Red Hat OpenStack Platform 13

컨테이너화된 Red Hat Ceph를 사용하여 오버클라우드 배포

컨테이너화된 Red Hat Ceph Cluster를 배포 및 사용하도록 Director 구성

Last Updated: 2022-10-16

Red Hat OpenStack Platform 13 컨테이너화된 Red Hat Ceph를 사용하여 오버클라우드 배포

컨테이너화된 Red Hat Ceph Cluster를 배포 및 사용하도록 Director 구성

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

법적 공지

초록

이 가이드에서는 Red Hat OpenStack Platform director를 사용하여 컨테이너화된 Red Hat Ceph Storage 클러스터에서 Overcloud를 생성하는 방법에 대한 정보를 제공합니다. 여기에는 director를 통해 Ceph 클러스터를 사용자 정의하는 지침이 포함됩니다.

차례

머리말

1장. RED HAT CEPH STORAGE와 오버클라우드 통합 소개

RHOSP(Red Hat OpenStack Platform) director는 *오버클라우드* 라는 클라우드 환경을 생성합니다. director는 오버클라우드에 추가 기능을 설정하는 기능을 제공합니다. 이러한 추가 기능 중 하나로 Red Hat Ceph Storage와의 통합이 포함됩니다. 여기에는 director 또는 기존 Ceph Storage 클러스터와 함께 생성된 Ceph Storage 클러스터가 모두 포함됩니다.

이 가이드에 설명된 Red Hat Ceph 클러스터는 컨테이너화된 Ceph Storage 기능을 제공합니다. RHOSP의 컨테이너화된 서비스에 대한 자세 한 내용은 *Director 설치 및 사용 가이드의 CLI 툴을 사용하여 기본 오버클라우드 구성* 을 참조하십시오.

1.1. CEPH STORAGE 정의

Red Hat Ceph Storage는 우수한 성능, 안정성 및 확장성을 제공하도록 설계된 분산 데이터 오브젝트 저장소입니다. 분산 오브젝트 저장소는 구조화되지 않은 데이터를 수용하기 때문에 스토리지의 미래이며 클라이언트가 최신 개체 인터페이스와 기존 인터페이스를 동시에 사용할 수 있기 때문입니다. 모든 Ceph 배포의 핵심은 Ceph Storage 클러스터이며, 이는 두 가지 유형의 데몬으로 구성됩니다.

Ceph Object Storage Daemon

Ceph OSD(오브젝트 스토리지 데몬)는 Ceph 클라이언트를 대신하여 데이터를 저장합니다. 또한 Ceph OSD는 Ceph 노드의 CPU 및 메모리를 사용하여 데이터 복제, 리밸런싱, 복구, 모니터링 및 보고 기능을 수행합니다.

Ceph Monitor

Ceph 모니터(MON)는 스토리지 클러스터의 현재 상태와 함께 Ceph Storage 클러스터 맵의 마스터 사본을 유지 관리합니다.

Red Hat Ceph Storage에 대한 자세한 내용은 Red Hat Ceph Storage 아키텍처 가이드를 참조하십시오 .

참고

NFS를 통한 Ceph 파일 시스템(CephFS)이 지원됩니다. 자세한 내용은 공유 파일 시스템 서비스의 NFS 백엔드 가이드를 통해 CephFS를 참조하십시오.

1.2. 시나리오 정의

이 가이드에서는 컨테이너화된 Red Hat Ceph 클러스터를 오버클라우드와 함께 배포하는 방법을 설명합니다. *이를 위해 director는* ***ceph-ansible*** *패키지를 통해 제공된 Ansible 플레이북을 사용합니다. director는 클러스터의 구성 및 스케일링 작업도 관리합니다.*

1.3. 요구 사항 설정

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

Red Hat OpenStack Platform director를 사용하여 Ceph Storage 노드를 생성하는 경우 다음 노드에 대한 요구 사항에 유의하십시오.

배치 그룹

Ceph는 배치 그룹을 사용하여 대규모의 효율적인 동적 오브젝트 추적을 지원합니다. OSD 오류가 발생하거나 클러스터를 재조정하는 경우 Ceph에서 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 요구 사항에 따라 올바른 배치 그룹 수를 계산하는 것이 중요합니다. 배치 그룹 계산기를 사용하여 올바른 개수를 계산할 수 있습니다. 풀당 PG(배치 그룹) 계산기

프로세서

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

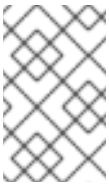
메모리

일반적으로 OSD 호스트당 16GB의 RAM과 OSD 데몬당 추가 2GB의 RAM으로 기준선을 정하는 것이 좋습니다.

디스크 레이아웃

크기 조정은 스토리지 요구 사항에 따라 다릅니다. 권장되는 Red Hat Ceph Storage 노드 구성에는 다음과 유사한 레이아웃에 3개 이상의 디스크가 필요합니다.

