Red Hat OpenStack Platform 16.1

컨테이너화된 Red Hat Ceph를 사용하여 오버클라우드 배포

컨테이너화된 Red Hat Ceph 클러스터를 배포하고 사용하도록 director 구성

Last Updated: 2023-12-06

Red Hat OpenStack Platform 16.1 컨테이너화된 Red Hat Ceph를 사용하여 오버클라우드 배포

컨테이너화된 Red Hat Ceph 클러스터를 배포하고 사용하도록 director 구성

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 가이드에서는 Red Hat OpenStack Platform director를 사용하여 컨테이너화된 Red Hat Ceph Storage 클러스터가 있는 오버클라우드를 생성하는 방법을 설명합니다. 여기에는 director를 통해 Ceph 클러스터를 사용자 지정하는 지침이 포함되어 있습니다.

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 어떻게 더 나은지 알려주십시오.

직접 문서 피드백(DDF) 기능 사용

피드백 추가 DDF 기능을 사용하여 특정 문장, 단락 또는 코드 블록에 대한 직접 의견을 제출할 수 있습니다.

1. *다중 페이지 HTML* 형식으로 문서를 봅니다.
2. 문서의 오른쪽 상단에 **피드백** 버튼이 표시되는지 확인합니다.
3. 주석 처리하려는 텍스트 부분을 강조 표시합니다.
4. **피드백 추가를** 클릭합니다.
5. 코멘트를 사용하여 **피드백 추가** 필드를 완료합니다.
6. 선택 사항: 문서 팀이 문제에 대한 자세한 설명을 위해 연락을 드릴 수 있도록 이메일 주소를 추가합니다.
7. **Submit** 을 클릭합니다.

1장. 소개

Red Hat OpenStack Platform director는 오버클라우드라는 클라우드 환경을 생성합니다. director를 사용하여 Red Hat Ceph Storage(director 또는 기존 Ceph Storage 클러스터와 함께 생성된 Ceph Storage 클러스터 모두)와의 통합을 포함하여 오버클라우드의 추가 기능을 구성할 수 있습니다.

이 가이드에서는 기존 Ceph Storage 클러스터를 오버클라우드와 통합하는 방법을 설명합니다. 즉, director는 스토리지 요구 사항에 Ceph Storage 클러스터를 사용하도록 오버클라우드를 구성합니다. 오버클라우드 구성 외부에서 클러스터 자체를 관리하고 확장합니다.

이 가이드에서는 오버클라우드와 함께 컨테이너화된 Red Hat Ceph Storage 클러스터를 배포하는 방법을 설명합니다. director는 **ceph-ansible** 패키지를 통해 제공되는 Ansible 플레이북을 사용하여 컨테이너화된 Ceph 클러스터를 배포합니다. director는 클러스터의 구성 및 확장 작업도 관리합니다.

RHOSP(Red Hat OpenStack Platform)의 컨테이너화된 서비스에 대한 자세한 내용은 *Director 설치 및 사용* 가이드 의 CLI 툴을 사용하여 기본 오버클라우드 구성을 참조하십시오.

1.1. CEPH STORAGE 소개

Red Hat Ceph Storage는 뛰어난 성능, 신뢰성 및 확장성을 제공하도록 설계된 분산 데이터 개체 저장소입니다. 분산 개체 저장소는 구조화되지 않은 데이터를 수용하고 클라이언트가 최신 개체 인터페이스와 기존 인터페이스를 동시에 사용할 수 있기 때문에 스토리지의 미래입니다. 모든 Ceph 배포의 핵심은 Ceph Storage 클러스터입니다. 이 클러스터는 여러 유형의 데몬으로 구성되지만, 주로 다음 두 가지입니다.

Ceph OSD(오브젝트 스토리지 데몬)

Ceph OSD는 Ceph 클라이언트를 대신하여 데이터를 저장합니다. 또한 Ceph OSD는 Ceph 노드의 CPU 및 메모리를 사용하여 데이터 복제, 재조정, 복구, 모니터링 및 보고 기능을 수행합니다.

Ceph 모니터

Ceph 모니터는 스토리지 클러스터의 현재 상태와 함께 Ceph 스토리지 클러스터 맵의 마스터 사본을 유지 관리합니다.

Red Hat Ceph Storage에 대한 자세한 내용은 Red Hat Ceph Storage Architecture Guide를 참조하십시오.

1.2. 요구 사항

이 가이드에는 Director 설치 및 사용 가이드에 대한 정보가 포함되어 있습니다.

오버클라우드를 사용하여 컨테이너화된 Ceph Storage 클러스터를 배포하기 전에 환경에 다음 구성이 포함되어야 합니다.

- RHOSP(Red Hat OpenStack Platform) director가 설치된 언더클라우드 호스트입니다. 언더클라우드의 director 설치를 참조하십시오.
- Red Hat Ceph Storage에 권장되는 추가 하드웨어입니다. 권장 하드웨어에 대한 자세한 내용은 Red Hat Ceph Storage 하드웨어 가이드를 참조하십시오 .

중요

Ceph Monitor 서비스는 오버클라우드 컨트롤러 노드에 설치되므로 성능 문제를 방지하려면 적절한 리소스를 제공해야 합니다. 해당 환경의 컨트롤러 노드에서 Ceph 모니터 데이터에 대해 메모리에 16GB 이상의 RAM과 SSD(반도체 드라이브) 스토리지를 사용하는지 확인합니다. 중간에서 대형 Ceph 설치의 경우 최소 500GB의 Ceph 모니터 데이터를 제공합니다. 클러스터가 불안정해지는 경우 levelDB 증가를 방지하려면 이 공간이 필요합니다. 다음 예제는 Ceph 스토리지 클러스터의 일반적인 크기입니다.

- 작은: 250테라바이트
- 중간: 페타바이트 1개
- 큰: 2페타바이트 이상.

Red Hat OpenStack Platform director를 사용하여 Ceph Storage 노드를 생성하는 경우 다음 요구 사항을 확인합니다.

1.2.1. Ceph Storage 노드 요구 사항

Ceph Storage 노드는 Red Hat OpenStack Platform 환경에 스토리지 오브젝트를 제공합니다.

Ceph Storage 노드의 프로세서, 메모리, NIC(네트워크 인터페이스 카드) 및 디스크 레이아웃을 선택하는 방법에 대한 자세한 내용은 *Red Hat Ceph Storage Hardware* 가이드의 Hardware selection recommendations for Red Hat Ceph Storage를 참조하십시오. 각 Ceph Storage 노드에는 서버의 마더보드에서 IPMI(Intelligent Platform Management Interface) 기능과 같은 지원되는 전원 관리 인터페이스가 필요합니다.

참고

RHOSP(Red Hat OpenStack Platform) director는 **ceph-ansible** 을 사용하며 Ceph Storage 노드의 루트 디스크에 OSD 설치를 지원하지 않습니다. 즉, 지원되는 Ceph Storage 노드에 대해 두 개 이상의 디스크가 필요합니다.

Ceph Storage 노드 및 RHEL 호환성

- RHOSP 16.1은 RHEL 8.2에서 지원됩니다. 그러나 Ceph Storage 역할 업데이트에 매핑된 호스트는 최신 주요 RHEL 릴리스로 제공됩니다. RHOSP 16.1 이상으로 업그레이드하기 전에 Red Hat Knowledgebase 문서 Red Hat Ceph Storage를 검토하십시오. 지원되는 설정 .

PG(배치 그룹)

- Ceph Storage는 배치 그룹(PG)을 사용하여 대규모의 효율적인 동적 오브젝트 추적을 지원합니다. OSD 오류가 발생하거나 클러스터를 재조정하는 경우 Ceph Storage에서 배치 그룹 및 해당 컨텐츠를 이동하거나 복제할 수 있으므로, Ceph 클러스터의 효율적인 재조정 및 복구가 가능합니다.

- director가 생성하는 기본 배치 그룹 수가 항상 최적의 개수는 아니므로 필요에 따라 올바른 배치 그룹 수를 계산해야 합니다. 배치 그룹 계산기를 사용하여 올바른 개수를 계산할 수 있습니다. PG 계산기를 사용하려면 서비스당 예상 스토리지 사용량을 백분율로 입력하고 개수 OSD와 같은 Ceph 클러스터에 대한 기타 속성을 입력합니다. 계산기는 풀당 최적의 PG 수를 반환합니다. 자세한 내용은 풀 계산기당 PG(배치 그룹)를 참조하십시오.

- 자동 확장은 배치 그룹을 관리하는 대체 방법입니다. 자동 확장 기능을 사용하면 특정 수의 배치 그룹이 아니라 서비스당 예상 Ceph Storage 요구 사항을 백분율로 설정합니다. Ceph는 클러스터

사용 방법에 따라 배치 그룹을 자동으로 확장합니다. 자세한 내용은 *Red Hat Ceph Storage Strategies Guide*의 자동 확장 배치 그룹을 참조하십시오.

프로세서

- Intel 64 또는 AMD64 CPU 확장을 지원하는 64비트 x86 프로세서입니다.

네트워크 인터페이스 카드

- 최소 1개의 1Gbps NIC(네트워크 인터페이스 카드)가 필요합니다. Red Hat에서는 프로덕션 환경에서 적어도 두 개 이상의 NIC를 사용하도록 권장합니다. 본딩된 인터페이스나 태그된 VLAN 트래픽 위임에는 추가 NIC를 사용하십시오. 대량의 트래픽에 서비스를 제공하는 RHOSP(Red Hat OpenStack Platform) 환경을 구축하는 경우 특히 스토리지 노드에 10Gbps 인터페이스를 사용합니다.

전원 관리

- 각 컨트롤러 노드에는 서버의 마더보드에서 IPMI(Intelligent Platform Management Interface) 기능과 같은 지원되는 전원 관리 인터페이스가 필요합니다.

1.3. 추가 리소스

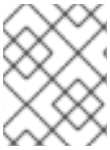
/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml 환경 파일은 director에 ceph-ansible 프로젝트에서 파생된 플레이북을 사용하도록 director에 지시합니다. 이러한 플레이북은 언더클라우드의 **/usr/share/ceph-ansible/** 에 설치됩니다. 특히 다음 파일에는 플레이북에서 적용하는 모든 기본 설정이 포함되어 있습니다.

- **/usr/share/ceph-ansible/group_vars/all.yml.sample**

주의

ceph-ansible 에서는 플레이북을 사용하여 컨테이너화된 Ceph Storage를 배포하지만 배포를 사용자 지정하도록 해당 파일을 편집하지 마십시오. 대신 heat 환경 파일을 사용하여 이러한 플레이북에서 설정한 기본값을 재정의합니다. **ceph-anible** 플레이북을 직접 편집하면 배포에 실패합니다.

컨테이너화된 Ceph Storage에 대해 director에서 적용하는 기본 설정에 대한 자세한 내용은 **/usr/share/openstack-tripleo-heat-templates/deployment/ceph-ansible** 의 heat 템플릿을 참조하십시오.



참고

이러한 템플릿을 읽으려면 director에서 환경 파일과 heat 템플릿이 작동하는 방법을 자세히 이해해야 합니다. 참조는 Heat 템플릿 및 환경 파일 이해를 참조하십시오.

RHOSP의 컨테이너화된 서비스에 대한 자세한 내용은 *Director 설치 및 사용 가이드의 CLI 툴을 사용하여 기본 오버클라우드 구성*을 참조하십시오.

2장. 오버클라우드 배포를 위한 CEPH STORAGE 노드 준비

이 시나리오의 모든 노드는 전원 관리에 IPMI를 사용하는 베어 메탈 시스템입니다. director가 Red Hat Enterprise Linux 8 이미지를 각 노드에 복사하므로 이러한 노드에는 운영 체제가 필요하지 않습니다. 또한 이러한 노드의 Ceph Storage 서비스가 컨테이너화되어 있습니다. director는 인트로스펙션 및 프로비저닝 프로세스 중에 프로비저닝 네트워크를 통해 각 노드에 통신합니다. 모든 노드는 기본 VLAN을 통해 이 네트워크에 연결됩니다.

2.1. CEPH STORAGE 노드 디스크 정리

Ceph 스토리지 OSD 및 저널 파티션에는 GPT 디스크 레이블이 필요합니다. 따라서 Ceph Storage의 추가 디스크를 사용하려면 Ceph OSD 서비스를 설치하기 전에 GPT로 변환해야 합니다. director가 GPT 레이블을 설정할 수 있도록 디스크에서 모든 메타데이터를 삭제해야 합니다.

/home/stack/undercloud.conf 파일에 다음 설정을 추가하여 기본적으로 모든 디스크 메타데이터를 삭제하도록 director를 구성할 수 있습니다.

```
clean_nodes=true
```

이 옵션을 사용하면 베어 메탈 프로비저닝 서비스에서 추가 단계를 실행하여 노드를 부팅하고 노드를 **사용할 수** 있도록 설정할 때마다 디스크를 정리합니다. 이 프로세스는 첫 번째 인트로스펙션 이후와 각 배포 전에 전원 사이클을 추가합니다. 베어 메탈 프로비저닝 서비스는 **wipefs --force --all** 명령을 사용하여 정리를 수행합니다.

이 옵션을 설정한 후 **openstack undercloud install** 명령을 실행하여 이 구성 변경을 실행합니다.

주의

wipefs --force --all 명령은 디스크의 모든 데이터와 메타데이터를 삭제하지만 안전한 지우기를 수행하지는 않습니다. 보안 지우는 시간이 훨씬 더 오래 걸립니다.

2.2. 노드 등록

director가 노드와 통신할 수 있도록 JSON 형식의 노드 인벤토리 파일(in**stackenv.json**)을 director로 가져옵니다. 이 인벤토리 파일에는 director가 노드를 등록하는 데 사용할 수 있는 하드웨어 및 전원 관리 세부 정보가 포함되어 있습니다.

```
{
    "nodes":[
        {
            "mac":[
                "b1:b1:b1:b1:b1:b1"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
```

```
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.205"
    },
    {
        "mac":[
            "b2:b2:b2:b2:b2:b2"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.206"
    },
    {
        "mac":[
            "b3:b3:b3:b3:b3:b3"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.207"
    },
    {
        "mac":[
            "c1:c1:c1:c1:c1:c1"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.208"
    },
    {
        "mac":[
            "c2:c2:c2:c2:c2:c2"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.209"
    },
    {
```

```
            "mac":[
                "c3:c3:c3:c3:c3:c3"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.210"
        },
        {
            "mac":[
                "d1:d1:d1:d1:d1:d1"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.211"
        },
        {
            "mac":[
                "d2:d2:d2:d2:d2:d2"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.212"
        },
        {
            "mac":[
                "d3:d3:d3:d3:d3:d3"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.213"
        }
    ]
}
```

절차

1. 인벤토리 파일을 생성한 후 stack 사용자의 홈 디렉터리(**/home/stack/instackenv.json)에 파일을 저장합니다.**

2. stack 사용자를 초기화한 다음 **instackenv.json** 인벤토리 파일을 director로 가져옵니다.

```
$ source ~/stackrc
$ openstack overcloud node import ~/instackenv.json
```

openstack overcloud node import 명령은 인벤토리 파일을 가져와 각 노드를 director에 등록합니다.

3. 커널 및 ramdisk 이미지를 각 노드에 할당합니다.

```
$ openstack overcloud node configure <node>
```

결과

노드가 director에 등록되어 구성됩니다.

2.3. CEPH STORAGE에 대한 사전 배포 검증

오버클라우드 배포 실패를 방지하려면 서버에 필요한 패키지가 있는지 확인합니다.

2.3.1. ceph-ansible 패키지 버전 확인

언더클라우드에는 오버클라우드를 배포하기 전에 잠재적인 문제를 식별하기 위해 실행할 수 있는 Ansible 기반 검증이 포함되어 있습니다. 이러한 검증을 통해 일반적인 문제가 발생하기 전에 식별하여 오버클라우드 배포 실패를 방지할 수 있습니다.

절차

ceph-ansible 패키지의 수정 버전이 설치되어 있는지 확인합니다.

```
$ ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-ansible-installed.yaml
```

2.3.2. 사전 프로비저닝된 노드의 패키지 확인

Ceph는 특정 패키지 집합이 있는 Overcloud 노드만 서비스할 수 있습니다. 사전 프로비저닝된 노드를 사용하는 경우 이러한 패키지가 있는지 확인할 수 있습니다.

사전 프로비저닝된 노드에 대한 자세 한 내용은 사전 프로비저닝된 노드가 있는 기본 오버클라우드 구성을 참조하십시오.

절차

서버에 필수 패키지가 포함되어 있는지 확인합니다.

```
ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-dependencies-installed.yaml
```

2.4. 프로필에 노드 수동 태그

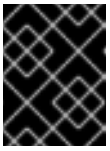
각 노드를 등록한 후에는 하드웨어를 검사하고 특정 프로필에 노드를 태그해야 합니다. 프로필 태그를 사용하여 노드를 플레이버에 일치시킨 다음 배포 역할에 플레이버를 할당합니다.

절차

1. 하드웨어 인트로스펙션을 트리거하여 각 노드의 하드웨어 속성을 검색합니다.

```
$ openstack overcloud node introspect --all-manageable --provide
```

   - all **-manageable** 옵션은 관리 상태에 있는 노드만 인트로스펙션합니다. 이 예에서는 모든 노드가 관리 상태에 있습니다.
   - **--provide** 옵션은 인트로스펙션 후 모든 노드를 **active** 상태로 재설정합니다.



중요

이 프로세스가 성공적으로 완료되었는지 확인합니다. 베어 메탈 노드의 경우 이 프로세스는 일반적으로 15분 정도 걸립니다.

2. 노드 목록을 검색하여 UUID를 확인합니다.

```
$ openstack baremetal node list
```

3. profile 옵션을 각 노드의 **properties/capabilities** 매개변수에 추가하여 특정 프로필에 노드를 수동으로 태그합니다. **profile** 옵션을 추가하면 각 프로필에 노드를 태그합니다.

참고

수동 태그 대신 AHC(Automated Health Check) 툴을 사용하여 벤치마킹 데이터를 기반으로 다수의 노드에 자동으로 태그를 지정합니다.

예를 들어 일반적인 배포에는 세 개의 프로필, 즉 **control**,**compute**, **ceph-storage** 가 포함되어 있습니다. 다음 명령을 실행하여 각 프로필에 대해 세 개의 노드를 태그합니다.

```
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local'
1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local'
6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local'
6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
484587b2-b3b3-40d5-925b-a26a2fa3036f
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
d010460b-38f2-4800-9cc4-d69f0d067efe
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
d930e613-3e14-44b9-8240-4f3559801ea6
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' 484587b2-b3b3-40d5-925b-a26a2fa3036f
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' d010460b-38f2-4800-9cc4-d69f0d067efe
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' d930e613-3e14-44b9-8240-4f3559801ea6
```

작은 정보

Ceph MON 및 Ceph MDS 서비스의 노드에 태그를 지정하는 데 사용할 수 있는 새 사용자 지정 프로필을 구성할 수도 있습니다. 자세한 내용은 3장. *전용 노드에 Ceph 서비스 배포* 을 참조하십시오.

2.5. 멀티 디스크 클러스터의 ROOT 디스크 정의

여러 디스크가 있는 노드의 경우 director가 프로비저닝 중에 root 디스크를 식별해야 합니다. 예를 들어 대부분의 Ceph Storage 노드는 여러 디스크를 사용합니다. 기본적으로 director는 프로비저닝 프로세스 중에 오버클라우드 이미지를 root 디스크에 씁니다.

director가 root 디스크를 쉽게 식별할 수 있도록 다음과 같은 속성을 정의할 수 있습니다.

- **모델** (문자열): 장치 식별자.
- **벤더** (문자열): 장치 벤더.
- **serial** (문자열): 디스크 일련 번호.
- **hctl** (문자열): host:Channel:Target: SCSI의 Lun.
- **크기** (정수): 장치 크기(GB)입니다.
- **WWN (** 문자열): 고유한 스토리지 식별자.
- **wwn_with_extension** (문자열): 공급업체 확장이 추가된 고유한 스토리지 식별자입니다.
- **wwn_vendor_extension** (문자열): 고유한 벤더 스토리지 식별자.
- **rotational** (부울): 회전 장치(HDD)의 경우 true이며 그렇지 않으면 false(SSD)입니다.
- **이름** (문자열): 장치 이름(예: /dev/sdb1)

중요

name 속성은 영구적인 이름이 있는 장치에만 사용합니다. 노드가 부팅될 때 값이 변경될 수 있으므로 **name**을 사용하여 다른 장치에 대해 root 디스크를 설정하지 마십시오.

일련 번호를 사용하여 root 장치를 지정할 수 있습니다.

절차

1. 각 노드의 하드웨어 인트로스펙션에서 디스크 정보를 확인합니다. 다음 명령을 실행하여 노드의 디스크 정보를 표시합니다.

```
(undercloud)$ openstack baremetal introspection data save 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 | jq ".inventory.disks"
```

예를 들어 노드 1개의 데이터에서 디스크 3개가 표시될 수 있습니다.

```
[
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
```

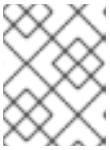
```
    "name": "/dev/sda",
    "wwn_vendor_extension": "0x1ea4dcc412a9632b",
    "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4dcc412a9632b"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdb",
    "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
    "wwn_with_extension": "0x61866da04f380d001ea4e13c12e36ad6",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380d00",
    "serial": "61866da04f380d001ea4e13c12e36ad6"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdc",
    "wwn_vendor_extension": "0x1ea4e31e121cfb45",
    "wwn_with_extension": "0x61866da04f37fc001ea4e31e121cfb45",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f37fc00",
    "serial": "61866da04f37fc001ea4e31e121cfb45"
  }
]
```

2. **openstack baremetal node set --property root_device=**를 입력하여 노드의 root 디스크를 설정합니다. root 디스크를 정의하는 데 가장 적절한 하드웨어 속성값을 포함시킵니다.

