator가 기본 관리 클러스터가 제거되었음을 확인하기 위해 **DRPolicy** 는 새로 **Unfenced** 클러스터에 대해 수정해야 합니다.

a. **Hub** 클러스터에서 **DRPolicy** 를 편집하고 **<** ***cluster1*** **>(**예**: ocp4perf1)**을 **ManuallyFenced** 에서 **Unfenced** 으로 변경합니다.

```
$ oc edit drpolicy odr-policy
```

출력 예:

```
[...]
spec:
  drClusterSet:
  - clusterFence: Unfenced  ## <-- Modify from ManuallyFenced to Unfenced
    name: ocp4perf1
    region: metro
    s3ProfileName: s3-primary
  - clusterFence: Unfenced
    name: ocp4perf2
    region: metro
    s3ProfileName: s3-secondary
[...]
```

출력 예:

```
drpolicy.ramendr.openshift.io/odr-policy edited
```

b.

Hub 클러스터의 DRPolicy 상태가 기본 관리형 클러스터 의 Unfenced 로 변경되었는지 확인합니다.

```
$ oc get drpolicies.ramendr.openshift.io odr-policy -o yaml | grep -A 6 drClusters
```

출력 예:

```
drClusters:
  ocp4perf1:
    status: Unfenced
    string: ocp4perf1
  ocp4perf2:
    status: Unfenced
    string: ocp4perf2
```

3.

DRPlacementControl 을 failback으로 수정

a.

Hub 클러스터에서 Installed Operators로 이동한 다음 Openshift DR Hub Operator 를 클릭합니다.

b.

DRPlacementControl 탭을 클릭합니다.

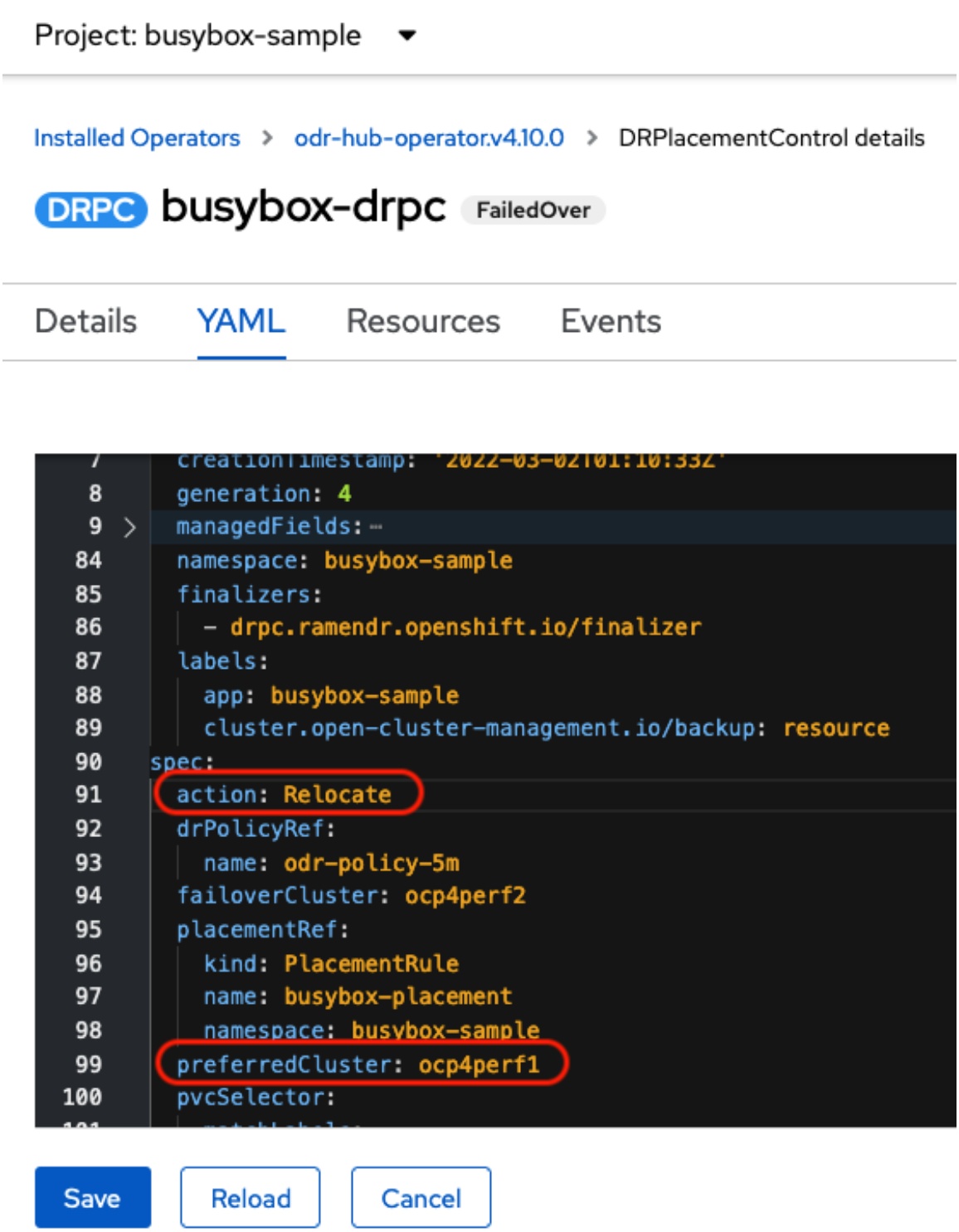
c.

DRPC busybox-drpc 를 클릭한 다음 YAML 보기를 클릭합니다.

d.

작업 변경 작업 수정 .

DRPlacementControl Relocate에 대한 작업 수정



e. 저장을 클릭합니다.

f. 이제 애플리케이션이 **Primary managed cluster**에서 실행 중인지 확인합니다. **failback**은 **YAML** 파일에 지정된 대로 기본**Cluster ocp4perf1** 에 해당합니다. 장애 조치**(failover)** 작업 전에 애플리케이션이 실행 중인 위치인 **YAML** 파일에 지정된 대로 **failback**은 기본 클러스터 **ocp4perf1**에 해당합니다.

```
$ oc get pods,pvc -n busybox-sample
```

출력 예:

```
NAME          READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          60s

NAME                                  STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS                  AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-79f2a74d-6e2c-48fb-9ed9-
666b74cfa1bb   5Gi        RWO            ocs-storagecluster-ceph-rbd   61s
```

g. **Secondary** 관리 클러스터에서 **busybox** 가 실행 중인지 확인합니다. **busybox** 애플리케이션은 이 관리 클러스터에서 더 이상 실행되지 않아야 합니다.

```
$ oc get pods,pvc -n busybox-sample
```

출력 예:

```
No resources found in busybox-sample namespace.
```

중요

릴리스 노트의 **Known issues** 섹션에 설명된 대로 메트로-DR 알려진 문제에 대해 주의하십시오.