- ***/dev/sda*** *- root 디스크입니다. director가 주요 Overcloud 이미지를 디스크에 복사합니다. 최소 50GB의 사용 가능한 디스크 공간이어야 합니다.*
- ***/dev/sdb*** *- 저널 디스크입니다. 이 디스크는 Ceph OSD 저널용 파티션으로 나뉩니다. 예를 들어 **/dev/sdb1**,**/dev/sdb2**,**/dev/sdb3** 등입니다. 저널 디스크는 일반적으로 시스템 성능을 지원하기 위한 SSD(솔리드 스테이트 드라이브)입니다.*
- ***/dev/sdc*** *이후 - OSD 디스크입니다. 스토리지 요구 사항에 따라 필요한 개수만큼 디스크를 사용합니다.*



참고

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

네트워크 인터페이스 카드

최소 1개의 1Gbps 네트워크 인터페이스 카드가 필요합니다. 프로덕션 환경에서는 적어도 두 개 이상의 NIC를 사용하는 것이 좋습니다. 본딩된 인터페이스나 태그된 VLAN 트래픽 위임에는 추가 네트워크 인터페이스 카드를 사용하십시오. 대량의 트래픽에 서비스를 제공하는 OpenStack Platform 환경을 구축하는 경우 특히 스토리지 노드에 10Gbps 인터페이스를 사용하는 것이 좋습니다.

전원 관리

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

이미지 속성

*Red Hat Ceph Storage 블록 장치 성능을 개선하기 위해 **virtio-scsi** 드라이버를 사용하도록 Glance 이미지를 구성할 수 있습니다. 이미지의 권장 이미지 속성에 대한 자세한 내용은 Red Hat Ceph Storage 설명서의 Glance 연결을 참조하십시오.*

이 가이드에는 다음이 필요합니다.

- *Red Hat OpenStack Platform director가 설치된 Undercloud 호스트. Director 설치 및 사용 가이드의 언더클라우드 설치를 참조하십시오.*
- *Red Hat Ceph Storage에 대한 추가 하드웨어 권장 사항. Red Hat Ceph Storage Hardware selection 가이드를 참조하십시오.*

중요

Ceph Monitor 서비스가 오버클라우드 컨트롤러 노드에 설치됩니다. 즉, 성능 문제를 완화하기 위해 적절한 리소스를 제공해야 합니다. 환경의 컨트롤러 노드가 Ceph 모니터 데이터를 위해 메모리 및 SSD(Solid-State Drive) 스토리지에 16GB 이상의 RAM을 사용하는지 확인합니다. 중간 규모의 Ceph 설치에서는 최소 500GB의 Ceph 모니터 데이터를 제공합니다. 클러스터가 불안정할 경우 levelDB 증가를 피하기 위해 이 공간이 필요합니다.

1.4. 추가 리소스

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

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

주의

ceph-ansible *은 플레이북을 사용하여 컨테이너화된 Ceph Storage를 배포하지만 이러한 파일을 편집하여 배포를 사용자 정의하지 마십시오. 대신 heat 환경 파일을 사용하여 이러한 플레이북에서 설정한 기본값을 재정의합니다.* ***ceph-ansible*** *플레이북을 직접 편집하면 배포가 실패합니다.*

컨테이너화된 Ceph Storage에 대해 director가 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 방식을 더 깊이 이해해야 합니다. 자세 한 내용은 Heat 템플릿 및 환경 파일 이해 를 참조하십시오.

마지막으로 OpenStack의 컨테이너화된 서비스에 대한 자세 한 내용은 Director 설치 및 사용 가이드의 CLI 툴을 사용하여 기본 오버클라우드 구성 을 참조하십시오.

2장. 오버클라우드 노드 준비

이 시나리오의 모든 노드는 전원 관리에 IPMI를 사용하는 베어 메탈 시스템입니다. director가 Red Hat Enterprise Linux 7 이미지를 각 노드에 복사하므로 이러한 노드에는 운영 체제가 필요하지 않습니다. 또한 여기에 설명된 노드의 Ceph Storage 서비스는 컨테이너화되어 있습니다. director는 인트로스펙션 및 프로비저닝 프로세스 중에 프로비저닝 네트워크를 통해 각 노드와 통신합니다. 모든 노드는 기본 VLAN을 통해 이 네트워크에 연결됩니다.

2.1. CEPH STORAGE 노드 디스크 정리

Ceph Storage OSD 및 저널 파티션에는 GPT 디스크 레이블이 필요합니다. 즉, Ceph OSD 서비스를 설치하기 전에 Ceph Storage의 추가 디스크를 GPT로 변환해야 합니다. 이 작업을 수행하려면 모든 메타데이터를 디스크에서 삭제해야 합니다. 이렇게 하면 director에서 GPT 레이블을 설정할 수 있습니다.