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":"
<serial_number>"}' <node-uuid>
```

예를 들어, root 장치를 일련 번호가 **61866da04f380d001ea4e13c12e36ad6**인 disk 2로 설정하려면 다음 명령을 입력합니다.

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":
"61866da04f380d001ea4e13c12e36ad6"}' 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
```



참고

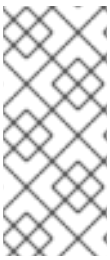
각 노드의 BIOS를 설정하여 선택한 root 디스크로 부팅이 포함되도록 합니다. 먼저 네트워크에서 부팅한 다음 root 디스크에서 부팅하도록 부팅 순서를 구성합니다.

director가 root 디스크로 사용할 특정 디스크를 식별합니다. **openstack overcloud deploy** 명령을 실행하면 director가 오버클라우드 이미지를 프로비저닝하고 root 디스크에 씁니다.

2.6. OVERCLOUD-MINIMAL 이미지를 사용하여 RED HAT 서브스크립션 인타이틀먼트 사용 방지

기본적으로 director는 프로비저닝 프로세스 중에 QCOW2 **overcloud-full** 이미지를 root 디스크에 씁니다. **overcloud-full** 이미지는 유효한 Red Hat 서브스크립션을 사용합니다. 그러나 예를 들어 **overcloud-minimal** 이미지를 사용하여 다른 OpenStack 서비스를 실행하지 않으려는 베어 OS를 프로비저닝하고 서브스크립션 인타이틀먼트를 사용할 수 있습니다.

이에 대한 가장 일반적인 사용 사례는 Ceph 데몬으로만 노드를 프로비저닝하는 경우 발생합니다. 이 경우와 유사한 사용 사례의 경우 **overcloud-minimal** 이미지 옵션을 사용하여 Red Hat 서브스크립션 구매 한도에 도달하지 않도록 할 수 있습니다. **overcloud-minimal** 이미지를 가져오는 방법에 관한 자세한 내용은 Obtaining images for overcloud nodes 를 참조하십시오.



참고

RHOSP(Red Hat OpenStack Platform) 서브스크립션에는 OVS(Open vSwitch)가 포함되지만 **overcloud-minimal** 이미지를 사용하는 경우에는 OVS와 같은 코어 서비스를 사용할 수 없습니다. Ceph Storage 노드를 배포하는 데는 OVS가 필요하지 않습니다. **ovs_bond**를 사용하여 본드를 정의하는 대신 **linux_bond**를 사용하십시오. **linux_bond**에 관한 자세한 내용은 Linux bonding options 를 참조하십시오.

절차

1. **overcloud-minimal** 이미지를 사용하도록 director를 구성하려면 다음 이미지 정의가 포함된 환경 파일을 생성합니다.

```
parameter_defaults:
  <roleName>Image: overcloud-minimal
```

2. **<roleName>**을 역할 이름으로 바꾸고 **Image**를 역할의 이름에 추가합니다. 다음 예에서는 Ceph 스토리지 노드의 **overcloud-minimal** 이미지를 보여줍니다.

```
parameter_defaults:
  CephStorageImage: overcloud-minimal
```

3. **roles_data.yaml** 역할 정의 파일에서 **rhsm_enforce** 매개변수를 **False**로 설정합니다.

```
rhsm_enforce: False
```

4. **openstack overcloud deploy** 명령에 환경 파일을 전달합니다.

참고

overcloud-minimal 이미지는 OVS가 아닌 표준 Linux 브릿지만 지원합니다. OVS는 Red Hat OpenStack Platform 서브스크립션 인타이틀먼트가 필요한 OpenStack 서비스이기 때문입니다.

3장. 전용 노드에 CEPH 서비스 배포

기본적으로 director는 컨트롤러 노드에 Ceph MON 및 Ceph MDS 서비스를 배포합니다. 이는 소규모 배포에 적합합니다. 그러나 대규모 배포를 수행하는 경우 Ceph 클러스터의 성능을 개선하기 위해 전용 노드에 Ceph MON 및 Ceph MDS 서비스를 배포하는 것이 좋습니다. 전용 노드에서 격리하려는 서비스에 대한 사용자 지정 역할을 생성합니다.

참고

사용자 지정 역할에 대한 자세한 내용은 Advanced Overcloud Customization 가이드의 Create a New Role 을 참조하십시오.

director는 다음 파일을 모든 오버클라우드 역할에 대한 기본 참조로 사용합니다.

- **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml**

3.1. 사용자 정의 역할 파일 생성

사용자 지정 역할 파일을 생성하려면 다음 단계를 완료합니다.

절차

1. 사용자 지정 역할을 추가할 수 있도록 **/home/stack/templates/** 에 **roles_data.yaml** 파일의 사본을 만듭니다.

```
$ cp /usr/share/openstack-tripleo-heat-templates/roles_data.yaml
/home/stack/templates/roles_data_custom.yaml
```

2. **openstack overcloud deploy** 명령에 새 사용자 지정 역할 파일을 포함합니다.

3.2. CEPH MON 서비스의 사용자 지정 역할 및 플레이버 만들기

Ceph MON 역할에 대해 사용자 지정 역할 **CephMon** 및 플레이버 **ceph-mon** 을 생성하려면 다음 단계를 완료합니다. 3장. *전용 노드에 Ceph 서비스 배포* 에 설명된 기본 역할 데이터 파일의 사본이 이미 있어야 합니다.

절차

1. **/home/stack/templates/roles_data_custom.yaml** 파일을 엽니다.
2. Controller 역할에서 Ceph MON 서비스, **OS::TripleO::Services::CephMon** 의 서비스 항목을 제거합니다.
3. 컨트롤러 역할에 **OS::TripleO::Services::CephClient** 서비스를 추가합니다.

```
[...]
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
```

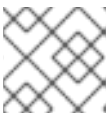
```
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
[...]
```

4. **roles_data_custom.yaml** 파일 끝에 **Ceph MON 서비스 및 기타 필요한 모든 노드 서비스가 포함된 사용자 지정 CephMon** 역할을 추가합니다.

```
- name: CephMon
  ServicesDefault:
    # Common Services
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    # Role-Specific Services
    - OS::TripleO::Services::CephMon
```

5. **openstack flavor create** 명령을 입력하여 **CephMon** 역할에 대해 **ceph-mon** 이라는 새 플레이버를 정의합니다.

```
$ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus 4 ceph-mon
```



참고

이 명령에 대한 자세한 내용은 **openstack flavor create --help** 를 입력합니다.

6. 이 플레이버를 **ceph-mon** 이라는 새 프로필에 매핑합니다.

```
$ openstack flavor set --property "cpu_arch"="x86_64" --property
"capabilities:boot_option"="local" --property "capabilities:profile"="ceph-mon" ceph-mon
```

참고

이 명령에 대한 자세한 내용은 **openstack flavor set --help** 를 입력합니다.

7. 노드를 새 **ceph-mon** 프로필에 태그합니다.

```
$ openstack baremetal node set --property capabilities='profile:ceph-mon',boot_option:local'
UUID
```

8. 다음 구성을 **node-info.yaml** 파일에 추가하여 **ceph-mon** 플레이버를 CephMon 역할과 연결합니다.

```
parameter_defaults:
  OvercloudCephMonFlavor: CephMon
  CephMonCount: 3
```

노드 태그 지정에 대한 자세한 내용은 2.4절. "프로필에 노드 수동 태그" 의 내용을 참조하십시오. 사용자 지정 역할 프로필에 대한 자세한 내용은 노드 Into Profiles 태그를 참조하십시오.

3.3. CEPH MDS 서비스에 대한 사용자 지정 역할 및 플레이버 만들기

Ceph MDS 역할에 대한 사용자 지정 역할 **CephMDS** 및 플레이버 **ceph-mds** 를 생성하려면 다음 단계를 완료합니다. 3장. *전용 노드에 Ceph 서비스 배포* 에 설명된 기본 역할 데이터 파일의 사본이 이미 있어야 합니다.

절차

1. **/home/stack/templates/roles_data_custom.yaml** 파일을 엽니다.

2. Controller 역할에서 Ceph MDS 서비스, **OS::TripleO::Services::CephMds** 의 서비스 항목을 제거합니다.

```
[...]
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    # - OS::TripleO::Services::CephMds (1)
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
[...]
```

(1) 이 줄을 주석 처리하십시오. 다음 단계에서는 새 사용자 지정 역할에 이 서비스를 추가합니다.

3. **roles_data_custom.yaml** 파일 끝에 **Ceph MDS 서비스 및 기타 필요한 모든 노드 서비스가 포함된 사용자 지정 CephMDS** 역할을 추가합니다.

```
- name: CephMDS
  ServicesDefault:
    # Common Services
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
```

```
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    # Role-Specific Services
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephClient 1
```

1 Ceph MDS 서비스에는 Ceph MON 또는 Ceph 클라이언트 서비스로 설정할 수 있는 관리자 인증 키가 필요합니다. Ceph MON 서비스 없이 전용 노드에 Ceph MDS를 배포하는 경우 새 Ceph **MDS** 역할에 Ceph 클라이언트 서비스도 포함해야 합니다.

4. **openstack flavor create** 명령을 입력하여 이 역할에 대해 **ceph-mds** 라는 새 플레이버를 정의합니다.

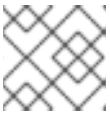
```
$ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus 4 ceph-mds
```

참고

이 명령에 대한 자세한 내용은 **openstack flavor create --help** 를 입력합니다.

5. **ceph-mds** 라는 새 프로필에 새 **ceph-mds** 플레이버를 매핑합니다.

```
$ openstack flavor set --property "cpu_arch"="x86_64" --property
"capabilities:boot_option"="local" --property "capabilities:profile"="ceph-mds" ceph-mds
```



참고

이 명령에 대한 자세한 내용은 **openstack flavor set --help** 를 입력합니다.

6. 노드를 새 **ceph-mds** 프로필에 태그합니다.

```
$ openstack baremetal node set --property capabilities='profile:ceph-mds,boot_option:local'
UUID
```

노드 태그 지정에 대한 자세한 내용은 2.4절. "프로필에 노드 수동 태그" 의 내용을 참조하십시오. 사용자 지정 역할 프로필에 대한 자세한 내용은 노드 Into Profiles 태그를 참조하십시오.

4장. 스토리지 서비스 사용자 정의

director에서 제공하는 heat 템플릿 컬렉션에는 기본 Ceph Storage 구성을 활성화하는 데 필요한 템플릿과 환경 파일이 이미 포함되어 있습니다.

director는 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 환경 파일을 사용하여 Ceph 클러스터를 생성하고 배포 중에 오버클라우드와 통합합니다. 이 클러스터에는 컨테이너화된 Ceph Storage 노드가 있습니다. OpenStack의 컨테이너화된 서비스에 대한 자세한 내용은 *Director 설치 및 사용 가이드의 CLI 툴을 사용하여 기본 오버클라우드 구성* 을 참조하십시오.

Red Hat OpenStack director는 배포된 Ceph 클러스터에 기본 기본 설정도 적용합니다. 또한 사용자 지정 환경 파일에서 추가 구성을 정의해야 합니다.

절차

1. **/home/stack/templates/** 에 **storage-config.yaml** 파일을 생성합니다. 이 예제에서 **~/templates/storage-config.yaml** 파일에는 환경에 대한 대부분의 오버클라우드 관련 사용자 지정 설정이 포함되어 있습니다. 사용자 지정 환경 파일에 포함된 매개변수는 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 파일의 해당 기본 설정을 재정의합니다.

2. **parameter_defaults** 섹션을 **~/templates/storage-config.yaml** 에 추가합니다. 이 섹션에는 오버클라우드에 대한 사용자 지정 설정이 포함되어 있습니다. 예를 들어 **vxlan** 을 네트워킹 서비스(**neutron**)의 네트워크 유형으로 설정하려면 사용자 지정 환경 파일에 다음 스니펫을 추가합니다.

```
parameter_defaults:
  NeutronNetworkType: vxlan
```

3. 필요한 경우 요구 사항에 따라 **parameter_defaults** 에서 다음 옵션을 설정합니다.

옵션	설명	기본값
CinderEnableIscsiBackend	iSCSI 백엔드 활성화	false
CinderEnableRbdBackend	Ceph Storage 백엔드 활성화	true
CinderBackupBackend	볼륨 백업의 백엔드로 ceph 또는 swift를 설정합니다. 자세한 내용은 4.4절. "Ceph를 사용하도록 백업 서비스 구성"의 내용을 참조하십시오.	Ceph
NovaEnableRbdBackend	Nova 임시 스토리지용 Ceph Storage 활성화	true
GlanceBackend	이미지 서비스에서 사용할 백엔드(rbd (Ceph), **swift** 또는 **file**)를 정의합니다.	rbd
GnocchiBackend	원격 분석 서비스에서 사용할 백엔드(rbd (Ceph), **swift** 또는 **file**)를 정의합니다.	rbd

참고

기본 설정을 사용하려는 경우 **~/templates/storage-config.yaml** 에서 옵션을 생략할 수 있습니다.

사용자 지정 환경 파일의 내용은 다음 섹션에서 적용하는 설정에 따라 변경됩니다. 완료된 예는 부록 A. *샘플 환경 파일: Ceph Storage 클러스터 생성* 을 참조하십시오.

다음 하위 섹션에는 director가 적용하는 공통 기본 스토리지 서비스 설정을 재정의하는 방법에 대한 정보가 포함되어 있습니다.

4.1. CEPH 메타데이터 서버 활성화

Ceph 메타데이터 서버(MDS)는 CephFS에 저장된 파일과 관련된 메타데이터를 관리하는 **ceph-mds** 데몬을 실행합니다. NFS를 통해 CephFS를 사용할 수 있습니다. NFS를 통한 CephFS 사용에 대한 자세한 내용은 공유 파일 시스템 서비스의 NFS 백엔드 가이드 및 CephFS를 통해 파일 시스템 가이드 및 CephFS를 참조하십시오.

참고

Red Hat은 공유 파일 시스템 서비스를 위한 NFS 백엔드를 통해 CephFS에서만 Ceph MDS 배포를 지원합니다.

절차

오버클라우드를 생성할 때 Ceph 메타데이터 서버를 활성화하려면 다음 환경 파일을 호출합니다.

- **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml**

자세한 내용은 7.2절. "오버클라우드 배포 시작"의 내용을 참조하십시오. Ceph 메타데이터 서버에 대한 자세한 내용은 메타데이터 서버 데몬 구성을 참조하십시오.

참고

기본적으로 Ceph 메타데이터 서버는 컨트롤러 노드에 배포됩니다. 전용 노드에 Ceph 메타데이터 서버를 배포할 수 있습니다. 자세한 내용은 3.3절. "Ceph MDS 서비스에 대한 사용자 지정 역할 및 플레이버 만들기"의 내용을 참조하십시오.

4.2. CEPH OBJECT GATEWAY 활성화

Ceph 개체 게이트웨이(RGW)는 애플리케이션에 Ceph Storage 클러스터 내의 오브젝트 스토리지 기능에 대한 인터페이스를 제공합니다. RGW를 배포할 때 기본 Object Storage 서비스(**swift**)를 Ceph로 바꿀 수 있습니다. 자세한 내용은 Object Gateway Configuration and Administration Guide를 참조하십시오 .

절차

배포에서 RGW를 활성화하려면 오버클라우드를 생성할 때 다음 환경 파일을 호출합니다.

- **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml**

자세한 내용은 7.2절. "오버클라우드 배포 시작"의 내용을 참조하십시오.

기본적으로 Ceph Storage는 OSD당 250개의 배치 그룹을 허용합니다. RGW를 활성화하면 Ceph Storage는 RGW에 필요한 6개의 추가 풀을 생성합니다. 새 풀은 다음과 같습니다.

- .rgw.root
- default.rgw.control
- default.rgw.meta
- default.rgw.log
- default.rgw.buckets.index
- default.rgw.buckets.data

참고

배포에서 **default** 는 풀이 속하는 영역의 이름으로 교체됩니다.

따라서 RGW를 활성화하면 **CephPoolDefaultPgNum** 매개변수를 사용하여 새 풀을 고려하여 기본 **pg_num** 을 설정합니다. Ceph 풀의 배치 그룹 수를 계산하는 방법에 대한 자세한 내용은 5.4절. "다른 Ceph 풀에 사용자 지정 속성 할당" 을 참조하십시오.

Ceph 오브젝트 게이트웨이는 기본 Object Storage 서비스를 직접 대체합니다. 따라서 일반적으로 **swift** 를 사용하는 기타 모든 서비스는 추가 구성 없이 Ceph Object Gateway를 원활하게 사용할 수 있습니다. 자세한 내용은 블록 스토리지 백업 가이드를 참조하십시오.

4.3. 외부 CEPH OBJECT GATEWAY를 사용하도록 CEPH 개체 저장소 구성

RHOSP(Red Hat OpenStack Platform) director는 외부 Ceph Object Gateway(RGW)를 오브젝트 스토어 서비스로 구성하는 작업을 지원합니다. 외부 RGW 서비스로 인증하려면 ID 서비스(keystone)에서 사용자와 해당 역할을 확인하도록 RGW를 구성해야 합니다.

외부 Ceph Object Gateway를 구성하는 방법에 대한 자세한 내용은 Ceph Object Gateway 사용 가이드에서 Keystone 인증을 사용하도록 *Ceph Object Gateway 구성을 참조하십시오*.

절차

1. 다음 **parameter_defaults** 를 사용자 지정 환경 파일(예: **swift-external-params.yaml**)에 추가하고 배포에 맞게 값을 조정합니다.

```
parameter_defaults:
  ExternalSwiftPublicUrl: 'http://<Public RGW endpoint or
loadbalancer>:8080/swift/v1/AUTH_%(project_id)s'
  ExternalSwiftInternalUrl: 'http://<Internal RGW endpoint>:8080/swift/v1/AUTH_%
(project_id)s'
  ExternalSwiftAdminUrl: 'http://<Admin RGW endpoint>:8080/swift/v1/AUTH_%(project_id)s'
  ExternalSwiftUserTenant: 'service'
  SwiftPassword: 'choose_a_random_password'
```

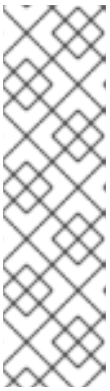
참고

예제 코드 스니펫에는 해당 환경에서 사용하는 값과 다를 수 있는 매개변수 값이 포함되어 있습니다.

- 원격 RGW 인스턴스가 수신 대기하는 기본 포트는 **8080** 입니다. 포트는 외부 RGW 구성 방법에 따라 다를 수 있습니다.
- Overcloud에서 생성된 **swift** 사용자는 **SwiftPassword** 매개 변수로 정의된 암호를 사용합니다. **rgw_keystone_admin_password** 를 사용하여 ID 서비스를 인증하도록 동일한 암호를 사용하도록 외부 RGW 인스턴스를 구성해야 합니다.

2. 다음 코드를 Ceph 구성 파일에 추가하여 ID 서비스를 사용하도록 RGW를 구성합니다. 환경에 맞게 변수 값을 바꿉니다.

```
    rgw_keystone_api_version = 3
    rgw_keystone_url = http://<public Keystone endpoint>:5000/
    rgw_keystone_accepted_roles = member, Member, admin
    rgw_keystone_accepted_admin_roles = ResellerAdmin, swiftoperator
    rgw_keystone_admin_domain = default
    rgw_keystone_admin_project = service
    rgw_keystone_admin_user = swift
    rgw_keystone_admin_password =
<password_as_defined_in_the_environment_parameters>
    rgw_keystone_implicit_tenants = true
    rgw_keystone_revocation_interval = 0
    rgw_s3_auth_use_keystone = true
    rgw_swift_versioning_enabled = true
    rgw_swift_account_in_url = true
```



참고

director는 기본적으로 ID 서비스에 다음 역할 및 사용자를 생성합니다.

- rgw_keystone_accepted_admin_roles: ResellerAdmin, swiftoperator
- rgw_keystone_admin_domain: default
- rgw_keystone_admin_project: service
- rgw_keystone_admin_user: swift

3. 배포와 관련된 기타 환경 파일을 사용하여 추가 환경 파일을 사용하여 오버클라우드를 배포합니다.

```
openstack overcloud deploy --templates \
-e <your_environment_files>
-e /usr/share/openstack-tripleo-heat-templates/environments/swift-external.yaml
-e swift-external-params.yaml
```

검증

1. **stack** 사용자로 언더클라우드에 로그인합니다.

2. **overcloudrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. 엔드포인트가 ID 서비스(keystone)에 있는지 확인합니다.

```
$ openstack endpoint list --service object-store

+---------+-----------+-------+-------+---------+-----------+---------------+
| ID | Region    | Service Name | Service Type | Enabled | Interface | URL |
+---------+-----------+-------+-------+---------+-----------+---------------+
| 233b7ea32aaf40c1ad782c696128aa0e | regionOne | swift | object-store | True    | admin     |
http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
| 4ccde35ac76444d7bb82c5816a97abd8 | regionOne | swift | object-store | True    | public    |
https://192.168.24.2:13808/v1/AUTH_%(project_id)s |
| b4ff283f445348639864f560aa2b2b41 | regionOne | swift | object-store | True    | internal  |
http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
+---------+-----------+-------+-------+---------+-----------+---------------+
```

4. 테스트 컨테이너를 생성합니다.

```
$ openstack container create <testcontainer>
+---------------+--------------+----------------------------------+
| account | container | x-trans-id |
+---------------+--------------+----------------------------------+
| AUTH_2852da3cf2fc490081114c434d1fc157 | testcontainer | tx6f5253e710a2449b8ef7e-
005f2d29e8 |
+---------------+--------------+----------------------------------+
```

5. 구성 파일을 생성하여 데이터를 컨테이너에 업로드할 수 있는지 확인합니다.

```
$ openstack object create testcontainer undercloud.conf
+----------------+--------------+----------------------------------+
| object         | container    | etag                             |
+----------------+--------------+----------------------------------+
| undercloud.conf | testcontainer | 09fcffe126cac1dbac7b89b8fd7a3e4b |
+----------------+--------------+----------------------------------+
```

6. 테스트 컨테이너를 삭제합니다.

```
$ openstack container delete -r <testcontainer>
```

4.4. CEPH를 사용하도록 백업 서비스 구성

블록 스토리지 백업 서비스(cinder**-backup**)는 기본적으로 비활성화되어 있습니다. 블록 스토리지 백업 서비스를 활성화하려면 다음 단계를 완료합니다.