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

```
clean_nodes=true
```

이 옵션을 사용하면 베어 메탈 프로비저닝 서비스에서 노드를 부팅하고 노드가 **사용** 가능하게 설정될 때마다 디스크를 정리 하는 추가 단계를 실행합니다. 그러면 첫 번째 인트로스펙션 이후 및 각 배포 전에 추가 전원 사이클이 추가됩니다. 베어 메탈 프로비저닝 서비스에서는 **wipefs --force --all** 을 사용하여 정리를 수행합니다.

이 옵션을 설정한 후 **openstack undercloud install** 명령을 실행하여 이 설정 변경 사항을 실행합니다.

주의

De **fs --force --all** 은 디스크의 모든 데이터 및 메타데이터를 삭제하지만 안전한 삭제 작업을 수행하지 않습니다. 안전한 클리닝은 훨씬 더 오래 걸립니다.

2.2. 노드 등록

노드 정의 템플릿(**instackenv.json**)은 JSON 형식 파일이며 노드 등록에 필요한 하드웨어 및 전원 관리 세부 정보를 포함합니다. 예를 들면 다음과 같습니다.

```
{
    "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"
    }
  ]
}
```

템플릿을 생성한 후 stack 사용자의 홈 디렉터리(**/home/stack/instackenv.json**)에 파일을 저장합니다. stack 사용자를 초기화한 다음 **instackenv.json** 을 director로 가져옵니다.

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

이렇게 하면 템플릿을 가져와서 템플릿의 각 노드를 director에 등록합니다.

각 노드에 커널 및 ramdisk 이미지를 할당합니다.

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

이제 노드가 director에 등록 및 설정됩니다.

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 확인하기 위해 실행할 수 있는 Ansible 기반 검증이 포함되어 있습니다. 이러한 검증은 일반적인 문제가 발생하기 전에 오버클라우드 배포 실패를 방지하는 데 도움이 될 수 있습니다.

절차

ceph-ansible 패키지의 수정 버전이 설치되어 있는지 확인합니다.

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

2.3.2. 사전 프로비저닝된 노드의 패키지 확인

오버클라우드 배포에서 사전 프로비저닝된 노드를 사용하는 경우 Ceph 서비스를 호스팅하는 오버클라우드 노드에 필요한 패키지가 서버에 있는지 확인할 수 있습니다.

사전 프로비저닝된 노드에 대한 자세 한 내용은 사전 프로비저닝된 노드를 사용하여 기본 오버클라우드 구성을 참조하십시오.

절차

서버에 필수 패키지가 포함되어 있는지 확인합니다.

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

2.4. 수동으로 노드 태그 지정

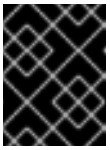
각 노드를 등록한 후 하드웨어를 검사하고 노드를 특정 프로필에 태그해야 합니다. 프로필 태그는 플레이버에 따라 노드에 일치하며, 그런 다음 플레이버가 배포 역할에 할당됩니다.

새 노드를 검사하고 태그를 지정하려면 다음 단계를 수행합니다.

1. 하드웨어 인트로스펙션을 트리거하여 각 노드의 하드웨어 속성을 검색합니다.

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

- ***--all-manageable*** *옵션은 관리 상태의 노드만 인트로스펙션합니다. 이 예제에서는 모든 것입니다.*
- ***--provide*** *옵션은 인트로스펙션 이후 모든 노드를* ***활성*** *상태로 재설정합니다.*



중요

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

2. *노드 목록을 검색하여 해당 UUID를 확인합니다.*

```
$ openstack baremetal node list
```

3. *각 노드의* ***properties/capabilities*** *매개변수에* ***profile*** *옵션을 추가하여 노드를 특정 프로필에 수동으로 태그합니다.*
예를 들어 일반적인 배포에서는 세 개의 프로필인 ***control***,***compute***, ***ceph-storage*** *를 사용합니다. 다음 명령은 각 프로필에 대해 세 개의 노드를 태그합니다.*

```
$ ironic node-update 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 5e3b2f50-fcd9-4404-b0a2-59d79924b38e add
properties/capabilities='profile:control,boot_option:local'
$ ironic node-update 484587b2-b3b3-40d5-925b-a26a2fa3036f add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d010460b-38f2-4800-9cc4-d69f0d067efe add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update d930e613-3e14-44b9-8240-4f3559801ea6 add
properties/capabilities='profile:compute,boot_option:local'
$ ironic node-update da0cc61b-4882-45e0-9f43-fab65cf4e52b add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update b9f70722-e124-4650-a9b1-aade8121b5ed add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update 68bf8f29-7731-4148-ba16-efb31ab8d34f add
properties/capabilities='profile:ceph-storage,boot_option:local'
```

작은 정보

Ceph MON 및 Ceph MDS 서비스의 노드에 태그를 지정하도록 새 사용자 지정 프로필을 구성할 수도 있습니다. 자세한 내용은 3장. 전용 노드에 기타 Ceph 서비스 배포 을 참조하십시오.

profile *옵션을 추가하면 각 프로필에 노드를 태그합니다.*

참고

수동 태그 지정 대신 AHC(Automated Health Check) 도구를 사용하여 벤치마킹 데이터를 기반으로 다수의 노드를 자동으로 태그합니다.