절차

오버클라우드를 생성할 때 다음 환경 파일을 호출합니다.

- **/usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml**

4.5. CEPH 노드에 여러 개의 본딩 인터페이스 구성

본딩된 인터페이스를 사용하여 여러 NIC를 결합하고 네트워크 연결에 중복을 추가합니다. Ceph 노드에 NIC가 충분한 경우 각 노드에 여러 개의 결합된 인터페이스를 생성하여 중복 기능을 확장할 수 있습니다.

그런 다음 노드에 필요한 각 네트워크 연결에 결합된 인터페이스를 사용할 수 있습니다. 이를 통해 각 네트워크에 대한 중복 및 전용 연결을 모두 제공합니다.

결합 인터페이스의 가장 간단한 구현에는 Ceph 노드에서 사용하는 각 스토리지 네트워크에 하나씩 두 개의 본딩을 사용하는 것이 포함됩니다. 이러한 네트워크는 다음과 같습니다.

프런트 엔드 스토리지 네트워크 (StorageNet)

Ceph 클라이언트는 이 네트워크를 사용하여 해당 Ceph 클러스터와 상호 작용합니다.

백엔드 스토리지 네트워크(StorageMgmtNet)

Ceph 클러스터는 이 네트워크를 사용하여 클러스터의 배치 그룹 정책에 따라 데이터의 균형을 조정합니다. 자세한 내용은 *Red Hat Ceph 아키텍처 가이드의 PG (배치 그룹) 를 참조하십시오*.

여러 개의 본딩된 인터페이스를 설정하려면 director에서 여러 NIC를 배포하는 데 사용할 수 있는 샘플 템플릿을 제공하지 않으므로 새 네트워크 인터페이스 템플릿을 생성해야 합니다. 그러나 director는 단일 본딩 인터페이스를 배포하는 템플릿을 제공합니다. 이 템플릿은 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 입니다. 이 템플릿에서 추가 NIC에 대해 연결된 추가 인터페이스를 정의할 수 있습니다.

참고

사용자 지정 인터페이스 템플릿 생성에 대한 자세한 내용은 *Advanced Overcloud Customization* 가이드에서 사용자 지정 인터페이스 템플릿 만들기 를 참조하십시오.

다음 코드 조각에는 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml 파일에 정의된 단일 본딩 인터페이스의 기본 정의가 포함되어 있습니다.**

```
type: ovs_bridge // 1
name: br-bond
members:
  -
    type: ovs_bond // 2
    name: bond1 // 3
    ovs_options: {get_param: BondInterfaceOvsOptions} 4
    members: // 5
      -
        type: interface
        name: nic2
        primary: true
      -
        type: interface
        name: nic3
  -
    type: vlan // 6
    device: bond1 // 7
    vlan_id: {get_param: StorageNetworkVlanID}
    addresses:
      -
        ip_netmask: {get_param: StorageIpSubnet}
```

```
-
  type: vlan
  device: bond1
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  addresses:
    -
      ip_netmask: {get_param: StorageMgmtIpSubnet}
```

1. **br-bond** 라는 단일 브리지는 이 템플릿에 정의된 본딩을 보유합니다. 이 행은 OVS라는 브리지 유형을 정의합니다.
2. **br-bond** 브리지의 첫 번째 멤버는 **bond1** 이라는 본딩 인터페이스 자체입니다. 이 행은 또한 OVS인 **bond1의 본딩** 유형을 정의합니다.
3. 기본 본딩의 이름은 **bond1** 입니다.
4. **ovs_options** 항목은 director에 특정 본딩 모듈 지시문을 사용하도록 지시합니다. 이러한 지시문은 **BondInterfaceOvsOptions** 를 통해 전달됩니다. 이 지시문은 이 파일에서 구성할 수도 있습니다. 본딩 모듈 지시문 구성에 대한 자세한 내용은 4.5.1절. "본딩 모듈 지시문 구성" 을 참조하십시오.
5. 본딩의 **members** 섹션에서는 **bond1** 에서 본딩하는 네트워크 인터페이스를 정의합니다. 이 예에서 본딩된 인터페이스는 **nic2(** 기본 인터페이스로 설정) 및 **nic3** 을 사용합니다.
6. **br-bond** 브리지에는 두 개의 다른 구성원, 즉 프런트 엔드(**StorageNetwork**) 및 백엔드 (**StorageMgmtNetwork**) 스토리지 네트워크 모두의 VLAN이 있습니다.
7. **device** 매개 변수는 VLAN에서 사용해야 하는 장치를 정의합니다. 이 예에서 두 VLAN 모두 본딩된 인터페이스 **bond1을** 사용합니다.

NIC가 2개 이상 더 있으면 추가 브리지 및 본딩 인터페이스를 정의할 수 있습니다. 그런 다음 VLAN 중 하나를 새 본딩 인터페이스로 이동하여 두 스토리지 네트워크 연결의 처리량과 안정성을 높일 수 있습니다.

이러한 목적을 위해 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 파일을 사용자 지정하면 기본 OVS 대신 Linux 본딩(type **: linux_bond**)을 사용할 것을 권장합니다(**유형 : ovs_bond**). 이 본딩 유형은 엔터프라이즈 프로덕션 배포에 더 적합합니다.

다음 편집된 코드 조각은 bond**2라는 새 Linux 본딩이 포함된 추가 OVS 브리지(br-bond**2)를 정의합니다 **. bond2** 인터페이스는 두 개의 추가 NIC인 **nic4** 및 **nic5** 를 사용하며 백엔드 스토리지 네트워크 트래픽에만 사용됩니다.

```
type: ovs_bridge
name: br-bond
members:
  -
    type: linux_bond
    name: bond1
    bonding_options: {get_param: BondInterfaceOvsOptions} // 1
    members:
      -
        type: interface
        name: nic2
        primary: true
      -
        type: interface
        name: nic3
```

```
    -
      type: vlan
      device: bond1
      vlan_id: {get_param: StorageNetworkVlanID}
      addresses:
        -
          ip_netmask: {get_param: StorageIpSubnet}
  -
    type: ovs_bridge
    name: br-bond2
    members:
      -
        type: linux_bond
        name: bond2
        bonding_options: {get_param: BondInterfaceOvsOptions}
        members:
          -
            type: interface
            name: nic4
            primary: true
          -
            type: interface
            name: nic5
      -
        type: vlan
        device: bond1
        vlan_id: {get_param: StorageMgmtNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: StorageMgmtIpSubnet}
```

1 **bond1** 및 **bond2** 는 둘 다(OVS 대신) Linux 본딩이므로 **ovs _options 대신 bonding _options** 를 사용하여 본딩 지시문을 설정합니다. 자세한 내용은 4.5.1절. "본딩 모듈 지시문 구성"의 내용을 참조하십시오.

이 사용자 지정 템플릿의 전체 내용은 부록 B. *샘플 사용자 정의 인터페이스 템플릿: 여러 개의 결합된 인터페이스* 을 참조하십시오.

4.5.1. 본딩 모듈 지시문 구성

본딩된 인터페이스를 추가하고 구성한 후 **BondInterfaceOvsOptions** 매개변수를 사용하여 각 결합된 인터페이스를 사용할 지시문을 설정합니다. 이 정보는 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 파일의 **parameters:** 섹션에서 확인할 수 있습니다. 다음 코드 조각은 이 매개변수의 기본 정의를 보여줍니다(즉, 비어 있음).

```
BondInterfaceOvsOptions:
    default: ''
    description: The ovs_options string for the bond interface. Set
                 things like lacp=active and/or bond_mode=balance-slb
                 using this option.
    type: string
```

필요한 옵션을 **default:** 행에 정의합니다. 예를 들어 802.3ad(모드 4) 및 LACP 속도 1(빠른)을 사용하려면 **'mode=4 lacp_rate=1'** 을 사용합니다.

-

```
BondInterfaceOvsOptions:
    default: 'mode=4 lacp_rate=1'
    description: The bonding_options string for the bond interface. Set
                 things like lacp=active and/or bond_mode=balance-slb
                 using this option.
    type: string
```

기타 지원되는 본딩 옵션에 대한 자세한 내용은 *Advanced Overcloud Optimization* 가이드의 Open vSwitch Bonding Options 를 참조하십시오. 사용자 지정된 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 템플릿의 전체 내용은 *부록 B. 샘플 사용자 정의 인터페이스 템플릿: 여러 개의 결합된 인터페이스* 을 참조하십시오.

5장. CEPH STORAGE 클러스터 사용자 지정

director는 기본 구성을 사용하여 컨테이너화된 Red Hat Ceph Storage를 배포합니다. 기본 설정을 재정의하여 Ceph Storage를 사용자 지정할 수 있습니다.

전제 조건

컨테이너화된 Ceph 스토리지를 배포하려면 오버클라우드 배포 중에 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 파일을 포함해야 합니다. 이 환경 파일은 다음 리소스를 정의합니다.

- **CephAnsibleDisksConfig** - 이 리소스는 Ceph Storage 노드 디스크 레이아웃을 매핑합니다. 자세한 내용은 5.3절. "Ceph Storage 노드 디스크 레이아웃 매핑" 의 내용을 참조하십시오.
- **CephConfigOverrides** - 이 리소스는 다른 모든 사용자 지정 설정을 Ceph Storage 클러스터에 적용합니다.

이러한 리소스를 사용하여 director에서 컨테이너화된 Ceph Storage의 기본값을 덮어씁니다.

절차

1. Red Hat Ceph Storage 4 Tools 리포지토리를 활성화합니다.

```
$ sudo subscription-manager repos --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms
```

2. 언더클라우드에 **ceph-ansible** 패키지를 설치합니다.

```
$ sudo dnf install ceph-ansible
```

3. Ceph Storage 클러스터를 사용자 지정하려면 새 환경 파일에서 사용자 지정 매개 변수를 정의합니다(예: **/home/stack/templates/ceph-config.yaml**). 환경 파일의 **parameter_defaults** 섹션에서 다음 구문을 사용하여 Ceph Storage 클러스터 설정을 적용할 수 있습니다.

```
parameter_defaults:
  CephConfigOverrides:
    section:
      KEY:VALUE
```

참고

CephConfigOverrides 매개변수를 **ceph.conf** 파일의 **[global]** 섹션과 [**osd], [mon], [client] 등의 다른 섹션에 적용할 수 있습니다**. 섹션을 지정하면 **키:값** 데이터가 지정된 섹션으로 이동합니다. 섹션을 지정하지 않으면 데이터는 기본적으로 **[global]** 섹션으로 이동합니다. Ceph Storage 구성, 사용자 지정 및 지원되는 매개변수에 대한 자세한 내용은 Red Hat Ceph Storage 구성 가이드를 참조하십시오 .

4. **KEY** 및 **VALUE** 를 적용하려는 Ceph 클러스터 설정으로 바꿉니다. 예를 들어 **글로벌 섹션에서 max _open_files** 는 **KEY** 이고 **131072** 는 해당 **VALUE** 입니다.

```
parameter_defaults:
  CephConfigOverrides:
    global:
```

```
    max_open_files: 131072
  osd:
    osd_scrub_during_recovery: false
```

이 구성으로 인해 Ceph 클러스터의 구성 파일에 정의된 다음 설정이 생성됩니다.

```
[global]
max_open_files = 131072
[osd]
osd_scrub_during_recovery = false
```

5.1. CEPH-ANSIBLE 그룹 변수 설정

ceph-anible 툴은 Ceph Storage 클러스터를 설치하고 관리하는 데 사용되는 플레이북입니다.

ceph-ansible 도구에는 해당 옵션에 대한 구성 옵션 및 기본 설정을 정의하는 **group_vars** 디렉터리가 있습니다. **group_vars** 디렉터리를 사용하여 Ceph Storage 매개 변수를 설정합니다.

group_vars 디렉터리에 대한 자세한 내용은 설치 가이드에서 Red Hat Ceph Storage 클러스터 설치를 *참조하십시오*.

director에서 변수 기본값을 변경하려면 **CephAnsibleExtraConfig** 매개변수를 사용하여 heat 환경 파일에 새 값을 전달합니다. 예를 들어 **ceph-ansible** 그룹 변수 **journal_size** 를 40960으로 설정하려면 다음 **journal_size** 정의로 환경 파일을 생성합니다.

```
parameter_defaults:
  CephAnsibleExtraConfig:
    journal_size: 40960
```

중요

덮어쓰기 매개 변수를 사용하여 **ceph-ansible** 그룹 변수를 변경합니다. 언더클라우드의 **/usr/share/ceph-ansible** 디렉터리에서 직접 그룹 변수를 편집하지 마십시오.

5.2. CEPH STORAGE를 사용하는 RED HAT OPENSTACK PLATFORM용 CEPH 컨테이너

외부 Ceph 클러스터에서도 Ceph를 사용하도록 OpenStack Platform을 구성하는 데 Ceph 컨테이너가 필요합니다. Red Hat Enterprise Linux 8과 호환되려면 RHOSP(Red Hat OpenStack Platform) 16에 Red Hat Ceph Storage 4가 필요합니다. Ceph Storage 4 컨테이너는 인증이 필요한 레지스트리인 registry.redhat.io에 호스팅됩니다.

heat 환경 매개변수 **ContainerImageRegistryCredentials** 를 사용하여 **registry.redhat.io** 에서 인증할 수 있습니다. 자세한 내용은 컨테이너 이미지 준비 매개변수를 참조하십시오.

5.3. CEPH STORAGE 노드 디스크 레이아웃 매핑

컨테이너화된 Ceph Storage를 배포할 때 디스크 레이아웃을 매핑하고 Ceph OSD 서비스의 전용 블록 장치를 지정해야 합니다. 이전에 만든 환경 파일에서 이 매핑을 수행하여 사용자 지정 Ceph 매개 변수 **/home/stack/templates/ceph-config.yaml** 을 정의할 수 있습니다.

parameter_defaults 의 **CephAnsibleDisksConfig** 리소스를 사용하여 디스크 레이아웃을 매핑합니다. 이 리소스는 다음 변수를 사용합니다.

Variable	필수 여부	기본값(설정되지 않은 경우)	설명
osd_scenario	있음	lvm 알림: 기본값은 **lvm** 입니다.	**lvm** 값을 사용하면 **ceph-volume** 을 사용하여 OSD, **block.db** 및 BlueStore WAL 장치를 구성할 수 있습니다.
장치	있음	없음. 변수를 설정해야 합니다.	노드의 OSD에 사용할 블록 장치 목록입니다.
dedicated_devices	예 (osd **_scenario** 가 **할당되지 않은**경우에만)	장치	**devices** 매개 변수의 각 항목을 전용 저널링 블록 장치에 매핑하는 블록 장치 목록입니다. **osd_scenario=non-collocated** 경우에만 이 변수를 사용할 수 있습니다.
dmcrypt	없음	false	OSD에 저장된 데이터 (**true**) 또는 암호화되지 않은(false) 여부를**설정합니다**.
osd_objectstore	없음	블루스토어 알림: 기본값은 **bluestore** 입니다.	Ceph에서 사용하는 스토리지 백엔드를 설정합니다. 알림: 값이 기본적으로 **bluestore로 지정되지만**, 집계된 또는 비축정된 시나리오에서 **osd_scenario** 를 **filestore** 로 설정할 수 있습니다. **dedicated_devices** 가 저널링 디스크를 식별하는 비차별 시나리오에서 값을 **filestore** 로 설정할 수 있습니다. 장치에 정의된 디스크를 **파티셔닝하고 동일한 장치에 OSD 데이터와 저널링 데이터를 저장하는 집계된 시나리오에서 값을 filestore** 로 설정할 수 있습니다.

5.3.1. BlueStore 사용

절차

절차

1. Ceph OSD로 사용할 블록 장치를 지정하려면 다음 스니펫의 변형을 사용합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/nvme0n1
    osd_scenario: lvm
    osd_objectstore: bluestore
```

2. **/dev/nvme0n1** 은 고성능 장치 클래스에 있으므로 예제 **parameter_defaults** 는 **/dev/sdb, /dev/sdc 및 /dev/ sdd** 에서 실행되는 OSD 3개를 생성합니다. 3개의 OSD는 **/dev/nvme0n1** 을 **block.db** 및 BlueStore WAL 장치로 사용합니다. **ceph-volume** 툴은 **batch** 하위 명령을 사용하여 이 작업을 수행합니다. 각 Ceph Storage 노드에 대해 동일한 구성이 중복되며 일관된 하드웨어가 가정됩니다. **block.db** 및 BlueStore WAL 데이터가 OSD와 동일한 디스크에 있는 경우 매개변수 기본값을 다음과 같이 변경합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

5.3.2. 영구 이름이 있는 장치 참조

절차

1. 일부 노드에서는 재부팅하는 동안 **/dev/sdb** 및 **/dev/sdc** 와 같은 디스크 경로가 동일한 블록 장치를 가리키지 않을 수 있습니다. Ceph Storage 노드의 경우 **/dev/disk/by-path/** symlink로 각 디스크를 지정하여 배포 전체에서 일관된 블록 장치 매핑을 보장합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:10:0
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:11:0

    dedicated_devices:
      - /dev/nvme0n1
      - /dev/nvme0n1
```

2. 선택 사항: 오버클라우드를 배포하기 전에 OSD 장치 목록을 설정해야 하므로 디스크 장치의 PCI 경로를 식별하고 설정할 수 없습니다. 이 경우 인트로스펙션 중 블록 장치에 대한 **/dev/disk/by-path/symlink** 데이터를 수집합니다.
   다음 예제에서는 첫 번째 명령을 실행하여 **b08-h03-r620-hci 서버에 대한 언더클라우드 오브젝트 스토리지 서비스(swift)에서 인트로스펙션 데이터를 다운로드하고 b08-h03-r620-hci.json 이라는 파일에 데이터를 저장합니다**. "by-path"에 대해 grep에 대해 두 번째 명령을 실행합니다. 이

명령의 출력에는 디스크를 식별하는 데 사용할 수 있는 고유한 **/dev/disk/by-path** 값이 포함되어 있습니다.

```
(undercloud) [stack@b08-h02-r620 ironic]$ openstack baremetal introspection data save b08-h03-r620-hci | jq . > b08-h03-r620-hci.json
(undercloud) [stack@b08-h02-r620 ironic]$ grep by-path b08-h03-r620-hci.json
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:0:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:1:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:3:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:4:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:5:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:6:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:7:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:0:0",
```

스토리지 장치의 이름 지정 규칙에 대한 자세한 내용은 스토리지 장치 *관리 가이드 의 영구 명명 속성 개요 를 참조하십시오*.

5.3.3. 고급 시나리오에서 OSD 구성

환경 파일에서 **CephAnsibleDisksConfig** 리소스의 **devices** 변수에서 OSD에 사용할 블록 장치를 나열합니다.

다른 **장치** 구성 매개 변수 없이 device 변수를 사용하는 경우 **ceph-volume lvm 배치는** 느린 장치에 대해 더 높은 성능 장치를 **block.db** 로 균등하게 공유하여 OSD 구성을 자동으로 최적화합니다.

다음 절차에 따라 **ceph-volume lvm 배치** 모드에서 실행되지 않도록 **장치를** 구성할 수 있습니다.

5.3.3.1. block.db를 사용하여 성능 개선

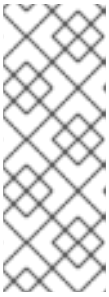
block.db 를 사용하면 처리량을 높이고 응답 시간을 개선하여 Ceph Storage 클러스터의 성능을 향상시킬 수 있습니다. **block.db** 는 데이터 세그먼트와 BlueStore 쓰기-ahead 로그(WAL)로 구성된 데이터베이스입니다.

절차

1. 환경 파일에 다음 내용을 추가합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sda
      - /dev/sdb
      - /dev/nvme0n1
      - /dev/sdc
      - /dev/sdd
      - /dev/nvme0n2
    osd_scenario: lvm
    osd_objectstore: bluestore
```

그러면 4개의 OSD, **sdb ,sdc, sdd 가 구성됩니다. 각 쌍에는 자체 데이터베이스 nvem0n1 및 nvme0n2 가 있습니다.**



참고

devices 목록의 장치 순서가 중요합니다. 드라이브와 **block.db** 및 **BlueStore WAL(DB-WAL)** 장치를 차례로 나열합니다. 이 예에서 **nvme0n1** 은 **DB-WAL for sda** 및 **sdb** 이며 **nvme0n2** 는**dc** 및 **sdd** 의 **DB-WAL**입니다. 자세한 내용은 **BlueStore** 사용을 참조하십시오.

2. 오버클라우드를 배포할 때 **-e** 옵션을 사용하여 배포 명령에 새 콘텐츠가 포함된 환경 파일을 포함합니다.

5.3.3.2. 전용 WAL(write-ahead log) 장치 사용

전용 **WAL(write-ahead log)** 장치를 지정할 수 있습니다. 장치,**dedicated_devices** 및 **bluestore_wal_devices** 를 함께 사용하면 **OSD**의 모든 구성 요소를 에 분리하여 성능을 향상시킬 수 있습니다.

다음 예제 절차에서 또 다른 사전 **bluestore_wal_devices** 는 **NVMe** 장치 **nvme0n1** 및 **nvme0n 2** 에서 쓰기**-ahead** 로그를 격리합니다.

절차

1. 환경 파일에 다음 내용을 추가합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sda
      - /dev/sdb
    dedicated_devices:
      - /dev/sdx
      - /dev/sdy
    bluestore_wal_devices:
      - /dev/nvme0n1
      - /dev/nvme0n2
```

2. 오버클라우드를 배포할 때 **-e** 옵션을 사용하여 배포 명령에 새 콘텐츠가 포함된 환경 파일을 포함합니다.

5.3.3.3. 사전 생성된 LVM을 사용하여 제어 강화

이전의 고급 시나리오에서 **ceph-volume** 은 다양한 유형의 장치 목록을 사용하여 **OSD**의 논리 볼륨을

생성합니다. **ceph-volume**이 실행되기 전에 논리 볼륨을 생성한 다음 **ceph- volumes** 를 해당 논리 볼륨의 **lvm_volumes** 목록에 전달할 수도 있습니다. 이 경우 논리 볼륨을 사전에 생성해야 하지만 보다 정확한 제어가 가능합니다. **director**는 하드웨어 프로비저닝도 담당하므로 최초 부팅 스크립트를 사용하여 이러한 **LVM**을 사전에 생성해야 합니다.

절차

1. **OS::TripleO::NodeUserData** 리소스 유형으로 **Heat** 템플릿을 등록하고 다음 콘텐츠를 포함하는 환경 파일 **/home/stack/templates/firstboot.yaml** 을 생성합니다.

```
resource_registry:
  OS::TripleO::NodeUserData: /home/stack/templates/ceph-lvm.yaml
```

2. 환경 파일 **/home/stack/templates/ceph-lvm.yaml** 을 생성합니다. 세 개의 물리 볼륨을 포함하는 다음 예제와 유사한 목록을 추가합니다. 장치 목록이 길어지면 요구 사항에 따라 예제를 확장합니다.

```
heat_template_version: 2014-10-16

description: >
  Extra hostname configuration

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: ceph_lvm_config}

  ceph_lvm_config:
    type: OS::Heat::SoftwareConfig
    properties:
      config: |
        #!/bin/bash -x
        pvcreate /dev/sda
        vgcreate ceph_vg_hdd /dev/sda
        pvcreate /dev/sdb
        vgcreate ceph_vg_ssd /dev/sdb
        pvcreate /dev/nvme0n1
        vgcreate ceph_vg_nvme /dev/nvme0n1
        lvcreate -n ceph_lv_wal1 -L 50G ceph_vg_nvme
        lvcreate -n ceph_lv_db1 -L 500G ceph_vg_ssd
        lvcreate -n ceph_lv_data1 -L 5T ceph_vg_hdd
        lvs

outputs:
  OS::stack_id:
    value: {get_resource: userdata}
```

3.

다음과 같은 방법으로 **devices** 목록 대신 **lvm_volumes** 매개 변수를 사용합니다. 이는 볼륨 그룹과 논리 볼륨이 이미 생성되었다고 가정합니다. 이 시나리오의 일반적인 사용 사례는 **WAL** 및 **DB LV**가 **SSD**에 있고 **data LV**는 **HDD**에 있다는 것입니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    osd_objectstore: bluestore
    osd_scenario: lvm
    lvm_volumes:
      - data: ceph_lv_data1
        data_vg: ceph_vg_hdd
        db: ceph_lv_db1
        db_vg: ceph_vg_ssd
        wal: ceph_lv_wal1
        wal_vg: ceph_vg_nvme
```

4. 오버클라우드를 배포할 때 **-e** 옵션을 사용하여 배포 명령에 새 콘텐츠가 포함된 환경 파일을 포함합니다.

참고

별도의 **WAL** 장치를 지정하려면 **WAL** 장치가 **DB** 장치보다 나은 하드웨어에 있는 경우에만 필요합니다. 일반적으로 별도의 **DB** 장치를 만드는 것이 충분하며 **WAL** 기능에 동일한 파티션을 사용합니다.

5.4. 다른 CEPH 풀에 사용자 지정 속성 할당

기본적으로 **director**로 생성된 **Ceph Storage** 풀에는 배치 그룹(**pg_num** 및 **pg p_num**)과 크기가 동일합니다. *5장. Ceph Storage 클러스터 사용자 지정* 의 두 방법 중 하나를 사용하여 이러한 설정을 전역적으로 재정의할 수 있습니다. 이렇게 하면 모든 풀에 동일한 값이 적용됩니다.

CephPools 매개변수를 사용하여 각 **Ceph Storage** 풀에 다른 속성을 적용하거나 새 사용자 지정 풀을 생성합니다.

절차

1. **POOL** 을 구성하려는 풀 이름으로 교체합니다.

```
parameter_defaults:
  CephPools:
    - name: POOL
```

2. 다음 중 하나를 수행하여 배치 그룹을 구성합니다.

- 기본 설정을 수동으로 재정의하려면 **pg_num** 을 배치 그룹 수로 설정합니다.

```
parameter_defaults:
  CephPools:
    - name: POOL
      pg_num: 128
      application: rbd
```

- 또는 배치 그룹을 자동으로 확장하려면 **pg_autoscale_mode** 를 **True** 로 설정하고 **target_size_ratio** 를 예상 **Ceph** 스토리지 요구 사항에 비해 백분율로 설정합니다.

```
parameter_defaults:
  CephPools:
    - name: POOL
      pg_autoscale_mode: True
      target_size_ratio: PERCENTAGE
      application: rbd
```

PERCENTAGE 를 **10**진수로 바꿉니다. 예를 들어 **0.5**는 **50%**와 같습니다. 총 백분율은 **1.0** 또는 **100 %**와 같아야합니다.

예를 들어 다음 값은 다음과 같습니다.

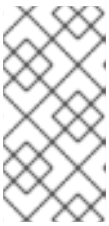
```
paramter_defaults:
  CephPools:
    - {"name": backups, "target_size_ratio": 0.1, "pg_autoscale_mode": True, "application":
rbd}
    - {"name": volumes, "target_size_ratio": 0.5, "pg_autoscale_mode": True, "application":
rbd}
    - {"name": vms,     "target_size_ratio": 0.2, "pg_autoscale_mode": True, "application":
rbd}
    - {"name": images,  "target_size_ratio": 0.2, "pg_autoscale_mode": True, "application":
rbd}
```

자세한 내용은 ***Red Hat Ceph Storage*** *설치 가이드* *의 배치 그룹 자동 확장기* *를 참조하십시오.*

3. 애플리케이션 유형을 지정합니다.

Compute, Block Storage, Image Storage의 애플리케이션 유형은'rbd'입니다. 그러나 풀을 사용하는 항목에 따라 다른 애플리케이션 유형을 지정할 수 있습니다.

예를 들어 gnocchi 지표 풀의 애플리케이션 유형은 openstack_gnocchi 입니다. 자세한 내용은 Enable Application in the *Storage Strategies Guide* 를 참조하십시오.



참고

CephPools 매개변수를 사용하지 않는 경우 director는 기본 풀 목록에 대해서만 적절한 애플리케이션 유형을 자동으로 설정합니다.

4. 선택 사항: custompool 이라는 풀을 추가하여 사용자 지정 풀을 생성하고 환경 요구 사항에 맞는 매개변수를 설정합니다.

```
parameter_defaults:
  CephPools:
    - name: custompool
      pg_num: 128
      application: rbd
```

작은 정보

일반적인 Ceph 사용 사례의 일반적인 풀 구성은 풀당 PG(배치 그룹) 계산기 를 참조하십시오. 이 계산기는 일반적으로 Ceph 풀을 수동으로 구성하기 위한 명령을 생성하는 데 사용됩니다. 이 배포에서는 director가 사양에 따라 풀을 구성합니다.

주의

Red Hat Ceph Storage 3(최소)에서는 OSD가 보유할 수 있는 최대 PG 수에 하드 제한을 도입했으며, 이는 기본적으로 200개입니다. 200 이상으로 이 매개변수를 재정의하지 마십시오. Ceph PG 번호가 최대값을 초과하기 때문에 문제가 있는 경우 mon_max _pg_per_osd가 아닌 풀당 pg_ num 을 조정하여 문제를 해결합니다.

5.5. 유사한 CEPH STORAGE 노드의 매개변수 덮어쓰기

CephStorage 또는 ComputeHCI 와 같이 Ceph OSD를 호스팅하는 역할이 있는 모든 노드는 5.3절. “Ceph Storage 노드 디스크 레이아웃 매핑” 에서 생성된 글로벌 장치 및 dedicated_devices 목록을 사

용합니다. 이 목록은 이러한 모든 서버에 동일한 하드웨어가 있다고 가정합니다. 하드웨어가 있는 서버가 있는 경우 이는 동일하지 않은 경우 노드별 디스크 구성을 사용하여 다른 장치 및 **dedicated_ devices** 목록에 대한 세부 정보로 **director**를 업데이트해야 합니다.

참고

Ceph OSD를 호스팅하는 역할에는 **roles_data.yaml** 파일에 **OS::TripleO::Services::CephOSD** 서비스가 포함됩니다.

다른 노드와 동일한 하드웨어가 없는 **Ceph Storage** 노드는 성능 문제를 일으킬 수 있습니다. **RHOSP(Red Hat OpenStack Platform)** 환경에서 노드별 재정의로 구성하는 표준 노드와 노드 간에 차이가 많을수록 성능이 저하됩니다.

5.5.1. 노드별 디스크 구성

동일한 하드웨어가 없는 의 서비스에 대해 **director**를 구성해야 합니다. 이를 노드별 디스크 구성이라고 합니다.

다음 방법 중 하나를 사용하여 노드별 디스크 구성을 생성할 수 있습니다.

- 자동: **JSON heat** 환경 파일을 생성하여 노드별 디스크 구성을 자동으로 생성할 수 있습니다.
- **Manual:** 노드 디스크 레이아웃을 변경하여 노드별 디스크 구성을 만들 수 있습니다.

5.5.1.1. Ceph 장치용 JSON heat 환경 파일 생성

/usr/share/openstack-tripleo-heat-templates/tools/make_ceph_disk_list.py 스크립트를 사용하여 베어 메탈 프로비저닝 서비스(**ironic**)의 인트로스펙션 데이터에서 유효한 **JSON heat** 환경 파일을 자동으로 생성할 수 있습니다. 이 **JSON** 파일을 사용하여 노드별 디스크 구성을 **director**에 전달합니다.

절차

1. 배포하려는 **Ceph** 노드의 베어 메탈 프로비저닝 서비스에서 인트로스펙션 데이터를 내보냅니다.

```
openstack baremetal introspection data save oc0-ceph-0 > ceph0.json
openstack baremetal introspection data save oc0-ceph-1 > ceph1.json
...
```

2. 유틸리티를 **Undercloud**에서 **stack** 사용자의 홈 디렉터리에 복사하고 이를 사용하여 **node_data_lookup.json** 파일을 생성합니다.

```
./make_ceph_disk_list.py -i ceph*.json -o node_data_lookup.json -k by_path
```

3. 배포 중에 **NodeDataLookup** 만 정의할 수 있기 때문에 **Ceph OSD**를 호스팅하는 모든 노드에 대해 **openstack baremetal introspection data save** 명령에서 인트로스펙션 데이터 파일을 유틸리티로 전달합니다. **i** 옵션은 ***.json** 과 같은 표현식이나 파일 목록을 입력으로 사용할 수 있습니다.

k 옵션을 사용하여 **OSD** 디스크를 식별하는 데 사용할 베어 메탈 프로비저닝 디스크 구조의 키를 정의합니다. 재부팅하는 동안 항상 동일한 장치를 가리키지 않을 수 있는 **/dev/sdd** 와 같은 장치의 파일을 생성하므로 **name** 을 사용하지 마십시오. 대신 **by_path** 를 사용합니다. 이는 **-k** 를 지정하지 않는 경우 기본값입니다.

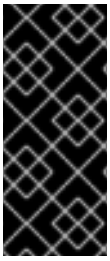
베어 메탈 프로비저닝 서비스는 시스템에서 사용 가능한 디스크 중 하나를 루트 디스크로 예약합니다. 유틸리티는 생성된 장치 목록에서 항상 루트 디스크를 제외합니다.

4. 선택 사항: 사용 가능한 다른 옵션을 보려면 **./make_ceph_disk_list.py -help** 를 사용할 수 있습니다.

5. 오버클라우드를 배포할 때 환경과 관련된 다른 환경 파일과 함께 **node_data_lookup.json** 파일을 포함합니다.

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e node_data_lookup.json \
…
```

5.5.1.2. Ceph Storage 노드의 디스크 레이아웃 변경



중요

동종이 아닌 **Ceph Storage** 노드는 성능 문제를 일으킬 수 있습니다. **RHOSP(Red Hat OpenStack Platform)** 환경에서 노드별 재정의로 구성하는 표준 노드와 노드 간에 차이가 많을수록 성능이 저하됩니다.

노드별 디스크 구성을 **director**에 전달하려면 **node-spec-overrides.yaml** 과 같은 **heat** 환경 파일을 **openstack overcloud deploy** 명령에 전달하고 파일 콘텐츠는 시스템 고유 **UUID**로 각 서버를 식별해야 글로벌 변수를 재정의해야 합니다.

개별 서버 또는 베어 메탈 프로비저닝 서비스(**ironic**) 데이터베이스에서 시스템 고유 **UUID**를 추출할 수 있습니다.

참고

다음 절차에서는 유효한 **JSON**이 포함된 유효한 **YAML** 환경 파일을 생성합니다. **make_ceph_disk_list.py** 를 사용하여 전체 **JSON** 파일을 생성하고 **YAML**처럼 배포 명령에 전달할 수도 있습니다. 자세한 내용은 **Ceph 장치에 대한 JSON heat 환경 파일 생성을** 참조하십시오.

절차

1. 개별 서버의 **UUID**를 찾으려면 서버에 로그인하여 다음 명령을 입력합니다.

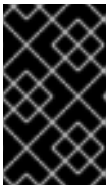
```
$ dmidecode -s system-uuid
```

2. 베어 메탈 프로비저닝 서비스 데이터베이스에서 **UUID**를 추출하려면 언더클라우드에 다음 명령을 입력합니다.

```
$ openstack baremetal introspection data save NODE-ID | jq .extra.system.product.uuid
```

주의

undercloud 설치 또는 업그레이드 및 인트로스펙션 전에 **undercloud.conf** 에 **inspection_extras = true** 가 없으면 시스템 고유 **UUID**가 베어 메탈 프로비저닝 서비스 데이터베이스에 없습니다.



중요

시스템 고유 **UUID**는 베어 메탈 프로비저닝 서비스 **UUID**가 아닙니다.

유효한 **node-spec-overrides.yaml** 파일은 다음과 같을 수 있습니다.

```
parameter_defaults:
  NodeDataLookup: {"32E87B4C-C4A7-418E-865B-191684A6883B": {"devices":
["/dev/sdc"]}}
```

3. 처음 두 줄 뒤의 모든 행은 유효한 **JSON**이어야 합니다. **jq** 명령을 사용하여 **JSON**이 유효한지 확인합니다.

   a. 파일에서 처음 두 줄(**parameter_defaults:** 및 **NodeDataLookup:**)을 일시적으로 제거합니다.

   b. **cat node-spec-overrides.yaml | jq .**

4. **node-spec-overrides.yaml** 파일이 증가함에 따라 **jq** 명령을 사용하여 포함된 **JSON**이 유효한지 확인할 수도 있습니다. 예를 들어, **devices** 및 **dedicated_devices** 목록이 동일한 길이여야 하므로 다음 명령을 사용하여 배포를 시작하기 전에 동일한 길이인지 확인합니다. 다음 예에서 **node-spec-c05-h17-h21-h25-6048r.yaml** 에는 슬롯 **h17, h21, h25**에 디스크가 누락된 랙 **c05**에 세 개의 서버가 있습니다.

```
(undercloud) [stack@b08-h02-r620 tht]$ cat node-spec-c05-h17-h21-h25-6048r.yaml | jq '.[] |
.devices | length'
33
30
33
(undercloud) [stack@b08-h02-r620 tht]$ cat node-spec-c05-h17-h21-h25-6048r.yaml | jq '.[] |
.dedicated_devices | length'
33
30
33
(undercloud) [stack@b08-h02-r620 tht]$
```

5. **JSON**이 검증되면 유효한 환경 **YAML** 파일(**parameter_defaults:** 및 **NodeDataLookup: **)으로 만드는 두 행을 다시 추가하고 배포에 **-e** 를 포함합니다. 아래 예제에서 업데이트된 **heat** 환경 파일은 **Ceph** 배포에 **NodeDataLookup** 을 사용합니다. 서버 목록에는 **35**개의 디스크가 있는 장치 목록이 있습니다. 단, 그 중 하나가 디스크가 누락되어 있습니다. 이 환경 파일은 해당 단일

노드에 대해서만 기본 장치 목록을 재정의하고 전역 목록 대신 사용해야 하는 34개의 디스크 목록을 제공합니다.

6. **JSON이 검증되면 유효한 환경 YAML 파일(parameter_defaults: 및 NodeDataLookup:)으로 만드는 두 행을 다시 추가하고 배포 명령에 -e 를 포함합니다.**

다음 예에서 업데이트된 heat 환경 파일은 Ceph 배포에 NodeDataLookup 을 사용합니다. 서버 목록에는 35개의 디스크가 있는 장치 목록이 있습니다. 단, 그 중 하나가 디스크가 누락되어 있습니다. 이 환경 파일은 해당 단일 노드의 기본 장치 목록을 재정의하고 글로벌 목록 대신 사용해야 하는 34개의 디스크 목록으로 노드를 제공합니다.

```
parameter_defaults:
  # c05-h01-6048r is missing scsi-0:2:35:0 (00000000-0000-0000-0000-0CC47A6EFD0C)
  NodeDataLookup: {
    "00000000-0000-0000-0000-0CC47A6EFD0C": {
      "devices": [
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:1:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:32:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:2:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:3:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:4:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:5:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:6:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:33:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:7:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:8:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:34:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:9:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:10:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:11:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:12:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:13:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:14:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:15:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:16:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:17:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:18:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:19:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:20:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:21:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:22:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:23:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:24:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:25:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:26:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:27:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:28:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:29:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:30:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:31:0"
        ],
      "dedicated_devices": [
```

```
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
"/dev/disk/by-path/pci-0000:84:00.0-nvme-1"
   ]
  }
}
```

5.5.2. BlueStore block.db 크기 변경

BlueStore block.db 는 데이터 세그먼트의 데이터베이스이며 BlueStore write-ahead 로그(WAL)입니다. 데이터베이스 크기를 변경하는 방법은 두 가지가 있습니다. 크기를 변경하려면 다음 방법 중 하나를 선택합니다.

5.5.2.1. ceph-volume을 사용할 때 BlueStore block.db 크기 변경

ceph-volume. ceph-volume 을 사용할 때 block.db 크기를 덮어쓰려면 다음 절차를 사용하십시오. osd_scenario: lvm.ceph-volume 이 block.db 크기를 자동으로 설정할 때 ceph-volume이 사용됩니다. 그러나 고급 시나리오의 block.db 크기를 재정의할 수 있습니다.

다음 예제에서는 **Ceph** 구성 파일이 재정의되지 않고 **ceph-ansible** 호스트 변수를 사용하여 사용되는 **block_db_size** 가 **ceph-volume** 호출에 전달되도록 합니다.

절차

1. 다음과 유사한 콘텐츠를 사용하여 **JSON** 환경 파일을 생성하되 요구 사항에 따라 값을 바꿉니다.

```
{
  "parameter_defaults": {
    "NodeDataLookup": {
      "32e87b4c-c4a7-41be-865b-191684a6883b": {
        "block_db_size": 3221225472
      },
      "ea6a84d6-cf89-4fe2-b7bd-869b3fe4dd6b": {
        "block_db_size": 3221225472
      }
    }
  }
}
```

2. 오버클라우드를 배포할 때 환경과 관련된 다른 환경 파일과 함께 **JSON** 파일을 포함합니다.

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e <json_environment_file>  \
…
```

5.5.2.2. ceph-disk를 사용할 때 BlueStore block.db 크기 변경

ceph-disk . ceph-disk . ceph-disk 를 사용할 때 다음 절차를 사용하여 **block.db** 를 덮어씁니다. **osd_scenario: non-col placed** 또는 **osd_scenario: col placed .ceph-disk** 가 사용됩니다.

다음 예제에서는 특정 노드에 대해 **Ceph** 구성 덮어쓰기를 사용하여 **blustore_block_db_size** 를 설정합니다. 이 **Ceph** 구성 옵션은 **ceph-volume** 을 사용할 때 무시되지만 **ceph-disk** 는 이 구성 옵션을 사용합니다.

절차

1. 다음과 유사한 콘텐츠를 사용하여 **JSON** 환경 파일을 생성하되 요구 사항에 따라 값을 바꿉니다.