2.5. 루트 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 여러 속성을 정의할 수 있습니다.

- ***model***(*문자열*): *장치 식별자*
- ***vendor***(*문자열*): *장치 벤더*
- ***serial***(*문자열*): *디스크 일련번호*
- ***hctl***(*문자열*): *Host:Channel:Target:Lun (SCSI 용)*
- ***size***(*정수*): *장치의 크기(GB 단위)*
- ***wwn***(*문자열*): *고유한 스토리지 식별자*
- ***wwn_with_extension***(*문자열*): *벤더 확장이 첨부된 고유한 스토리지 식별자*
- ***wwn_vendor_extension***(*문자열*): *고유한 벤더 스토리지 식별자*
- ***rotational***(*부울*): *회전 장치인 경우(HDD) True, 그렇지 않은 경우 false(SSD)*
- ***name***(*문자열*): *장치의 이름(예: /dev/sdb1)*
- ***by_path*** (*문자열*): *장치의 고유한 PCI 경로입니다. 장치의 UUID를 사용하지 않으려면 이 속성을 사용합니다.*

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.*

일련번호를 사용하여 root 장치를 지정하려면 다음 단계를 완료합니다.

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 노드1개의 데이터에서 디스크3개가 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. 노드 정의의 **root_device** 매개변수로 변경합니다. 다음 예제에서는 루트 장치를 일련 번호로 **61866da04f380d001ea4e13c12e36ad6** 으로 설정하는 방법을 보여줍니다.

```
(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. 네트워크에서 먼저 부팅한 다음, root 디스크에서 부팅하도록 부팅 순서를 구성합니다.

director가 root 디스크로 사용할 특정 디스크를 식별합니다. **openstack overcloud deploy** 명령을 실행하면 director가 Overcloud 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 대한 자세한 내용은 Obtaining images for overcloud nodes.

절차

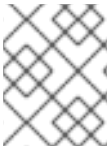
1. ***overcloud-minimal*** *이미지를 사용하도록 director를 구성하려면 다음 이미지 정의가 포함된 환경 파일을 생성합니다.*

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



참고

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

3장. 전용 노드에 기타 CEPH 서비스 배포

기본적으로 director는 컨트롤러 노드에 Ceph MON 및 Ceph MDS 서비스를 배포합니다. 이는 소규모 배포에 적합합니다. 그러나 대규모 배포에서는 Ceph 클러스터의 성능을 향상시키기 위해 전용 노드에 Ceph MON 및 Ceph MDS 서비스를 배포하는 것이 좋습니다. 둘 중 하나에 대한 사용자 지정 역할 을 생성하여 이 작업을 수행할 수 있습니다.

참고

사용자 지정 역할에 대한 자세 한 내용은 Advanced Overcloud Customization 가이드의 새 역할 생성 가이드를 참조하십시오.

director는 다음 파일을 모든 오버클라우드 역할에 대한 기본 참조로 사용합니다.

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

사용자 지정 역할을 추가할 수 있도록 이 파일을 **/home/stack/templates/** 에 복사합니다.

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

나중에 오버클라우드 생성 중에 **/home/stack/templates/roles_data_custom.yaml** 파일을 호출합니다.7.2절. "오버클라우드 배포 시작" 다음 하위 섹션에서는 Ceph MON 및 Ceph MDS 서비스에 대한 사용자 지정 역할을 구성하는 방법을 설명합니다.

3.1. CEPH MON 서비스의 사용자 지정 역할 및 플레이버 생성

이 섹션에서는 Ceph MON 역할에 대한 사용자 지정 역할(**CephMon**) 및 플레이버(**ceph-mon**)를 생성하는 방법에 대해 설명합니다. 3장. 전용 노드에 기타 Ceph 서비스 배포 에 설명된 대로 기본 역할 데이터 파일의 사본이 이미 있어야 합니다.

1. **/home/stack/templates/roles_data_custom.yaml** 파일을 엽니다.
2. 컨트롤러 역할의 에서 Ceph MON 서비스(예: **OS::TripleO::Services::CephMon**)의 서비스 항목을 제거합니다.
3. 컨트롤러 역할에 **OS::TripleO::Services::CephClient** 서비스를 추가합니다.

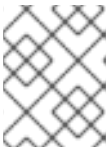
```
[...]
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
   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

4. **roles_data_custom.yaml** 이 끝나면 Ceph MON 서비스와 기타 모든 필수 노드 서비스가 포함된 사용자 지정 **CephMon** 역할을 추가합니다. 예를 들면 다음과 같습니다.

```
- 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::Fluentd
    - 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 사용하여 이 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 내용을 보려면 **openstack flavor create --help** 를 실행하십시오.

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 실행하십시오.

7. 새 **ceph-mon** 프로필에 노드를 태그합니다.