```
{
  "parameter_defaults": {
    "NodeDataLookup": {
      "32e87b4c-c4a7-41be-865b-191684a6883b": {
        "ceph_conf_overrides": {
          "osd": {
            "bluestore_block_db_size": 3221225472
          }
        }
      },
      "ea6a84d6-cf89-4fe2-b7bd-869b3fe4dd6b": {
        "ceph_conf_overrides": {
          "osd": {
            "bluestore_block_db_size": 3221225472
          }
        }
      }
    }
  }
}
```

2. 오버클라우드를 배포할 때 환경과 관련된 다른 환경 파일과 함께 **JSON** 파일을 포함합니다.

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e <json_environment_file>  \
…
```

5.6. 대규모 CEPH 클러스터의 재시작 지연 증가

배포하는 동안 **OSD** 및 모니터와 같은 **Ceph** 서비스가 다시 시작되고 서비스가 다시 실행될 때까지 배포가 계속되지 않습니다. **Ansible**은 **15**초 동안 대기하고 서비스가 시작될 때까지 **5**번(재시도) 확인합니다. 서비스를 다시 시작하지 않으면 운영자가 개입할 수 있도록 배포가 중지됩니다.

Ceph 클러스터의 크기에 따라 재시도 또는 지연 값을 늘려야 할 수 있습니다. 이러한 매개변수 및 기본 값의 정확한 이름은 다음과 같습니다.

```
health_mon_check_retries: 5
health_mon_check_delay: 15
health_osd_check_retries: 5
health_osd_check_delay: 15
```

절차

1. **CephAnsibleExtraConfig** 매개변수를 업데이트하여 기본 지연 및 재시도 값을 변경합니다.

```
parameter_defaults:
  CephAnsibleExtraConfig:
    health_osd_check_delay: 40
    health_osd_check_retries: 30
    health_mon_check_delay: 20
    health_mon_check_retries: 10
```

이 예제에서는 클러스터가 **30**번 검사하고 **Ceph OSD**를 검사할 때마다 **40**초 동안 기다린 다음, 각 **Ceph MON**이 확인될 때까지 **10**초 동안 **20**번 기다렸다가 **10**초 동안 기다립니다.

2. 변경 사항을 통합하려면 **openstack overcloud deploy** 를 사용하여 **-e** 로 업데이트된 **yaml** 파일을 전달합니다.

5.7. ANSIBLE 환경 변수 재정의

Red Hat OpenStack Platform Workflow 서비스(**mistral**)는 **Ansible**을 사용하여 **Ceph Storage**를 구성하지만 **Ansible** 환경 변수를 사용하여 **Ansible** 환경을 사용자 지정할 수 있습니다.

절차

an ANSIBLE_* 환경 변수를 재정의하려면 **CephAnsibleEnvironmentVariables heat** 템플릿 매개변수를 사용합니다.

이 예제 구성에서는 포크 및 **SSH** 재시도 횟수가 증가합니다.

```
parameter_defaults:
  CephAnsibleEnvironmentVariables:
    ANSIBLE_SSH_RETRIES: '6'
    DEFAULT_FORKS: '35'
```

Ansible 환경 변수에 대한 자세한 내용은 **Ansible** 구성 설정을 참조하십시오.

Ceph Storage 클러스터를 사용자 지정하는 방법에 대한 자세한 내용은 **Ceph Storage** 클러스터 사용자 지정을 참조하십시오.

5.8. CEPH ON-WIRE 암호화 활성화

Red Hat Ceph Storage 4 이상에서는 네트워크를 통해 버전 **2** 프로토콜을 도입하여 네트워크를 통해 모든 **Ceph** 트래픽에 대해 암호화를 활성화할 수 있습니다. **v2**의 보안 모드 설정은 **Ceph** 데몬과 **Ceph** 클라이언트 간의 통신을 암호화하여 포괄적인 암호화를 제공합니다.

이 기능은 이번 릴리스에서 *기술 프리뷰*로 제공되므로 **Red Hat**에서 완전히 지원되지 않습니다. 테스트 용도로만 사용해야 하며 프로덕션 환경에 배포해서는 안 됩니다. 기술 프리뷰 기능에 대한 자세한 내용은 적용 범위 상세 정보를 참조하십시오.

참고

이 기능은 현재 기술 프리뷰이지만 **RHOSP(Red Hat OpenStack Platform)** 버전 **16.1** 이상과 함께 사용됩니다. 외부 **Red Hat Ceph Storage** 버전 **4**를 사용하는 **RHOSP** 버전 **13** 배포에서는 지원되지 않습니다. 자세한 내용은 ***Red Hat Ceph Storage Architecture Guide*의 *Ceph on-wire encryption*** *을 참조하십시오*.

RHOSP에서 **Ceph on-wire** 암호화를 활성화하려면 해당 환경의 오버라이드 파일에서 다음 매개변수를 설정합니다.

```
parameter_defaults:
  CephConfigOverrides:
    global:
      ms_cluster_mode: secure
      ms_service_mode: secure
      ms_client_mode: secure
```

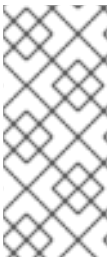
Ceph on-wire encryption에 대한 자세한 내용은 *아키텍처 가이드 의 Ceph on-wire encryption 을 참조하십시오*.

6장. DIRECTOR를 사용하여 CEPH STORAGE 클러스터에서 다양한 워크로드를 위한 성능 계층 정의

RHOSP(Red Hat OpenStack Platform) director를 사용하여 다양한 Red Hat Ceph Storage 성능 계층을 배포할 수 있습니다. Ceph CRUSH 규칙과 CephPools director 매개변수를 결합하여 장치 클래스 기능을 사용하고 다양한 계층을 빌드하여 성능 요구 사항이 다른 워크로드를 수용할 수 있습니다. 예를 들어 일반 워크로드에 대해 HDD 클래스와 고성능 로드를 위해 SSD에서만 데이터를 배포하는 SSD 클래스를 정의할 수 있습니다. 이 시나리오에서는 새 블록 스토리지 볼륨을 생성할 때 HDD 또는 SSD 중 하나의 성능 계층을 선택할 수 있습니다.

경고

기존 환경에 성능 계층을 정의하면 Ceph 클러스터에서 대용량 데이터 이동이 발생할 수 있습니다. 스택 업데이트 중에 director가 트리거하는 ceph-ansible 에는 풀이 클러스터에 이미 정의되어 있는지와 데이터가 포함되어 있는지 확인하는 논리가 없습니다. 즉, 풀과 연결된 기본 CRUSH 규칙을 변경하면 데이터 이동 때문에 기존 환경에서 성능 계층을 정의하는 것은 위험할 수 있습니다. 노드 추가 또는 제거에 대한 지원이나 권장 사항이 필요한 경우 Red Hat 지원에 문의하십시오.



참고

Ceph는 디스크 유형을 자동 감지하고 Linux 커널에서 노출하는 하드웨어 속성을 기반으로 해당 장치 클래스(HD, SSD 또는 NVMe)에 할당합니다. 그러나 필요에 따라 범주를 사용자 지정할 수도 있습니다.

사전 요구 사항

- 새 배포의 경우 RHCS(Red Hat Ceph Storage) 버전 4.1 이상입니다.
- 기존 배포의 경우 RHCS(Red Hat Ceph Storage) 버전 4.2 이상입니다.

다른 Red Hat Ceph Storage 성능 계층을 배포하려면 CRUSH 맵 세부 정보가 포함된 새 환경 파일을 생성한 다음 배포 명령에 포함합니다.

다음 절차에서 각 Ceph Storage 노드에는 3개의 OSD가 포함되어 있으며 sdb 및 sdc 는 디스크 회전이며 sdc 는 SSD입니다. Ceph는 올바른 디스크 유형을 자동으로 감지합니다. 그런 다음 두 개의 CRUSH 규칙(HDD 및 SSD)을 구성하여 두 개의 해당 장치 클래스에 매핑합니다. HDD 규칙은 기본값이며 다른 규칙을 사용하여 풀을 구성하지 않는 한 모든 풀에 적용됩니다.

마지막으로 fastpool 이라는 추가 풀을 생성하여 SSD 규칙에 매핑합니다. 이 풀은 궁극적으로 Block Storage(cinder) 백엔드를 통해 노출됩니다. 이 블록 스토리지 백엔드를 사용하는 모든 워크로드는 SSD에서 빠른 성능만 지원합니다. 이를 데이터 또는 볼륨에서 부팅하는 데 활용할 수 있습니다.

6.1. 성능 계층 구성

경고

기존 환경에 성능 계층을 정의하면 **Ceph** 클러스터에서 대용량 데이터 이동이 발생할 수 있습니다. 스택 업데이트 중에 **director**가 트리거하는 **ceph-ansible** 에는 풀이 클러스터에 이미 정의되어 있는지와 데이터가 포함되어 있는지 확인하는 논리가 없습니다. 즉, 풀과 연결된 기본 **CRUSH** 규칙을 변경하면 데이터 이동 때문에 기존 환경에서 성능 계층을 정의하는 것은 위험할 수 있습니다. 노드 추가 또는 제거에 대한 지원이나 권장 사항이 필요한 경우 **Red Hat** 지원에 문의하십시오.

director는 이 기능을 다룰 특정 매개 변수를 노출하지 않지만 다음 단계를 완료하여 **ceph-ansible** 예상 변수를 생성할 수 있습니다.

절차

1. **stack** 사용자로 **Undercloud** 노드에 로그인합니다.

2. **Ceph** 구성 매개 변수 및 장치 클래스 변수를 포함하도록 **/home/stack/templates/ceph-config.yaml** 과 같은 환경 파일을 생성합니다. 또는 기존 환경 파일에 다음 구성을 추가할 수 있습니다.

3. 환경 파일에서 **CephAnsibleDisksConfig** 매개변수를 사용하여 **Ceph OSD**로 사용하려는 블록 장치를 나열합니다.

```
CephAnsibleDisksConfig:
  devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
  osd_scenario: lvm
  osd_objectstore: bluestore
```

4. 선택 사항**: Ceph**는 디스크 유형을 자동으로 탐지하여 해당 장치 클래스에 할당합니다. 그러나 **authenticate _device_class** 속성을 사용하여 특정 장치가 특정 클래스에 속하거나 고유한 사용자 정의 클래스를 만들 수 있습니다. 다음 예제에는 지정된 클래스와 동일한 **OSD** 목록이 포함되어 있습니다.

```
CephAnsibleDisksConfig:
  lvm_volumes:
    - data: '/dev/sdb'
      crush_device_class: 'hdd'
    - data: '/dev/sdc'
      crush_device_class: 'hdd'
    - data: '/dev/sdd'
```

```
    crush_device_class: 'ssd'
  osd_scenario: lvm
  osd_objectstore: bluestore
```

5. **CephAnsibleExtraVars** 매개변수를 추가합니다. **assets _rules** 매개 변수에는 **Ceph**가 자동으로 탐지하는 각 클래스에 대한 규칙이 포함되어야 합니다. 새 풀을 만들 때 규칙이 지정되지 않은 경우 **Ceph**가 기본값으로 사용할 규칙을 선택합니다.

```
CephAnsibleExtraConfig:
  crush_rule_config: true
  create_crush_tree: true
  crush_rules:
    - name: HDD
      root: default
      type: host
      class: hdd
      default: true
    - name: SSD
      root: default
      type: host
      class: ssd
      default: false
```

6. **CephPools** 매개변수를 추가합니다.

- **rule_name** 매개 변수를 사용하여 기본 규칙을 사용하지 않는 각 풀의 계층을 지정합니다. 다음 예에서 **fastpool** 풀은 빠른 계층으로 구성된 **SSD** 장치 클래스를 사용하여 블록 스토리지 볼륨을 관리합니다.

- **<appropriate_PG_num>** 을 적절한 **PG(**배포 그룹**)** 수로 바꿉니다. 또는 배치 그룹 자동 확장기를 사용하여 **Ceph** 풀의 **PG** 수를 계산합니다.

  자세한 내용은 다른 **Ceph** 풀에 사용자 지정 속성 할당을 참조하십시오.

- **CinderRbdExtraPools** 매개변수를 사용하여 **fastpool** 을 블록 스토리지 백엔드로 구성합니다.

```
CephPools:
  - name: fastpool
    pg_num: <appropraiate_PG_num>
    rule_name: SSD
    application: rbd
CinderRbdExtraPools: fastpool
```

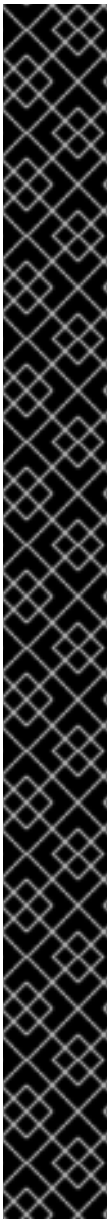
7. 다음 예제를 사용하여 환경 파일에 올바른 값이 포함되어 있는지 확인합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - '/dev/sdb'
      - '/dev/sdc'
      - '/dev/sdd'
    osd_scenario: lvm
    osd_objectstore: bluestore
  CephAnsibleExtraConfig:
    crush_rule_config: true
    create_crush_tree: true
    crush_rules:
      - name: HDD
        root: default
        type: host
        class: hdd
        default: true
      - name: SSD
        root: default
        type: host
        class: ssd
        default: false
  CinderRbdExtraPools: fastpool
  CephPools:
    - name: fastpool
      pg_num: <appropriate_PG_num>
      rule_name: SSD
      application: rbd
```

8. **openstack overcloud deploy** 명령에 새 환경 파일을 포함합니다.

```
$ openstack overcloud deploy \
--templates \
…
-e <other_overcloud_environment_files> \
-e /home/stack/templates/ceph-config.yaml  \
…
```

<other_overcloud_environment_files> 를 배포에 포함된 환경 파일 목록으로 바꿉니다.



중요

기존 **Ceph** 클러스터에 환경 파일을 적용하면 기존 **Ceph** 풀이 새 규칙에 따라 업데이트되지 않습니다. 따라서 규칙을 지정된 풀로 설정하려면 배포가 완료된 후 다음 명령을 입력해야 합니다.

```
$ ceph osd pool set <pool> crush_rule <rule>
```

- **<pool>** 을 새 규칙을 적용할 풀의 이름으로 바꿉니다.
- **<rule>** 을 지정한 규칙 이름 중 하나로 교체합니다.
- **<appropriate_PG_num>** 을 적절한 배치 그룹 수 또는 **target_size_ratio**로 바꾸고 **pg_ autoscale_mode** 를 **true** 로 설정합니다.

이 명령을 사용하여 변경할 모든 규칙에 대해 기존 항목을 업데이트하거나 기존 템플릿의 **CephPools** 매개변수에 새 항목을 추가합니다.

```
CephPools:
  - name: <pool>
    pg_num: <appropriate_PG_num>
    rule_name: <rule>
    application: rbd
```

6.2. BLOCK STORAGE(CINDER) 유형을 새 CEPH 풀에 매핑

경고

기존 환경에 성능 계층을 정의하면 **Ceph** 클러스터에서 대용량 데이터 이동이 발생할 수 있습니다. 스택 업데이트 중에 **director**가 트리거하는 **ceph-ansible** 에는 풀이 클러스터에 이미 정의되어 있는지와 데이터가 포함되어 있는지 확인하는 논리가 없습니다. 즉, 풀과 연결된 기본 **CRUSH** 규칙을 변경하면 데이터 이동 때문에 기존 환경에서 성능 계층을 정의하는 것은 위험할 수 있습니다. 노드 추가 또는 제거에 대한 지원이나 권장 사항이 필요한 경우 **Red Hat** 지원에 문의하십시오.

구성 단계를 완료한 후 **Block Storage(cinder)**를 사용하여 생성한 **fastpool** 계층에 매핑되는 유형을 생성하여 **RHOSP** 테넌트에서 성능 계층 기능을 사용하도록 설정합니다.

절차

1. **stack** 사용자로 **Undercloud** 노드에 로그인합니다.

2. **overcloudrc** 파일을 소싱합니다.

```
$ source overcloudrc
```

3. 블록 스토리지 볼륨 기존 유형을 확인합니다.

```
$ cinder type-list
```

4. 새 **Block Storage** 볼륨 **fast_tier**를 만듭니다.

```
$ cinder type-create fast_tier
```

5. **Block Storage** 유형이 생성되었는지 확인합니다.

```
$ cinder type-list
```

6. **fast_tier** 블록 스토리지 유형을 사용할 수 있으면 **fastpool** 을 생성한 새 계층의 블록 스토리지 볼륨 백엔드로 설정합니다.

```
$ cinder type-key fast_tier set volume_backend_name=tripleo_ceph_fastpool
```

7. 새 계층을 사용하여 새 볼륨을 생성합니다.

```
$ cinder create 1 --volume-type fast_tier --name fastdisk
```

6.3. CRUSH 규칙이 생성되고 풀이 올바른 CRUSH 규칙으로 설정되어 있는지 확인

경고

기존 환경에 성능 계층을 정의하면 **Ceph** 클러스터에서 대용량 데이터 이동이 발생할 수 있습니다. 스택 업데이트 중에 **director**가 트리거하는 **ceph-ansible** 에는 풀이 클러스터에 이미 정의되어 있는지와 데이터가 포함되어 있는지 확인하는 논리가 없습니다. 즉, 풀과 연결된 기본 **CRUSH** 규칙을 변경하면 데이터 이동 때문에 기존 환경에서 성능 계층을 정의하는 것은 위험할 수 있습니다. 노드 추가 또는 제거에 대한 지원이나 권장 사항이 필요한 경우 **Red Hat** 지원에 문의하십시오.

절차

1. 오버클라우드 컨트롤러 노드에 **heat-admin** 사용자로 로그인합니다.

2. **OSD** 계층이 성공적으로 설정되었는지 확인하려면 다음 명령을 입력합니다. **<controller_hostname>** 을 호스트 컨트롤러 노드 이름으로 바꿉니다.

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd tree
```

3. 결과 트리 보기에서 **CLASS** 열에 설정한 각 **OSD**에 대해 올바른 장치 클래스가 표시되는지 확인합니다.

4. 또한 다음 명령을 사용하여 **OSD**가 장치 클래스에 올바르게 할당되었는지 확인합니다. **<controller_hostname>** 을 호스트 컨트롤러 노드 이름으로 바꿉니다.

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd crush tree --show-shadow
```

5. 결과 계층 구조를 다음 명령의 결과와 비교하여 각 규칙에 대해 동일한 값이 적용되는지 확인합니다.

- **<controller_hostname>** 을 호스트 컨트롤러 노드 이름으로 바꿉니다.
- **<rule_name>** 을 확인하려는 규칙의 이름으로 바꿉니다.

```
$ sudo podman exec <controller_hostname> ceph osd crush rule dump <rule_name>
```

6. 생성한 규칙 이름과 **ID**가 배포 중에 사용한 **convention _rules** 매개변수에 따라 올바른지 확인합니다. **<controller_hostname>** 을 호스트 컨트롤러 노드 이름으로 바꿉니다.

```
$ sudo podman exec -it ceph-mon-<controller_hostname>  ceph osd crush rule dump | grep -E "rule_(id|name)"
```

7. **Ceph** 풀이 **3**단계에서 검색한 올바른 **CRUSH** 규칙 **ID**에 연결되어 있는지 확인합니다. **<controller_hostname>** 을 호스트 컨트롤러 노드 이름으로 바꿉니다.

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd dump | grep pool
```

8. **각 풀에 대해 규칙 ID가 예상 규칙 이름과 일치하는지 확인합니다.**

7장. 오버클라우드 생성

사용자 지정 환경 파일이 준비되면 각 역할이 사용하는 플레이버와 노드를 지정한 다음 배포를 실행할 수 있습니다. 다음 하위 섹션에서는 두 단계를 자세히 설명합니다.

7.1. 역할에 노드 및 플레이버 할당

Overcloud 배포를 계획하려면 각 역할에 할당할 노드 수와 플레이버를 지정해야 합니다. 모든 **Heat** 템플릿 매개변수와 마찬가지로 이러한 역할 사양은 환경 파일의 **parameter_defaults** 섹션에 선언됩니다(이 경우 **~/templates/storage-config.yaml**).

이를 위해 다음 매개변수를 사용합니다.

표 7.1. 오버클라우드 노드의 역할 및 플레이버

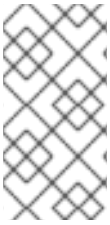
Heat 템플릿 매개변수	설명
ControllerCount	확장할 컨트롤러 노드 수
OvercloudControlFlavor	컨트롤러 노드에 사용할 플레이버(**제어**)
ComputeCount	확장할 컴퓨팅 노드 수
OvercloudComputeFlavor	컴퓨팅 노드에 사용할 플레이버(컴퓨팅**)**
CephStorageCount	확장할 Ceph 스토리지(OSD) 노드 수
OvercloudCephStorageFlavor	Ceph Storage(OSD) 노드에 사용할 플레이버(**ceph-storage**)
CephMonCount	확장할 전용 Ceph MON 노드 수
OvercloudCephMonFlavor	전용 Ceph MON 노드(**ceph-mon**)에 사용할 플레이버
CephMdsCount	확장할 전용 Ceph MDS 노드 수
OvercloudCephMdsFlavor	전용 Ceph MDS 노드(**ceph-mds**)에 사용할 플레이버

중요

CephMonCount,CephMdsCount,OvercloudCephMonFlavor 및 OvercloudCephMdsFlavor 매개변수(ceph-mon 및 ceph-mds 플레이버와 함께)는 3장. *전용 노드에 Ceph 서비스 배포* 에 설명된 대로 사용자 지정 CephMON 및 CephMds 역할을 생성한 경우에만 유효합니다.

예를 들어 각 역할(Controller, Compute, Ceph-Storage, CephMon)에 3개의 노드를 배포하도록 오버클라우드를 구성하려면 parameter_defaults 에 다음을 추가합니다.

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
  CephMdsCount: 3
  OvercloudCephMdsFlavor: ceph-mds
```



참고

Heat 템플릿 매개변수 의 전체 목록은 Director 설치 및 사용 가이드에서 CLI 툴을 사용하여 Overcloud 생성을 참조하십시오.

7.2. 오버클라우드 배포 시작

참고

언더클라우드를 설치하는 동안 undercloud.conf 파일에 generate_service_certificate=false 를 설정합니다. 그렇지 않으면 *Advanced Overcloud Customization* 가이드의 Overcloud Public Endpoint에서 SSL/TLS 활성화 에 설명된 대로 오버클라우드를 배포할 때 신뢰 앵커를 삽입해야 합니다.

참고

오버클라우드 배포 중에 Ceph 대시보드를 추가하려면 8장. *오버클라우드 배포에 Red Hat Ceph Storage 대시보드 추가* 을 참조하십시오.

오버클라우드를 생성하려면 openstack overcloud deploy 명령에 추가 인수가 필요합니다. 예를 들면 다음과 같습니다.

```
$ openstack overcloud deploy --templates -r /home/stack/templates/roles_data_custom.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml \
  -e /home/stack/templates/storage-config.yaml \
  -e /home/stack/templates/ceph-config.yaml \
  --ntp-server pool.ntp.org
```

위의 명령은 다음 옵션을 사용합니다.

- **--templates -** 기본 **Heat** 템플릿 컬렉션(즉, **/usr/share/openstack-tripleo-heat-templates/**)에서 **Overcloud**를 생성합니다.
- **-r /home/stack/templates/roles_data_custom.yaml -** 3장. *전용 노드에 Ceph 서비스 배포*에서 사용자 지정 역할 정의 파일을 지정합니다. 이 파일은 **Ceph MON** 또는 **Ceph MDS** 서비스에 대한 사용자 지정 역할을 추가합니다. 이러한 역할을 통해 두 서비스를 전용 노드에 설치할 수 있습니다.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml - director**를 설정하여 **Ceph** 클러스터를 생성합니다. 특히 이 환경 파일은 *컨테이너화된* **Ceph Storage** 노드가 있는 **Ceph** 클러스터를 배포합니다.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml -** 4.2절. “Ceph Object Gateway 활성화”에 설명된 대로 **Ceph Object Gateway**를 활성화합니다.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml -**에 설명된 대로 **Ceph** 메타데이터 서버를 활성화합니다. 4.1절. “Ceph 메타데이터 서버 활성화”.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml -** 4.4절. “Ceph를 사용하도록 백업 서비스 구성”에 설명된 대로 **Block Storage Backup** 서비스 **(cinder-backup)**를 활성화합니다.
- **-e /home/stack/templates/storage-config.yaml -** 사용자 지정 **Ceph Storage** 구성이 포함된 환경 파일을 추가합니다.

- **-e /home/stack/templates/ceph-config.yaml - *5장. Ceph Storage 클러스터 사용자 지정*** 에 설명된 대로 사용자 지정 **Ceph** 클러스터 설정이 포함된 환경 파일을 추가합니다.

- **--ntp-server pool.ntp.org - NTP** 서버를 설정합니다.

작은 정보

응답 파일을 사용하여 모든 템플릿과 환경 파일을 호출할 수도 있습니다. 예를 들어 다음 명령을 사용하여 동일한 오버클라우드를 배포할 수 있습니다.

```
$ openstack overcloud deploy -r /home/stack/templates/roles_data_custom.yaml \
  --answers-file /home/stack/templates/answers.yaml --ntp-server pool.ntp.org
```

이 경우 **/home/stack/templates/answers.yaml** 응답 파일에는 다음이 포함됩니다.

```
templates: /usr/share/openstack-tripleo-heat-templates/
environments:
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-rgw.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-mds.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml
  - /home/stack/templates/storage-config.yaml
  - /home/stack/templates/ceph-config.yaml
```

자세 한 내용은 오버클라우드 배포에서 환경 파일 포함을 참조하십시오.

전체 옵션 목록은 다음을 입력합니다.

```
$ openstack help overcloud deploy
```

자세 한 내용은 ***Director 설치 및 사용 가이드의 CLI 툴을 사용하여 기본 오버클라우드 구성 을*** 참조하십시오.

오버클라우드 생성 프로세스가 시작되고 **director**가 노드를 프로비저닝합니다. 이 프로세스를 완료하는 데 다소 시간이 걸립니다. 오버클라우드 생성 상태를 보려면 **stack** 사용자로 별도의 터미널을 열고 다음 명령을 입력합니다.

```
$ source ~/stackrc
$ openstack stack list --nested
```

7.2.1. ceph-anible 이 실행되는 노드 제한

ceph-anible 이 실행되는 노드를 제한하여 배포 업데이트 시간을 줄일 수 있습니다. **RHOSP(Red Hat OpenStack Platform)**에서 **config-download** 를 사용하여 **Ceph**를 구성하는 경우 전체 배포에서 **config-download** 및 **ceph- anible** 을 실행하는 대신 **--limit** 옵션을 사용하여 노드 목록을 지정할 수 있습니다. 이 기능은 예를 들어 오버클라우드를 확장하거나 실패한 디스크를 교체하는 데 유용합니다. 이러한 시나리오에서는 환경에 추가하는 새 노드에서만 배포를 실행할 수 있습니다.

실패한 디스크 교체에서 **--limit** 를 사용하는 시나리오의 예

다음 예제 절차에서 **Ceph storage** 노드 **oc0-cephstorage-0** 에는 디스크 오류가 있으므로 새 팩토리 클린 디스크를 받을 수 있습니다. 새 디스크를 **OSD**로 사용할 수 있도록 **oc0-cephstorage-0** 노드에서 **Ansible**을 실행해야 하지만 다른 모든 **Ceph** 스토리지 노드에서 실행할 필요는 없습니다. 환경 파일과 노드 이름을 환경에 적합한 이름으로 바꿉니다.

절차

1. **stack** 사용자로 언더클라우드 노드에 로그인하고 **stack rc** 자격 증명 파일을 가져옵니다.

```
# source stackrc
```

2. 새 디스크를 사용하여 누락된 **OSD**를 시작하도록 다음 단계 중 하나를 완료합니다.

- 스택 업데이트를 실행하고 **--limit** 옵션을 포함하여 **ceph-ansible** 을 실행할 노드를 지정합니다.

```
$ openstack overcloud deploy --templates \
  -r /home/stack/roles_data.yaml \
  -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e ~/my-ceph-settings.yaml \
  -e <other-environment_files> \
  --limit oc0-controller-0:oc0-controller-2:oc0-controller-1:oc0-cephstorage-0:undercloud
```

이 예에서는 **Ceph mons**에서 **OSD** 정의를 변경하는 데 **Ansible**이 필요하므로 컨트롤러가 포함됩니다.

- **config-download** 에서 **ansible-playbook-command.sh** 스크립트를 생성한 경우 **--**

limit 옵션으로 스크립트를 실행하여 지정된 노드를 **ceph-ansible** 에 전달할 수도 있습니다.

```
./ansible-playbook-command.sh --limit oc0-controller-0:oc0-controller-2:oc0-controller-1:oc0-cephstorage-0:undercloud
```

경고

항상 제한 목록에 언더클라우드를 포함해야 합니다. 그러지 않으면 **--limit** 를 사용하는 경우 **ceph-anible** 을 실행할 수 없습니다. 이는 **ceph-ansible** 실행이 언더클라우드에서만 실행되는 **external_deploy_steps_tasks** 플레이북을 통해 발생하기 때문에 필요합니다.

8장. 오버클라우드 배포에 RED HAT CEPH STORAGE 대시보드 추가

Red Hat Ceph Storage Dashboard는 기본적으로 비활성화되어 있지만 Red Hat OpenStack Platform director를 사용하여 오버클라우드에서 활성화할 수 있습니다. Ceph 대시보드는 클러스터의 다양한 측면과 오브젝트를 관리하는 기본 제공 웹 기반 Ceph 관리 및 모니터링 애플리케이션입니다. Red Hat Ceph Storage Dashboard는 다음과 같은 구성 요소로 구성됩니다.

- Ceph 대시보드 관리자 모듈은 사용자 인터페이스를 제공하고 플랫폼 프런트엔드인 Grafana를 포함합니다.
- 모니터링 플러그인인 Prometheus.
- Alertmanager는 경고를 대시보드로 보냅니다.
- 노드 내보내기는 클러스터 데이터를 대시보드로 내보냅니다.

참고

이 기능은 Ceph Storage 4.1 이상에서 지원됩니다. 시스템에 설치된 Ceph Storage 버전을 결정하는 방법에 대한 자세한 내용은 Red Hat Ceph Storage 릴리스 및 해당 Ceph 패키지 버전을 참조하십시오.

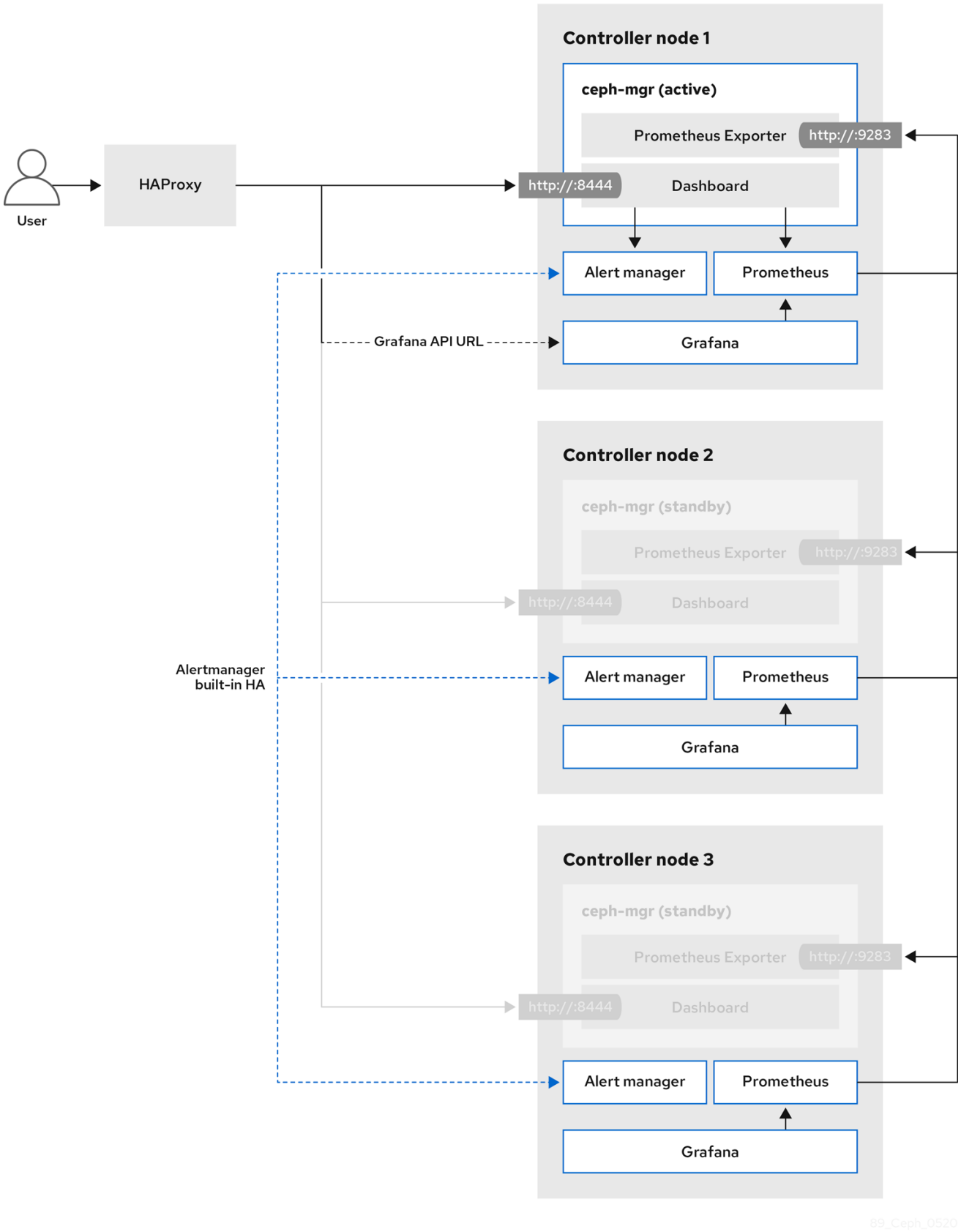
참고

Red Hat Ceph Storage 대시보드는 항상 다른 Ceph 관리자 구성 요소와 동일한 노드에 공동 배치됩니다.

참고

초기 오버클라우드 배포 중에 Ceph 대시보드를 추가하려면 7.2절. “오버클라우드 배포 시작” 에 초기 오버클라우드를 배포하기 전에 이 장의 절차를 완료하십시오.

다음 다이어그램에서는 Red Hat OpenStack Platform의 Ceph 대시보드 아키텍처를 보여줍니다.



대시보드 및 해당 기능 및 제한 사항에 대한 자세한 내용은 ***Red Hat Ceph Storage*** *대시보드 가이드의 대시보드 기능을 참조하십시오.*

Ceph 대시보드를 사용하여 어디서나 TLS

대시보드 프런트엔드는 모든 모든 프레임워크에서 **TLS**와 완벽하게 통합됩니다. 필요한 환경 파일이 있고 오버클라우드 배포 명령에 포함된 제공된 모든 위치에서 **TLS**를 활성화할 수 있습니다. 이렇게 하면 **Grafana** 및 **Ceph** 대시보드 모두에 대한 인증서 요청을 트리거하고 생성된 인증서 및 키 파일은 **Overcloud** 배포 중에 **ceph-anible** 로 전달됩니다. 대시보드 및 기타 **openstack** 서비스에 대한 **TLS** 활성화 방법에 대한 자세한 내용은 ***Advanced Overcloud Customization*** 가이드의 다음 위치를 참조하십시오.

- **Overcloud** 공용 엔드포인트에서 **SSL/TLS** 활성화.
- **ID** 관리를 사용하여 내부 및 공용 엔드포인트에서 **SSL/TLS** 활성화.

참고

Ceph 대시보드에 도달할 포트는 **TLS-everywhere** 컨텍스트에서도 동일하게 유지됩니다.

8.1. CEPH 대시보드에 필요한 컨테이너 포함

오버클라우드에 **Ceph** 대시보드 템플릿을 추가하려면 **containers-prepare-parameter.yaml** 파일을 사용하여 필요한 컨테이너를 포함해야 합니다. 컨테이너 이미지를 준비하기 위해 **containers-prepare-parameter.yaml** 파일을 생성하려면 다음 단계를 완료합니다.

절차

1. **stack** 사용자로 언더클라우드 호스트에 로그인합니다.
2. 기본 컨테이너 이미지 준비 파일을 생성합니다.

```
$ sudo openstack tripleo container image prepare default \
  --local-push-destination \
  --output-env-file containers-prepare-parameter.yaml
```

3. **containers-prepare-parameter.yaml** 파일을 편집하고 요구 사항에 맞게 수정합니다. 다음 예제 **containers-prepare-parameter.yaml** 파일에는 **Grafana, Prometheus, Alertmanager, Node Exporter**를 포함하여 대시보드 서비스와 관련된 이미지 위치 및 태그가 포함되어 있습니다. 특정 시나리오에 따라 값을 편집합니다.

```
parameter_defaults:
    ContainerImagePrepare:
    - push_destination: true
```

```
    set:
      ceph_alertmanager_image: ose-prometheus-alertmanager
      ceph_alertmanager_namespace: registry.redhat.io/openshift4
      ceph_alertmanager_tag: v4.1
      ceph_grafana_image: rhceph-4-dashboard-rhel8
      ceph_grafana_namespace: registry.redhat.io/rhceph
      ceph_grafana_tag: 4
      ceph_image: rhceph-4-rhel8
      ceph_namespace: registry.redhat.io/rhceph
      ceph_node_exporter_image: ose-prometheus-node-exporter
      ceph_node_exporter_namespace: registry.redhat.io/openshift4
      ceph_node_exporter_tag: v4.1
      ceph_prometheus_image: ose-prometheus
      ceph_prometheus_namespace: registry.redhat.io/openshift4
      ceph_prometheus_tag: v4.1
      ceph_tag: latest
```

containers-prepare-parameter.yaml 파일을 사용한 레지스트리 및 이미지 구성에 대한 자세한 내용은 컨테이너 이미지 준비 매개 변수 의 *컨테이너 이미지 준비 매개변수를 참조하십시오*.

8.2. CEPH 대시보드 배포

참고

구성 가능한 네트워크를 사용하여 **Ceph** 대시보드를 배포하려면 다음을 참조하십시오. 8.3절. “구성 가능한 네트워크를 사용하여 Ceph 대시보드 배포”

참고

Ceph 대시보드 관리자 사용자 역할은 기본적으로 읽기 전용 모드로 설정됩니다. **Ceph** 대시보드 관리자 기본 모드를 변경하려면 8.4절. “기본 권한 변경” 을 참조하십시오.

절차

1. **stack** 사용자로 **Undercloud** 노드에 로그인합니다.

2. **openstack overcloud deploy** 명령에 배포에 포함된 모든 환경 파일을 사용하여 다음 환경 파일을 포함합니다.

```
$ openstack overcloud deploy \
 --templates \
 -e <overcloud_environment_files> \
 -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-dashboard.yaml
```

<overcloud_environment_files> 를 배포에 포함된 환경 파일 목록으로 바꿉니다.

결과

결과 배포는 **grafana, prometheus, alertmanager** 및 **node-exporter** 컨테이너가 있는 외부 스택으로 구성됩니다. **Ceph Dashboard Manager** 모듈은 이 스택의 백엔드이며, 최종 사용자에게 **ceph** 클러스터별 지표를 제공하기 위해 **grafana** 레이아웃을 포함합니다.

8.3. 구성 가능한 네트워크를 사용하여 CEPH 대시보드 배포

기본 프로비저닝 네트워크가 아니라 구성 가능 네트워크에 **Ceph** 대시보드를 배포할 수 있습니다. 이렇게 하면 프로비저닝 네트워크에 **Ceph** 대시보드 서비스를 노출할 필요가 없습니다. 구성 가능 네트워크에 대시보드를 배포할 때 별도의 권한 부여 프로필을 구현할 수도 있습니다.

Overcloud를 처음 배포할 때만 대시보드를 새 네트워크에 적용할 수 있으므로 배포 전에 사용할 네트워크를 선택해야 합니다. 대시보드를 기존 외부 네트워크에 적용하거나 프로비저닝 네트워크 이외의 기존 네트워크 중 하나를 재사용할 수 없습니다. 다음 절차에 따라 배포 전에 구성 가능 네트워크를 선택합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.

2. 대시보드 구성 가능 네트워크를 포함하도록 컨트롤러 특정 역할을 생성합니다.

```
$ openstack overcloud roles generate -o /home/stack/roles_data_dashboard.yaml
ControllerStorageDashboard Compute BlockStorage ObjectStorage CephStorage
```

결과

- 명령의 출력으로 정의된 **roles_data.yaml** 내부에 새 **ControllerStorageDashboard** 역할이 생성됩니다. **overcloud deploy** 명령을 사용할 때 이 파일을 템플릿 목록에 포함해야 합니다.

  알림: **ControllerStorageDashboard** 역할에 **CephNFS** 또는 **network_data_dashboard.yaml** 이 포함되지 않습니다.

- **director**는 구성 가능 네트워크가 정의된 네트워크 환경 파일을 제공합니다. 이 파일의 기본 위치는 **/usr/share/openstack-tripleo-heat-**

templates/network_data_dashboard.yaml 입니다. 오버클라우드 배포 명령을 사용할 때 오버클라우드 템플릿 목록에 이 파일을 포함해야 합니다.

3. **openstack overcloud deploy** 명령에 배포에 포함된 모든 환경 파일을 사용하여 다음 환경 파일을 포함합니다.

```
$ openstack overcloud deploy \
  --templates \
  -r /home/stack/roles_data.yaml \
  -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
  -e <overcloud_environment_files> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
dashboard.yaml
```

<overcloud_environment_files> 를 배포에 포함된 환경 파일 목록으로 바꿉니다.

결과

결과 배포는 **grafana, prometheus, alertmanager** 및 **node-exporter** 컨테이너가 있는 외부 스택으로 구성됩니다. **Ceph Dashboard Manager** 모듈은 이 스택의 백엔드이며, 최종 사용자에게 **Ceph** 클러스터별 지표를 제공하기 위해 **grafana** 레이아웃을 포함합니다.

8.4. 기본 권한 변경

Ceph Dashboard admin 사용자 역할은 **Ceph** 클러스터의 안전한 모니터링을 위해 기본적으로 읽기 전용 모드로 설정됩니다. 관리자가 대시보드를 사용하여 **Ceph** 클러스터의 요소를 변경할 수 있도록 관리자 권한이 상승된 권한을 갖도록 허용하려면 **CephDashboardAdminRO** 매개 변수를 사용하여 기본 관리 권한을 변경할 수 있습니다.

경고

전체 권한이 있는 사용자는 **director**에서 구성하는 클러스터 요소를 변경할 수 있습니다. 이로 인해 스택 업데이트를 실행할 때 **director**가 구성된 옵션과 충돌할 수 있습니다. 이 문제를 방지하려면 **Ceph** 대시보드(예: **Ceph OSP** 풀) 속성을 사용하여 **director** 구성 옵션을 변경하지 마십시오.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.

2. 다음 **ceph_dashboard_admin.yaml** 환경 파일을 생성합니다.

```
parameter_defaults:
  CephDashboardAdminRO: false
```

3. 오버클라우드 배포 명령을 실행하여 기존 스택을 업데이트하고 기존 배포의 일부인 다른 모든 환경 파일과 함께 환경 파일을 포함합니다.