```
$ ironic node-update UUID add properties/capabilities='profile:ceph-mon,boot_option:local'
```

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

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

노드의 태그 지정에 대한 자세한 내용은 2.4절. "수동으로 노드 태그 지정" 을 참조하십시오. 사용자 지정 역할 프로필에 대한 관련 정보는 프로필 태그 지정을 참조하십시오.

3.2. CEPH MDS 서비스의 사용자 정의 역할 및 플레이버 생성

이 섹션에서는 Ceph MDS 역할에 대한 사용자 지정 역할(**CephMDS**) 및 플레이버(이름 **ceph-mds**)를 생성하는 방법을 설명합니다. 3장. 전용 노드에 기타 Ceph 서비스 배포 에 설명된 대로 기본 역할 데이터 파일의 사본이 이미 있어야 합니다.

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

2. Controller 역할의 에서 Ceph MDS 서비스(예: **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 행을 주석 처리하십시오. 이 동일한 서비스가 다음 단계에서 사용자 지정 역할에 추가됩니다.

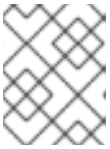
3. **roles_data_custom.yaml** 이 끝나면 Ceph MDS 서비스와 기타 모든 필수 노드 서비스가 포함된 사용자 지정 **CephMDS** 역할을 추가합니다. 예를 들면 다음과 같습니다.

```
- 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::Fluentd
    - 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 Client 서비스에서 설정할 수 있는 관리자 인증 키가 필요합니다. Ceph MON 서비스 없이 전용 노드에 Ceph MDS를 배포하고 있으므로 역할에 Ceph 클라이언트 서비스를 포함합니다.*

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

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



참고

이 명령에 대한 자세한 내용을 보려면 ***openstack flavor create --help*** *를 실행하십시오.*

5. *이 플레이버를* ***ceph-mds*** *라는 새 프로필에 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 실행하십시오.*

노드를 새 ***ceph-mds*** *프로필에 태그합니다.*

```
$ ironic node-update UUID add properties/capabilities='profile:ceph-mds,boot_option:local'
```

노드의 태그 지정에 대한 자세한 내용은 2.4절. "수동으로 노드 태그 지정" 을 참조하십시오. 사용자 지정 역할 프로필에 대한 관련 정보는 프로필 태그 지정을 참조하십시오.

4장. 스토리지 서비스 사용자 정의

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

/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml 환경 파일은 Ceph 클러스터를 생성하여 배포 시 오버클라우드와 통합합니다. 이 클러스터에는 컨테이너화된 Ceph Storage 노드가 포함되어 있습니다. OpenStack의 컨테이너화된 서비스에 대한 자세 한 내용은 Director 설치 및 사용 가이드의 CLI 툴을 사용하여 기본 오버클라우드 구성 을 참조하십시오.

Red Hat OpenStack director는 배포된 Ceph 클러스터에 기본 설정도 적용합니다. 사용자 지정 환경 파일이 있어야 Ceph 클러스터에 사용자 지정 설정을 전달합니다.

절차

1. **/home/stack/templates/** 에 **storage-config.yaml** 파일을 생성합니다. 이 문서의 목적을 위해 **~/templates/storage-config.yaml** 에는 사용자 환경에 대한 대부분의 오버클라우드 관련 사용자 지정 설정이 포함되어 있습니다. director가 오버클라우드에 적용한 모든 기본 설정을 덮어씁니다.

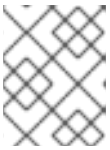
2. **~/templates/storage-config.yaml** 에 **parameter_defaults** 섹션을 추가합니다. 이 섹션에는 오버클라우드의 사용자 정의 설정이 포함되어 있습니다. 예를 들어 **vxlan** 을 Networking 서비스(neutron)의 네트워크 유형으로 설정하려면 다음을 수행합니다.

```
parameter_defaults:
  NeutronNetworkType: vxlan
```

3. 선택 사항: 필요에 따라 **parameter_defaults** 에서 다음 옵션을 설정할 수 있습니다.

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” 을 참조하십시오.	Ceph
NovaEnableRbdBackend	Nova 임시 스토리지용 Ceph Storage 사용	true
GlanceBackend	Image 서비스에서 사용할 백엔드(Ceph), **swift** 또는 **파일**을 정의합니다.	rbd

옵션	설명	기본값
GnocchiBackend	원격 분석 서비스에서 사용할 백엔드(Ceph), **swift** 또는 **파일**을 정의합니다.	rbd



참고

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

환경 파일의 내용은 다음 섹션에 적용된 설정에 따라 변경됩니다. 완료된 예제는 부록 A. 샘플 환경 파일: Ceph 클러스터 생성 을 참조하십시오.

다음 하위 섹션에서는 director가 적용하는 일반적인 기본 스토리지 서비스 설정을 재정의하는 방법을 설명합니다.

4.1. CEPH 메타데이터 서버 활성화

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

참고

Red Hat은 공유 파일 시스템 서비스의 NFS 백엔드를 통해 CephFS를 사용하여 Ceph MDS 배포를 지원합니다.

Ceph Metadata Server를 활성화하려면 오버클라우드를 생성할 때 다음 환경 파일을 호출합니다.

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

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

참고

기본적으로 Ceph 메타데이터 서버는 컨트롤러 노드에 배포됩니다. Ceph 메타데이터 서버를 자체 전용 노드에 배포할 수 있습니다. 3.2 절. "Ceph MDS 서비스의 사용자 정의 역할 및 플레이버 생성"

4.2. CEPH OBJECT GATEWAY 활성화

*Ceph Object Gateway(RGW)는 Ceph Storage 클러스터 내의 오브젝트 스토리지 기능에 대한 인터페이스를 제공합니다. RGW를 배포할 때 기본 Object Storage 서비스(**swift**)를 Ceph로 교체할 수 있습니다. 자세한 내용은 Red Hat Enterprise Linux의 오브젝트 게이트웨이 가이드를 참조하십시오 .*

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 Storage에서는 OSD당 250개의 배치 그룹을 허용합니다. RGW를 활성화하면 Ceph Storage에서 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 속한 영역의 이름으로 교체됩니다.

따라서 RGW를 활성화하면 새 풀을 고려하도록 **CephPoolDefaultPgNum** 매개변수를 사용하여 기본 **pg_num** 을 설정해야 합니다. Ceph 풀의 배치 그룹 수를 계산하는 방법에 대한 자세한 내용은 6.9절. "다양한 Ceph 풀에 사용자 정의 속성 할당" 을 참조하십시오.

Ceph Object Gateway는 기본 Object Storage 서비스의 드롭인 대체 역할을 합니다. 따라서 일반적으로 **swift** 를 사용하는 다른 모든 서비스는 추가 구성 없이 Ceph Object Gateway를 원활하게 사용할 수 있습니다. 예를 들어 Ceph Object Gateway를 사용하도록 Block Storage Backup 서비스(**cinder-backup**)를 구성할 때 **ceph** 를 대상 백엔드로 설정합니다(4.4절. "Ceph를 사용하도록 백업 서비스 구성" 참조하십시오.

4.3. 외부 CEPH 개체 게이트웨이를 사용하도록 CEPH 오브젝트 저장소 구성

RHOSP(Red Hat OpenStack Platform) director는 외부 Ceph Object Gateway(RGW)를 오브젝트 저장소 서비스로 구성을 지원합니다. 외부 RGW 서비스로 인증하려면 ID 서비스(keystone)에서 사용자 및 해당 역할을 확인하도록 RGW를 구성해야 합니다.

외부 Ceph 개체 게이트웨이를 구성하는 방법에 대한 자세한 내용은 Ceph 오브젝트 게이트웨이를 사용하여 Keystone을 사용하여 Ceph 개체 게이트웨이 구성 을 참조하십시오.

절차

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

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

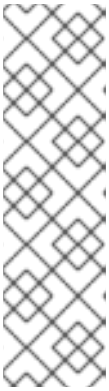
참고

코드 조각 예제에는 사용자 환경에서 사용하는 값과 다를 수 있는 매개변수 값이 포함되어 있습니다.

- *원격 RGW 인스턴스가 수신 대기하는 기본 포트는 **8080** 입니다. 외부 RGW가 구성된 방법에 따라 포트가 다를 수 있습니다.*
- *오버클라우드에서 생성된 **swift** 사용자는 **SwiftPassword** 매개변수로 정의된 암호를 사용합니다. **rgw_keystone_admin_password** 를 사용하여 ID 서비스로 인증하기 위해 동일한 암호를 사용하도록 외부 RGW 인스턴스를 구성해야 합니다.*

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

```
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. *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
```

4.4. CEPH를 사용하도록 백업 서비스 구성

*Block Storage Backup 서비스(**cinder-backup**)는 기본적으로 비활성화되어 있습니다. 활성화하려면 오버클라우드를 생성할 때 다음 환경 파일을 호출합니다.*

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

자세한 내용은 7.2 절. "오버클라우드 배포 시작" 를 참조하십시오.

***cinder-backup** 을 활성화하는 경우(4.2 절. "Ceph Object Gateway 활성화" 와 같이) Ceph 에 백업을 저장하도록 구성할 수 있습니다. 여기에는 환경 파일의 **parameter_defaults** (예: **~/templates/storage-config.yaml**) 에 다음 행을 추가해야 합니다.*

```
CinderBackupBackend: ceph
```

4.5. CEPH 노드당 여러 본딩 인터페이스 구성

결합된 인터페이스 를 사용하여 여러 NIC 를 결합하여 네트워크 연결에 중복성을 추가할 수 있습니다. Ceph 노드에 NIC 가 충분한 경우 노드당 결합된 인터페이스를 여러 개 생성하여 이 단계를 추가로 수행할 수 있습니다.