```
$ openstack overcloud deploy \
--templates \
-e <existing_overcloud_environment_files> \
-e ceph_dashboard_admin.yml
```

<existing_overcloud_environment_files> 를 기존 배포의 일부인 환경 파일 목록으로 바꿉니다.

8.5. CEPH 대시보드에 액세스

Ceph 대시보드가 올바르게 실행되고 있는지 테스트하려면 다음 확인 단계를 완료하고 Ceph 클러스터에서 표시되는 데이터가 올바른지 확인합니다.

절차

1. **stack** 사용자로 **Undercloud** 노드에 로그인합니다.

2. 대시보드 관리자 로그인 자격 증명을 검색합니다.

```
[stack@undercloud ~]$ grep dashboard_admin_password /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml
```

3. **VIP** 주소를 검색하여 **Ceph** 대시보드에 액세스합니다.

```
[stack@undercloud-0 ~]$ grep dashboard_frontend_vip /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml
```

4. 웹 브라우저를 사용하여 프론트엔드 **VIP**를 가리키고 대시보드에 액세스합니다. **director**는 프로비저닝 네트워크에서 대시보드를 구성하고 노출하므로 검색한 **VIP**를 사용하여 **TCP** 포트 **8444**에서 대시보드에 직접 액세스할 수 있습니다. 다음 조건이 충족되었는지 확인합니다.

- 웹 클라이언트 호스트는 프로비저닝 네트워크에 연결된 계층 2입니다.

- 프로비저닝 네트워크는 올바르게 라우팅되거나 프록시되며, 웹 클라이언트 호스트에서 연결할 수 있습니다. 이러한 조건이 충족되지 않은 경우 SSH 터널을 열어 오버클라우드의 대시보드 VIP에 도달할 수 있습니다.

```
client_host$ ssh -L 8444:<dashboard_vip>:8444 stack@<your undercloud>
```

  <dashboard_vip>를 검색한 컨트롤 플레인 VIP의 IP 주소로 바꿉니다.

5. 대시보드에 액세스하려면 웹 브라우저에서 http://localhost:8444 로 이동하여 다음 세부 정보를 사용하여 로그인합니다.

- ceph-ansible 이 생성하는 기본 사용자 admin.

- /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml 의 암호.

결과

- Ceph 대시보드에 액세스할 수 있습니다.

- 대시보드에서 표시하는 숫자와 그래프는 CLI 명령인 ceph -s 가 반환하는 것과 동일한 클러스터 상태를 반영합니다.

Red Hat Ceph Storage 대시보드에 대한 자세한 내용은 *Red Hat Ceph Storage 관리 가이드를 참조하십시오*.

9장. 배포 후

다음 하위 섹션에서는 Ceph 클러스터 관리를 위한 여러 배포 후 작업을 설명합니다.

9.1. 오버클라우드 액세스

director는 언더클라우드에서 오버클라우드와의 상호 작용을 구성하고 인증을 지원하는 스크립트를 생성합니다. director는 이 파일(overcloudrc)을 stack 사용자의 홈 디렉터리에 저장합니다. 이 파일을 사용하려면 다음 명령을 실행합니다.

```
$ source ~/overcloudrc
```

이렇게 하면 언더클라우드 CLI에서 오버클라우드와 상호 작용하는 데 필요한 환경 변수가 로드됩니다. 언더클라우드와 상호 작용으로 돌아가려면 다음 명령을 실행합니다.

```
$ source ~/stackrc
```

9.2. CEPH STORAGE 노드 모니터링

오버클라우드를 생성한 후 Ceph Storage 클러스터의 상태를 확인하여 올바르게 작동하는지 확인합니다.

절차

1. 컨트롤러 노드에 heat-admin 사용자로 로그인합니다.

```
$ nova list
$ ssh heat-admin@192.168.0.25
```

2. 클러스터 상태를 확인합니다.

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph health
```

클러스터에 문제가 없는 경우 명령에서 HEALTH_OK 를 다시 보고합니다. 즉, 클러스터가 안전하게 사용할 수 있습니다.

3. Ceph 모니터 서비스를 실행하는 오버클라우드 노드에 로그인하고 클러스터의 모든 OSD 상

태를 확인합니다.

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd tree
```

4. **Ceph Monitor** 쿼럼의 상태를 확인합니다.

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph quorum_status
```

쿼럼에 참여하는 모니터와 리더를 보여줍니다.

5. 모든 **Ceph OSD**가 실행 중인지 확인합니다.

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd stat
```

Ceph Storage 클러스터 모니터링에 대한 자세한 내용은 ***Red Hat Ceph Storage** 관리 가이드의 **Monitoring** 을 참조하십시오.*

10장. 환경 재부팅

환경을 재부팅해야 하는 경우 발생할 수 있습니다. 예를 들어 물리적 서버를 수정해야 하거나 정전으로부터 복구해야 할 수 있습니다. 이 경우 **Ceph Storage** 노드가 올바르게 부팅되어야 합니다.

다음 순서로 노드를 부팅해야 합니다.

- 먼저 모든 **Ceph Monitor** 노드를 부팅합니다. 그러면 고가용성 클러스터에서 **Ceph Monitor** 서비스가 활성화됩니다. 기본적으로 **Ceph Monitor** 서비스는 컨트롤러 노드에 설치됩니다. **Ceph Monitor**가 사용자 지정 역할의 컨트롤러와 별개인 경우 이 사용자 지정 **Ceph Monitor** 역할이 활성 상태인지 확인합니다.

- 모든 **Ceph Storage** 노드 부팅 - 이렇게 하면 **Ceph OSD** 클러스터가 컨트롤러 노드의 활성 **Ceph Monitor** 클러스터에 연결할 수 있습니다.

10.1. CEPH STORAGE(OSD) 클러스터 재부팅

Ceph Storage(OSD) 노드 클러스터를 재부팅하려면 다음 단계를 완료합니다.

사전 요구 사항

- **ceph-mon** 서비스를 실행하는 **Ceph Monitor** 또는 컨트롤러 노드에서 **Red Hat Ceph Storage** 클러스터 상태가 정상이고 **pg** 상태가 **active+clean** 인지 확인합니다.

```
$ sudo podman exec -it ceph-mon-controller-0 ceph -s
```

Ceph 클러스터가 정상이면 **HEALTH_OK** 상태를 반환합니다.

Ceph 클러스터 상태가 비정상인 경우 **HEALTH_WARN** 또는 **HEALTH_ERR** 의 상태를 반환합니다. 문제 해결 지침은 **Red Hat Ceph Storage 4** 문제 해결 가이드를 참조하십시오.

절차

1. **ceph-mon** 서비스를 실행하는 **Ceph Monitor** 또는 컨트롤러 노드에 로그인하고 **Ceph Storage** 클러스터 재조정을 일시적으로 비활성화합니다.

```
$ sudo podman exec -it ceph-mon-controller-0 ceph osd set noout
$ sudo podman exec -it ceph-mon-controller-0 ceph osd set norebalance
```

참고

다중 스택 또는 **DCN(Distributed Compute node)** 아키텍처가 있는 경우 **noout** 및 **norebalance** 플래그를 설정할 때 클러스터 이름을 지정해야 합니다. 예: **sudo podman exec -it ceph-mon-controller-0 ceph osd set noout --cluster_name>**

2. 재부팅할 첫 번째 **Ceph Storage** 노드를 선택하고 노드에 로그인합니다.

3. 노드를 재부팅합니다.

```
$ sudo reboot
```

4. 노드가 부팅될 때까지 기다립니다.

5. 노드에 로그인하고 클러스터 상태를 확인합니다.

```
$ sudo podman exec -it ceph-mon-controller-0 ceph status
```

pgmap이 모든 **pgs**를 정상**(active+clean)**으로 보고하는지 확인합니다.

6. 노드에서 로그아웃하고, 다음 노드를 재부팅한 후 상태를 확인합니다. 모든 **Ceph Storage** 노드가 재부팅될 때까지 이 프로세스를 반복합니다.

7. 완료되면 **ceph-mon** 서비스를 실행하는 **Ceph Monitor** 또는 컨트롤러 노드에 로그인하고 클러스터 재조정을 다시 활성화합니다.

```
$ sudo podman exec -it ceph-mon-controller-0 ceph osd unset noout
$ sudo podman exec -it ceph-mon-controller-0 ceph osd unset norebalance
```

참고

다중 스택 또는 DCN(Distributed Compute node) 아키텍처가 있는 경우 noout 및 norebalance 플래그를 설정할 때 클러스터 이름을 지정해야 합니다. 예: sudo podman exec -it ceph-mon-controller-0 ceph osd set noout --cluster_name>

8. **최종 상태 검사를 수행하여 클러스터가 HEALTH_OK를 보고하는지 확인합니다.**

```
$ sudo podman exec -it ceph-mon-controller-0 ceph status
```

모든 Overcloud 노드가 동시에 부팅되는 경우 Ceph OSD 서비스가 Ceph Storage 노드에서 올바르게 시작되지 않을 수 있습니다. 이 경우 Ceph Storage OSD를 재부팅하여 Ceph Monitor 서비스에 연결할 수 있습니다.

다음 명령을 사용하여 Ceph Storage 노드 클러스터의 HEALTH_OK 상태를 확인합니다.

```
$ sudo ceph status
```

11장. CEPH STORAGE 클러스터 스케일링

11.1. CEPH STORAGE 클러스터 확장

필요한 Ceph Storage 노드 수로 배포를 다시 실행하여 오버클라우드의 Ceph Storage 노드 수를 확장할 수 있습니다.

이 작업을 수행하기 전에 업데이트된 배포에 필요한 노드가 충분히 있는지 확인합니다. 이러한 노드는 director에 등록하고 그에 따라 태그를 지정해야 합니다.

새 Ceph Storage 노드 등록

director에 새 Ceph 스토리지 노드를 등록하려면 다음 단계를 완료합니다.

절차

1. stack 사용자로 언더클라우드에 로그인하고 director 구성을 초기화합니다.

```
$ source ~/stackrc
```

2. 새 노드 정의 템플릿에서 새 노드의 하드웨어 및 전원 관리 세부 정보(예: instackenv-scale.json)를 정의합니다.

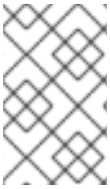
3. 이 파일을 director로 가져옵니다.

```
$ openstack overcloud node import ~/instackenv-scale.json
```

노드 정의 템플릿을 가져오면 해당 노드에 정의된 각 노드가 director에 등록됩니다.

4. 커널 및 ramdisk 이미지를 모든 노드에 할당합니다.

```
$ openstack overcloud node configure
```



참고

새 노드 등록에 대한 자세한 내용은 2.2절. “노드 등록” 을 참조하십시오.

새 노드 수동 태그

각 노드를 등록한 후에는 하드웨어를 검사하고 특정 프로필에 노드를 태그해야 합니다. 프로필 태그를 사용하여 노드를 플레이버에 일치시킨 다음 배포 역할에 플레이버를 할당합니다.

절차

1. 하드웨어 인트로스펙션을 트리거하여 각 노드의 하드웨어 속성을 검색합니다.

```
$ openstack overcloud node introspect --all-manageable --provide
```

- **all -manageable** 옵션은 관리 상태에 있는 노드만 인트로스펙션합니다. 이 예에서는 모든 노드가 관리 상태에 있습니다.
- **--provide** 옵션은 인트로스펙션 후 모든 노드를 **active** 상태로 재설정합니다.

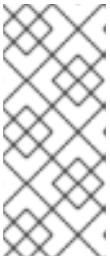
중요

이 프로세스가 성공적으로 완료되었는지 확인합니다. 베어 메탈 노드의 경우 이 프로세스는 일반적으로 **15**분 정도 걸립니다.

2. 노드 목록을 검색하여 **UUID**를 확인합니다.

```
$ openstack baremetal node list
```

3. **profile** 옵션을 각 노드의 **properties/capabilities** 매개변수에 추가하여 특정 프로필에 노드를 수동으로 태그합니다. **profile** 옵션을 추가하면 각 프로필에 노드를 태그합니다.



참고

수동 태그 대신 **AHC(Automated Health Check)** 툴을 사용하여 벤치마킹 데이터를 기반으로 다수의 노드에 자동으로 태그를 지정합니다. 예를 들어 다음 명령은 **ceph-storage** 프로필로 세 개의 추가 노드를 태그합니다.

```
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
551d81f5-4df2-4e0f-93da-6c5de0b868f7
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
5e735154-bd6b-42dd-9cc2-b6195c4196d7
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
1a2b090c-299d-4c20-a25d-57dd21a7085b
```

작은 정보

태그를 지정하고 등록된 노드에서 여러 디스크를 사용하는 경우 각 노드에서 특정 루트 디스크를 사용하도록 **director**를 설정할 수 있습니다. 자세한 내용은 **2.5절. "멀티 디스크 클러스터의 root 디스크 정의"**의 내용을 참조하십시오.

추가 **Ceph Storage** 노드를 사용하여 오버클라우드 재배포

새 노드를 등록하고 태그한 후 오버클라우드를 재배포하여 **Ceph Storage** 노드 수를 확장할 수 있습니다.

절차

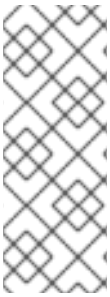
1. 오버클라우드를 재배포하기 전에 환경 파일의 **parameter_defaults** (이 경우 **~/templates/storage-config.yaml**)에 **CephStorageCount** 매개변수를 설정합니다. **7.1절. "역할에 노드 및 플레이버 할당"** 에서 오버클라우드는 세 개의 **Ceph Storage** 노드로 배포되도록 구성되어 있습니다. 다음 예제에서는 오버클라우드를 **6**개의 노드로 확장합니다.

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 6
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
```

2. **Overcloud**를 재배포합니다. 이제 오버클라우드에 **3**개 대신 **6**개의 **Ceph Storage** 노드가 있습니다.

11.2. CEPH STORAGE 노드 축소 및 교체

Ceph Storage 노드에 결함이 있는 경우와 같이 Ceph 클러스터를 축소하거나 Ceph Storage 노드를 교체해야 하는 경우도 있습니다. 두 경우 모두 데이터 손실을 방지하기 위해 오버클라우드에서 삭제하려는 Ceph Storage 노드를 비활성화하고 리밸런싱해야 합니다.



참고

이 절차에서는 ***Red Hat Ceph Storage** 관리 가이드* 의 단계를 사용하여 Ceph Storage 노드를 수동으로 제거합니다. Ceph Storage 노드를 수동으로 제거하는 방법에 대한 자세한 내용은 컨테이너에서 실행되는 Ceph 데몬 시작, 중지 및 다시 시작하고 명령줄 인터페이스를 사용하여 Ceph OSD 제거를 참조하십시오.

절차

1. 컨트롤러 노드에 heat-admin 사용자로 로그인합니다. director stack 사용자에게는 heat-admin 사용자에 액세스할 수 있는 SSH 키가 있습니다.

2. OSD 트리를 나열하고 노드의 OSD를 찾습니다. 예를 들어 삭제하려는 노드에는 다음 OSD가 포함될 수 있습니다.

```
-2 0.09998     host overcloud-cephstorage-0
0 0.04999        osd.0                 up  1.00000          1.00000
1 0.04999        osd.1                 up  1.00000          1.00000
```

3. Ceph Storage 노드에서 OSD를 비활성화합니다. 이 경우 OSD ID는 0과 1입니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd out 0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd out 1
```

4. Ceph Storage 클러스터는 리밸런싱을 시작합니다. 이 프로세스가 완료될 때까지 기다립니다. 다음 명령을 사용하여 상태를 따릅니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
-w
```

5. Ceph 클러스터 재조정이 완료되면 제거할 Ceph Storage 노드에 heat-admin 사용자로 overcloud-cephstorage-0 에 로그인하고 노드를 중지하고 비활성화합니다.

```
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@1
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl disable ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl disable ceph-osd@1
```

6. **OSD**를 중지합니다.

```
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@1
```

7. 컨트롤러 노드에 로그인하는 동안 더 이상 데이터를 받지 않도록 **CRUSH** 맵에서 **OSD**를 제거합니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd crush remove osd.0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd crush remove osd.1
```

8. **OSD** 인증 키를 제거합니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
auth del osd.0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
auth del osd.1
```

9. 클러스터에서 **OSD**를 제거합니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd rm 0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd rm 1
```

10. **CRUSH** 맵에서 **Storage** 노드를 제거합니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo docker exec ceph-mon-<HOSTNAME> ceph
osd crush rm <NODE>
[heat-admin@overcloud-controller-0 ~]$ sudo ceph osd crush remove <NODE>
```

CRUSH 트리를 검색하여 **CRUSH** 맵에 정의된 **<NODE>** 이름을 확인할 수 있습니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
```

```
osd crush tree | grep overcloud-osd-compute-3 -A 4
            "name": "overcloud-osd-compute-3",
            "type": "host",
            "type_id": 1,
            "items": []
        },
[heat-admin@overcloud-controller-0 ~]$
```

CRUSH 트리에서 **items** 목록이 비어 있는지 확인합니다. 목록이 비어 있지 않으면 **7**단계를 다시 방문하십시오.

11. 노드를 종료하고 **stack** 사용자로 언더클라우드로 돌아갑니다.

```
[heat-admin@overcloud-controller-0 ~]$ exit
[stack@director ~]$
```

12. **Ceph Storage** 노드를 비활성화하여 **director**가 다시 프로비저닝하지 않도록 합니다.

```
[stack@director ~]$ openstack baremetal node list
[stack@director ~]$ openstack baremetal node maintenance set UUID
```

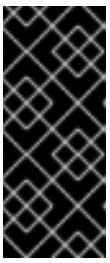
13. **Ceph Storage** 노드를 제거하려면 로컬 템플릿 파일을 사용하여 **director**의 오버클라우드 스택을 업데이트해야 합니다. 먼저 오버클라우드 스택의 **UUID**를 확인합니다.

```
$ openstack stack list
```

14. 삭제할 **Ceph Storage** 노드의 **UUID**를 확인합니다.

```
$ openstack server list
```

15. 스택에서 노드를 삭제하고 그에 따라 계획을 업데이트합니다.



중요

오버클라우드를 생성할 때 추가 환경 파일을 전달한 경우 **-e** 옵션을 사용하여 오버클라우드를 원치 않게 변경하지 않도록 다시 여기에 전달합니다. 자세한 내용은 ***Director** 설치 및 사용* 가이드 의 **Overcloud** 환경 수정을 참조하십시오.

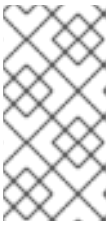
```
$ openstack overcloud node delete /
```

```
--stack <stack-name> /
--templates /
-e <other-environment-files> /
<node_UUID>
```

16. 스택이 업데이트가 완료될 때까지 기다립니다. **heat stack-list --show-nested** 명령을 사용하여 스택 업데이트를 모니터링합니다.

17. **director** 노드 풀에 새 노드를 추가하고 **Ceph Storage** 노드로 배포합니다. 환경 파일의 **parameter_defaults** 에서 **CephStorageCount** 매개 변수를 사용하여 **~/templates/storage-config.yaml** 을 사용하여 오버클라우드에서 총 **Ceph Storage** 노드 수를 정의합니다.

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
```



참고

역할당 노드 수를 정의하는 방법에 대한 자세한 내용은 **7.1절. “역할에 노드 및 플레이버 할당”** 을 참조하십시오.

18. 환경 파일을 업데이트한 후 오버클라우드를 재배포합니다.

```
$ openstack overcloud deploy --templates -e <ENVIRONMENT_FILE>
```

director가 새 노드를 프로비저닝하고 새 노드의 세부 정보로 전체 스택을 업데이트합니다.

19. 컨트롤러 노드에 **heat-admin** 사용자로 로그인하고 **Ceph Storage** 노드의 상태를 확인합니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo ceph status
```

20. **osdmap** 섹션의 값이 원하는 클러스터의 노드 수와 일치하는지 확인합니다. 제거한 **Ceph Storage** 노드는 새 노드로 교체됩니다.

11.3. CEPH STORAGE 노드에 OSD 추가

다음 절차에서는 노드에 **OSD**를 추가하는 방법을 설명합니다. **Ceph OSD**에 대한 자세한 내용은 ***Red Hat Ceph Storage Operations Guide의 Ceph OSD** 를 참조하십시오.*

절차

1. 다음 **heat** 템플릿은 **3**개의 **OSD** 장치가 있는 **Ceph Storage**를 배포합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

2. **OSD**를 추가하려면 **5.3**절. **“Ceph Storage** 노드 디스크 레이아웃 매핑**”** 에 설명된 대로 노드 디스크 레이아웃을 업데이트합니다. 이 예제에서는 템플릿에 **/dev/sde** 를 추가합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/sde
    osd_scenario: lvm
    osd_objectstore: bluestore
```

3. **openstack overcloud deploy**를 실행하여 **Overcloud**를 업데이트합니다.

참고

이 예에서 **OSD**가 있는 모든 호스트에는 **/dev/sde** 라는 새 장치가 있습니다. 모든 노드에 새 장치를 사용하지 않으려면 **heat** 템플릿을 업데이트합니다. 다양한 장치 목록을 사용하여 호스트를 정의하는 방법에 대한 자세한 내용은 **5.5**절. **“유사한 Ceph Storage** 노드의 매개변수 덮어쓰기**”** 및 **5.5.1.2**절. **“Ceph Storage** 노드의 디스크 레이아웃 변경**”** 을 참조하십시오.

11.4. CEPH STORAGE 노드에서 OSD 제거

다음 절차에서는 노드에서 **OSD**를 제거하는 방법을 설명합니다. 환경에 대해 다음을 가정합니다.