이를 통해 노드에 필요한 각 네트워크 연결에 본딩된 인터페이스를 사용할 수 있습니다. 중복과 각 네트워크에 대한 전용 연결을 제공합니다.

가장 간단한 구현에는 두 개의 본딩을 사용해야 합니다. 하나는 Ceph 노드에서 사용하는 스토리지 네트워크 각각에 하나씩 사용됩니다. 이러한 네트워크는 다음과 같습니다.

프론트 엔드 스토리지 네트워크(StorageNet)

Ceph 클라이언트는 이 네트워크를 사용하여 Ceph 클러스터와 상호 작용합니다.

백엔드 스토리지 네트워크(스토리지MgmtNet)

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.

*이를 구성하려면 director 에서 여러 결합된 NIC 를 배포하는 샘플 템플릿을 제공하지 않으므로 네트워크 인터페이스 템플릿을 사용자 지정해야 합니다. 그러나 director 는 단일 본딩 인터페이스를 배포하는 템플릿을 제공합니다. **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml**. 추가 NIC 에 대한 결합된 인터페이스를 여기에 정의하여 추가할 수 있습니다.*

참고

사용자 지정 인터페이스 템플릿 생성에 대한 자세한 내용은 Advanced Overcloud Customization 가이드에서 사용자 지정 인터페이스 템플릿 생성

*다음 스니펫에는 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 에서 정의한 단일 본딩 인터페이스에 대한 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 브릿지에는 이 템플릿에서 정의한 본딩이 있습니다. 이 행은 브리지 유형을 정의합니다(예: OVS).*

2 ***br-bond*** *브리지의 첫 번째 멤버는* ***bond1*** *이라는 본딩 인터페이스 자체입니다. 이 행은* ***bond1*** *의 본딩 유형을 정의합니다. 이는 OVS이기도 합니다.*

3 *기본 본딩의 이름은 이 행에 정의된* ***bond1*** *입니다.*

4 ***ovs_options*** *항목은 director에 특정 본딩 모듈 지시문 세트를 사용하도록 지시합니다. 이러한 지시문은* ***BondInterfaceOvsOptions*** *를 통해 전달되며 동일한 파일에서도 구성할 수 있습니다. 이를 구성하는 방법에 대한 자세한 내용은 4.5.1절. "bonding 모듈 지시문 구성" 을 참조하십시오.*

5 *bond의* ***members*** *섹션에서는* ***bond1*** *에 의해 결합된 네트워크 인터페이스를 정의합니다. 이 경우 본딩된 인터페이스는* ***nic2(*** *기본 인터페이스로 설정) 및* ***nic3*** *를 사용합니다.*

6 ***br-bond*** *브릿지에는 두 개의 다른 멤버가 있습니다. 즉, 프런트 엔드(* ***StorageNetwork****) 및 백엔드 (****StorageMgmtNetwork****) 스토리지 네트워크의 VLAN입니다.*

7 ***device*** *매개 변수는 VLAN에서 사용해야 하는 장치를 정의합니다. 이 경우 두 VLAN 모두 결합된 인터페이스* ***bond1*** *을 사용합니다.*

NIC를 두 개 이상 사용하면 추가 브리지 및 본딩 인터페이스를 정의할 수 있습니다. 그런 다음 VLAN 중 하나를 새 본딩 인터페이스로 이동할 수 있습니다. 따라서 두 스토리지 네트워크 연결에 모두 처리량과 안정성이 향상됩니다.

이러한 목적으로 ***/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml*** *을 사용자 정의할 때 기본 OVS 대신 Linux 본딩(* ***type: linux_bond*** *)을 사용하는 것이 좋습니다.* ***이 본딩 유형은 엔터프라이즈 프로덕션 배포에 더 적합합니다.***

다음 편집된 스니펫에서는 추가 OVS 브리지(br- bond2)를 정의합니다. 이 브릿지에는 bond2라는 새 Linux 본딩이 있습니다. bond2 인터페이스는 두 개의 추가 NIC(즉, 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 *OVS* 대신 *Linux* 본딩(*OVS* 대신)이며, *ovs_options* 대신 *bonding_options* 를 사용하여 본딩 지시문을 설정합니다. 관련 정보는 *4.5.1*절. "*bonding* 모듈 지시문 구성" 에서 참조하십시오.**

이 사용자 지정 템플릿의 전체 내용은 부록 B. 샘플 사용자 정의 인터페이스 템플릿: 여러 결합된 인터

페이스 을 참조하십시오.