- 서버(**ceph-storage0**)에는 **/dev/sde** 에서실행되는 **OSD(ceph-osd@4)**가 있습니다.
- **Ceph** 모니터 서비스(**ceph-mon**)는 **controller0** 에서 실행되고 있습니다.
- 스토리지 클러스터의 비율이 거의 없는지 확인하기에 충분한 **OSD**가 있습니다.

Ceph OSD에 대한 자세한 내용은 ***Red Hat Ceph Storage Operations Guide의 Ceph OSD 를 참조하십시오.***

절차

1. **ceph-storage0** 에 **SSH**로 연결하고 **root** 로 로그인합니다.
2. **OSD** 서비스를 비활성화하고 중지합니다.

```
[root@ceph-storage0 ~]# systemctl disable ceph-osd@4
[root@ceph-stoarge0 ~]# systemctl stop ceph-osd@4
```

3. **ceph-storage0** 에서 연결을 끊습니다.
4. **controller0** 에 **SSH**로 연결하고 **root** 로 로그인합니다.
5. **Ceph** 모니터 컨테이너의 이름을 확인합니다.

```
[root@controller0 ~]# podman ps | grep ceph-mon
ceph-mon-controller0
[root@controller0 ~]#
```

6. **Ceph** 모니터 컨테이너를 활성화하여 바람직하지 않은 **OSD**를 외부로 표시합니다.

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd out 4
```

참고

이 명령을 사용하면 Ceph에서 스토리지 클러스터를 리밸런싱하고 데이터를 클러스터의 다른 OSD에 복사합니다. 클러스터는 리밸런싱이 완료될 때까지 active+clean 상태를 일시적으로 유지합니다.

7. **다음 명령을 실행하고 스토리지 클러스터 상태가 active+clean 이 될 때까지 기다립니다.**

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph -w
```

8. **더 이상 데이터를 받지 않도록 CRUSH 맵에서 OSD를 제거합니다.**

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd crush remove osd.4
```

9. **OSD 인증 키를 제거합니다.**

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph auth del osd.4
```

10. **OSD를 제거합니다.**

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd rm 4
```

11. **controller0 에서 연결을 끊습니다.**

12. **stack 사용자로 언더클라우드에 SSH로 연결하고 CephAnsibleDisksConfig 매개변수를 정의한 heat 환경 파일을 찾습니다.**

13. **heat 템플릿에 OSD 4개가 포함되어 있습니다.**

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
```

```
    - /dev/sde
  osd_scenario: lvm
  osd_objectstore: bluestore
```

14. **/dev/sde** 를 제거하도록 템플릿을 수정합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

15. **openstack overcloud deploy**를 실행하여 **Overcloud**를 업데이트합니다.

참고

이 예제에서는 **OSD**가 있는 모든 호스트에서 **/dev/sde** 장치를 제거합니다. 모든 노드에서 동일한 장치를 제거하지 않으면 **heat** 템플릿을 업데이트합니다. 다양한 장치 목록을 사용하여 호스트를 정의하는 방법에 대한 자세한 내용은 **5.5절. "유사한 Ceph Storage 노드의 매개변수 덮어쓰기"** 을 참조하십시오.

12장. 실패한 디스크 교체

Ceph 클러스터에서 디스크 중 하나가 실패하면 다음 절차를 완료하여 교체하십시오.

1. 장치 이름이 변경될지 여부를 확인하고 12.1절. "장치 이름 변경 여부 확인" 을 참조하십시오.

2. OSD가 다운되고 삭제되었는지 확인하려면 12.2절. "OSD가 다운되고 삭제되었는지 확인" 을 참조하십시오.

3. 시스템에서 이전 디스크를 제거하고 교체 디스크를 설치한 경우 12.3절. "시스템에서 이전 디스크 제거 및 교체 디스크 설치" 을 참조하십시오.

4. 디스크 교체에 성공했는지 확인하려면 12.4절. "디스크 교체에 성공했는지 확인" 을 참조하십시오.

12.1. 장치 이름 변경 여부 확인

디스크를 교체하기 전에 대체 OSD의 대체 디스크의 이름이 운영 체제에서 교체할 장치와 다른 이름이 있는지 확인합니다. 교체 디스크에 다른 이름이 있는 경우 director가 ceph-ansible을 실행할 때를 포함하여 후속 ceph-anible 실행을 포함하도록 장치 목록에 대한 Ansible 매개 변수를 업데이트해야 변경으로 인해 실패하지 않습니다. director를 사용할 때 변경해야 하는 장치 목록의 예는 5.3절. "Ceph Storage 노드 디스크 레이아웃 매핑" 을 참조하십시오.

주의

장치 이름이 변경되고 다음 절차를 사용하여 ceph-ansible 또는 director 외부에서 시스템을 업데이트하는 경우, 구성 관리 도구가 시스템 정의 파일을 업데이트할 때까지 관리하는 시스템과 동기화되지 않을 위험이 있으며 구성은 오류 없이 재평가될 수 있습니다.

스토리지 장치의 영구 이름 지정

sd 드라이버에서 관리하는 스토리지 장치에 재부팅 시 이름이 항상 같은 것은 아닙니다. 예를 들어, 일반적으로 /dev/sdc 로 식별되는 디스크의 이름은 /dev/sdb 입니다. /dev/sdc 를 대체 디스크로 사용하려는 경우에도 /dev/ sdc 를 운영 체제에 /dev/sdd 로 표시할 수도 있습니다. 이 문제를 해결하려면 영구적

인 이름을 사용하고 다음 패턴과 일치하는 /dev/disk/by-* 를 사용합니다. 자세한 내용은 RHEL(Red Hat Enterprise Linux) 7 *스토리지 관리 가이드의 영구 명명 가이드를 참조하십시오*.

Ceph 배포에 사용하는 명명 방법에 따라 OSD를 교체한 후 장치 목록을 업데이트해야 할 수 있습니다. 다음 네이밍 방법 목록을 사용하여 장치 목록을 변경해야 하는지 확인합니다.

메이저 및 마이너 번호 범위 방법

sd 를 사용하고 계속 사용하려면 새 디스크를 설치한 후 이름이 변경되었는지 확인합니다. 이름이 변경되지 않은 경우 예를 들어 동일한 이름이 /dev/sdd 와 올바르게 표시되는 경우 디스크 교체 절차를 완료한 후 이름을 변경할 필요가 없습니다.

중요

이 명명 방법은 시간이 지남에 따라 이름이 일관되지 않게 되는 위험이 있으므로 권장되지 않습니다. 자세한 내용은 RHEL 7 *스토리지 관리 가이드의 영구 명명 가이드를 참조하십시오*.

by-path 방법

이 방법을 사용하고 동일한 슬롯에 대체 디스크를 추가하는 경우 경로가 일관되고 변경이 필요하지 않습니다.

중요

이 명명 방법은 주 번호 범위 방법보다 우선하지만, 타겟 번호가 변경되지 않도록 주의하십시오. 예를 들어 호스트 어댑터가 다른 PCI 슬롯으로 이동하는 경우 영구 바인딩을 사용하고 이름을 업데이트합니다. 또한 HBA가 프로브에 실패하거나 드라이버가 다른 순서로 로드되거나 시스템에 새 HBA가 설치된 경우 SCSI 호스트 번호가 변경될 수 있습니다. 경로 이름 지정 방법은 RHEL7과 RHEL8 사이에서도 다릅니다. 자세한 내용은 다음을 참조하십시오.

- 문서 [RHEL8 및 RHEL7에서 생성된 "경로" 링크의 차이점은 무엇입니까?] https://access.redhat.com/solutions/5171991
- RHEL 8 *파일 시스템 관리* 가이드 의 영구 명명 속성 개요.

by-uuid 방법

이 방법을 사용하는 경우 blkid 유틸리티를 사용하여 새 디스크를 이전 디스크와 동일한 UUID로

설정할 수 있습니다. 자세한 내용은 **RHEL 7** *스토리지 관리 가이드의 영구 명명 가이드를 참조하십시오.*

by-id 방법

이 식별자는 장치의 속성이고 장치가 교체되었으므로 이 메서드를 사용하는 경우 장치 목록을 변경해야 합니다.

새 디스크를 시스템에 추가하는 경우 ***RHEL7*** *스토리지 관리자 가이드에* 따라 영구 명명 속성을 수정할 수 있는 경우 장치 이름이 변경되지 않도록 영구 명명 을 참조하십시오. 그러면 장치 목록을 업데이트하고 **ceph-ansible**을 다시 실행하거나 **director**가 **ceph - anible** 을 다시 실행하고 디스크 교체 절차를 진행할 수 있습니다. 그러나 **ceph-ansible** 을 다시 실행하여 변경에 불일치가 발생하지 않도록 할 수 있습니다.

12.2. OSD가 다운되고 삭제되었는지 확인

Ceph Monitor를 호스팅하는 서버에서 실행 중인 모니터 컨테이너에서 **ceph** 명령을 사용하여 교체할 **OSD**가 다운되었는지 확인한 다음 삭제합니다.

절차

1. 실행 중인 **Ceph** 모니터 컨테이너의 이름을 확인하고 **MON** 이라는 환경 변수에 저장합니다.

```
MON=$(podman ps | grep ceph-mon | awk {'print $1'})
```

2. 실행 중인 **Ceph** 모니터 컨테이너 내에서 실행되도록 **ceph** 명령을 별칭합니다.

```
alias ceph="podman exec $MON ceph"
```

3. 새 별칭을 사용하여 교체할 **OSD**가 종료되었는지 확인합니다.

```
[root@overcloud-controller-0 ~]# ceph osd tree | grep 27
27   hdd 0.04790         osd.27                 down  1.00000 1.00000
```

4. **OSD**를 삭제합니다. 다음 예제 명령은 **OSD 27** 을 삭제합니다.

```
[root@overcloud-controller-0 ~]# ceph osd destroy 27 --yes-i-really-mean-it
destroyed osd.27
```

12.3. 시스템에서 이전 디스크 제거 및 교체 디스크 설치

교체할 OSD가 있는 컨테이너 호스트에서 이전 디스크를 시스템에서 제거하고 대체 디스크를 설치합니다.

사전 요구 사항

- 장치 ID가 변경되었는지 확인합니다. 자세한 내용은 12.1절. "장치 이름 변경 여부 확인"의 내용을 참조하십시오.

ceph-volume 명령은 Ceph 컨테이너에 있지만 Overcloud 노드에 설치되지 않습니다. ceph-volume 명령이 Ceph 컨테이너 내에서 ceph-volume 바이너리를 실행하도록 별칭을 만듭니다. 그런 다음 ceph-volume 명령을 사용하여 새 디스크를 정리하고 OSD로 추가합니다.

절차

1. 실패한 OSD가 실행 중이 아닌지 확인합니다.

```
systemctl stop ceph-osd@27
```

2. ceph 컨테이너 이미지의 이미지 ID를 식별하고 IMG 라는 환경 변수에 저장합니다.

```
IMG=$(podman images | grep ceph | awk {'print $3'})
```

3. ceph-volume 진입점과 관련 디렉터리를 사용하여 $IMG Ceph 컨테이너 내부에서 실행되도록 ceph-volume 명령을 별칭합니다.

```
alias ceph-volume="podman run --rm --privileged --net=host --ipc=host -v
/run/lock/lvm:/run/lock/lvm:z -v /var/run/udev/:/var/run/udev/:z -v /dev:/dev -v
/etc/ceph:/etc/ceph:z -v /var/lib/ceph/:/var/lib/ceph/:z -v /var/log/ceph/:/var/log/ceph/:z --
entrypoint=ceph-volume $IMG --cluster ceph"
```

4. 별칭 명령이 성공적으로 실행되는지 확인합니다.

```
ceph-volume lvm list
```

5. 새 OSD 장치가 LVM에 포함되어 있지 않은지 확인합니다. pvdisplay 명령을 사용하여 장치를 검사하고 VG Name(VG 이름) 필드가 비어 있는지 확인합니다. <NEW_DEVICE> 를 새 OSD 장치의 /dev/* 경로로 바꿉니다.

```
[root@overcloud-computehci-2 ~]# pvdisplay <NEW_DEVICE>
  --- Physical volume ---
  PV Name               /dev/sdj
  VG Name               ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2
  PV Size               50.00 GiB / not usable 1.00 GiB
  Allocatable           yes (but full)
  PE Size               1.00 GiB
  Total PE              49
  Free PE               0
  Allocated PE          49
  PV UUID               kOO0lf-ge2F-UH44-6S1z-9tAv-7ypT-7by4cp
[root@overcloud-computehci-2 ~]#
```

VG Name(VG 이름) 필드가 비어 있지 않으면 장치는 제거해야 하는 볼륨 그룹에 속합니다.

6. **장치가 볼륨 그룹에 속하는 경우 lvdisplay 명령을 사용하여 볼륨 그룹에 논리 볼륨이 있는지 확인합니다. <VOLUME_GROUP> 을 pvdisplay 명령에서 검색한 VG 이름 필드 값으로 바꿉니다.**

```
[root@overcloud-computehci-2 ~]# lvdisplay | grep <VOLUME_GROUP>
  LV Path                /dev/ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2/osd-data-a0810722-
7673-43c7-8511-2fd9db1dbbc6
  VG Name                ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2
[root@overcloud-computehci-2 ~]#
```

LV Path(LV 경로) 필드가 비어 있지 않으면 장치에 제거해야 하는 논리 볼륨이 포함됩니다.

7. **새 장치가 논리 볼륨 또는 볼륨 그룹의 일부인 경우 LVM 시스템 내의 물리 볼륨으로 논리 볼륨, 볼륨 그룹 및 장치 연결을 제거합니다.**

- **<LV_PATH> 를 LV Path 필드 값으로 바꿉니다.**
- **<VOLUME_GROUP> 을 VG 이름 필드 값으로 바꿉니다.**
- **<NEW_DEVICE> 를 새 OSD 장치의 /dev/* 경로로 바꿉니다.**

```
[root@overcloud-computehci-2 ~]# lvremove --force <LV_PATH>
  Logical volume "osd-data-a0810722-7673-43c7-8511-2fd9db1dbbc6" successfully
removed
```

```
[root@overcloud-computehci-2 ~]# vgremove --force <VOLUME_GROUP>
  Volume group "ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2" successfully removed
```

```
[root@overcloud-computehci-2 ~]# pvremove <NEW_DEVICE>
  Labels on physical volume "/dev/sdj" successfully wiped.
```

8. 새 OSD 장치가 정리되었는지 확인합니다. 다음 예에서 장치는 /dev/sdj 입니다.

```
[root@overcloud-computehci-2 ~]# ceph-volume lvm zap /dev/sdj
--> Zapping: /dev/sdj
--> --destroy was not specified, but zapping a whole device will remove the partition table
Running command: /usr/sbin/wipefs --all /dev/sdj
Running command: /bin/dd if=/dev/zero of=/dev/sdj bs=1M count=10
 stderr: 10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.010618 s, 988 MB/s
--> Zapping successful for: <Raw Device: /dev/sdj>
[root@overcloud-computehci-2 ~]#
```

9. **ceph-volume** 이 **OSD**를 시작하지 않도록 새 장치를 사용하지만 **--no-systemd** 를 전달하여 기존 **OSD ID**로 새 **OSD**를 만듭니다. 컨테이너 내에서는 이 작업을 수행할 수 없습니다.

```
ceph-volume lvm create --osd-id 27 --data /dev/sdj --no-systemd
```

10. 컨테이너 외부에서 **OSD**를 시작합니다.

```
systemctl start ceph-osd@27
```

12.4. 디스크 교체에 성공했는지 확인

디스크 교체에 성공했는지 확인하려면 언더클라우드에서 다음 단계를 완료합니다.

절차

1. 장치 이름이 변경되었는지 확인하고 **Ceph**를 배포하는 데 사용한 이름 지정 방법에 따라 장치 목록을 업데이트합니다. 자세한 내용은 **12.1**절. “장치 이름 변경 여부 확인”의 내용을 참조하십시오.

2. 변경 사항에 불일치가 발생하지 않도록 하려면 **overcloud deploy** 명령을 다시 실행하여 스택 업데이트를 수행합니다.

3. 장치 목록이 다른 호스트가 있는 경우 예외를 정의해야 할 수 있습니다. 예를 들어 다음 예제 **heat** 환경 파일을 사용하여 **OSD** 장치 **3**개가 있는 노드를 배포할 수 있습니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

CephAnsibleDisksConfig 매개변수는 **OSD**를 호스팅하는 모든 노드에 적용되므로 **devices** 매개 변수를 새 **device** 목록으로 업데이트할 수 없습니다. 대신 다른 장치 목록이 있는 새 호스트에 대한 예외를 정의해야 합니다. 예외 정의에 대한 자세한 내용은 **5.5**절. “유사한 **Ceph Storage** 노드의 매개변수 덮어쓰기” 및 **5.5.1.2**절. “**Ceph Storage** 노드의 디스크 레이아웃 변경” 을 참조하십시오.

부록 A. 샘플 환경 파일: CEPH STORAGE 클러스터 생성

다음 사용자 지정 환경 파일은 2장. *오버클라우드 배포를 위한 Ceph Storage 노드 준비* 전체에서 설명된 많은 옵션을 사용합니다. 이 샘플에는 주석 처리된 옵션이 포함되지 않습니다. 환경 파일에 대한 개요는 환경 파일 (Advanced Overcloud Customization 가이드의)을 참조하십시오.

/home/stack/templates/storage-config.yaml

```
parameter_defaults: 1
  CinderBackupBackend: ceph 2
  CephAnsibleDisksConfig: 3
    osd_scenario: lvm
    osd_objectstore: bluestore
    dmcrypt: true
    devices:
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:10:0
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:11:0
      - /dev/nvme0n1
  ControllerCount: 3 4
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
  CephMdsCount: 3
  OvercloudCephMdsFlavor: ceph-mds
  NeutronNetworkType: vxlan 5
```

1

parameter_defaults 섹션에서는 모든 템플릿의 매개 변수의 기본값을 수정합니다. 여기에 나열된 대부분의 항목은 4장. *스토리지 서비스 사용자 정의* 에 설명되어 있습니다.

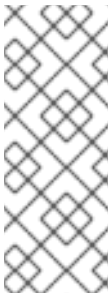
2

Ceph Object Gateway를 배포하는 경우 **Ceph Object Storage(ceph-rgw)**를 백업 대상으로 사용할 수 있습니다. 이를 구성하려면 **CinderBackupBackend** 를 **swift** 로 설정합니다. 자세한 내용은 4.2절. “Ceph Object Gateway 활성화” 을 참조하십시오.

3

4

각 역할에 대해 *Count 매개 변수는 Overcloud*Flavor 매개 변수가 플레이버를 할당하는 동안 여러 개의 노드를 할당합니다. 예를 들면 ControllerCount입니다. 3 컨트롤러 역할에 3개의 노드를 할당하고 OvercloudControlFlavor: control 은 제어 플레이버 를 사용하도록 각 역할을 설정합니다. 자세한 내용은 7.1절. “역할에 노드 및 플레이버 할당” 을 참조하십시오.



참고

CephMonCount,CephMdsCount,OvercloudCephMonFlavor 및 OvercloudCephMdsFlavor 매개변수(ceph-mon 및 ceph-mds 플레이버와 함께)는 3장. *전용 노드에 Ceph 서비스 배포* 에 설명된 대로 사용자 지정 CephMON 및 CephMds 역할을 생성한 경우에만 유효합니다.

5

NeutronNetworkType: neutron 서비스에서 사용해야 하는 네트워크 유형을 설정합니다(이 경우 vxlan).

부록 B. 샘플 사용자 정의 인터페이스 템플릿: 여러 개의 결합된 인터페이스

다음 템플릿은 사용자 지정 버전의 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 입니다. 4.5절. “Ceph 노드에 여러 개의 본딩 인터페이스 구성” 에 설명된 대로 백엔드 및 프론트엔드 스토리지 네트워크 트래픽을 격리하는 여러 개의 결합된 인터페이스가 포함되어 있으며, 두 연결에 대한 중복을 격리합니다.

또한 4.5.1절. “본딩 모듈 지시문 구성” 에 설명된 대로 사용자 지정 본딩 옵션 **'mode=4 lacp_rate=1'** 을 사용합니다.

/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml(사용자 정의)

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with VLANs attached for the ceph storage role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
```

```
  BondInterfaceOvsOptions:
    default: 'mode=4 lacp_rate=1'
    description: The bonding_options string for the bond interface. Set
                 things like lacp=active and/or bond_mode=balance-slb
                 using this option.
    type: string
    constraints:
      - allowed_pattern: "^((?!balance.tcp).)*$"
        description: |
          The balance-tcp bond mode is known to cause packet loss and
          should not be used in BondInterfaceOvsOptions.
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
```

```
    description: The IP address of the EC2 metadata server.
    type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
                -
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
            -
              type: ovs_bridge
              name: br-bond
              members:
                -
                  type: linux_bond
                  name: bond1
                  bonding_options: {get_param: BondInterfaceOvsOptions}
                  members:
                    -
                      type: interface
                      name: nic2
                      primary: true
                    -
                      type: interface
                      name: nic3
                -
                  type: vlan
                  device: bond1
                  vlan_id: {get_param: StorageNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageIpSubnet}
            -
              type: ovs_bridge
              name: br-bond2
```

```
            members:
             -
               type: linux_bond
               name: bond2
               bonding_options: {get_param: BondInterfaceOvsOptions}
               members:
                -
                  type: interface
                  name: nic4
                  primary: true
                -
                  type: interface
                  name: nic5
             -
               type: vlan
               device: bond1
               vlan_id: {get_param: StorageMgmtNetworkVlanID}
               addresses:
                -
                  ip_netmask: {get_param: StorageMgmtIpSubnet}
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```