4.5.1. bonding 모듈 지시문 구성

*결합된 인터페이스를 추가하고 구성한 후 **BondInterfaceOvsOptions** 매개변수를 사용하여 각각 사용하는 지시문을 설정합니다. **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 매개변수 섹션에서 확인할 수 있습니다. 다음 스니펫에서는 이 매개변수의 기본 정의(예: 빈)를 보여줍니다.*

```
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
```

***default:** 행에 필요한 옵션을 정의합니다. 예를 들어 **802.3ad(mode 4)**와 **LACP** 속도 **1(fast)**을 사용하려면 다음과 같이 **'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** 을 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 결합된 인터페이스 을 참조하십시오.*

5장. CEPH STORAGE 클러스터 사용자 정의

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

전제 조건

컨테이너화된 Ceph Storage를 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 5.2절. “Ceph Storage 노드 디스크 레이아웃 매핑”의 내용을 참조하십시오.
- CephConfigOverrides - 이 리소스는 기타 모든 사용자 지정 설정을 Ceph Storage 클러스터에 적용합니다.

절차

1. Red Hat Ceph Storage 3 툴 리포지토리를 활성화합니다.

```
$ sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
```

2. 언더클라우드에 ceph-ansible 패키지를 설치합니다.

```
$ sudo yum install ceph-ansible
```

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

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

참고

CephConfigOverrides 매개변수를 ceph.conf 파일의 [global] 섹션과 [osd], [mon] 및 [client] 와 같은 다른 섹션에 적용할 수 있습니다. 섹션을 지정하면 key:value 데이터가 지정된 섹션으로 이동합니다. 섹션을 지정하지 않으면 기본적으로 데이터는 [global] 섹션으로 이동합니다. Ceph Storage 구성, 사용자 정의 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 다음과 같은 설정이 생성됩니다.

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

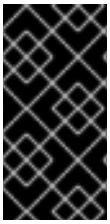
5.1. CEPH-ANSIBLE 그룹 변수 설정

ceph-ansible 툴은 Ceph Storage 클러스터를 설치 및 관리하는 데 사용되는 플레이북입니다.

group_vars 디렉터리에 대한 자세한 내용은 3.2를 참조하십시오. Red Hat Ceph Storage Cluster 설치 in the Installation Guide for Red Hat Enterprise Linux.

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

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



중요

***override** 매개변수를 사용하여 **ceph-ansible** 그룹 변수를 변경합니다. 언더클라우드의 **/usr/share/ceph-ansible** 디렉토리에서 직접 그룹 변수를 편집하지 마십시오.*

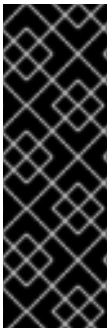
5.2. CEPH STORAGE 노드 디스크 레이아웃 매핑

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

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

변수	필수 여부	기본값(설정이 설정되지 않은 경우)	설명
osd_scenario	있음	lvm 참고: Ceph 3.2 이상을 사용하는 새 배포의 경우 **lvm** 이 기본값입니다. Ceph 3.1 이하의 경우 기본값은 col located입니다 **.**	Ceph 3.2에서 **lvm** 을 사용하면 ceph-ansible이 **ceph-volume** 을 사용하여 OSD 및 BlueStore WAL 장치를 구성할 수 있습니다. Ceph 3.1에서 값은 OSD를 생성하여 다음 중 하나를 사용하여 OSD를 생성해야 하는지와 같은 저널링 시나리오를 설정합니다. - 파일 저장(col placed)에 대해 **동일한** 장치에 배치된**경우**, 또는 - filestore 전용 장치에 저장 **됩니다** (**col 배치되지 않음**).
devices	있음	*NONE. 변수를 설정해야 합니다.*	OSD용으로 노드에서 사용할 블록 장치 목록입니다.

변수	필수 여부	기본값(설정이 설정되지 않은 경우)	설명
dedicated_devices	예 (**osd_scenario** 가 **col 배치 되지 않은 경우에만**)	devices	장치 아래의 각 항목을 전용 저널링 블록 장치에 매핑하는 블록 장치 목록입니다. **osd_scenario=non-col 배치된 경우에만 이 변수를 사용합니다**.
dmcrypt	없음	false	OSD에 저장된 데이터가 암호화되었는지(**true**) 여부(**false**)인지 여부를 설정합니다.
osd_objectstore	없음	bluestore 참고: Ceph 3.2 이상을 사용하는 새 배포의 경우 **bluestore** 가 기본값입니다. Ceph 3.1 이하의 경우 기본값은 filestore **입니다**.	Ceph에서 사용하는 스토리지 백엔드를 설정합니다.



중요

3.3 이전 버전의 ceph-ansible 을 사용하고 osd_scenario 가 col placed 또는 non- co l placed 로 설정된 경우 장치 이름 지정 불일치로 인해 OSD 재부팅 오류가 발생할 수 있습니다. 이 오류에 대한 자세한 내용은 https://bugzilla.redhat.com/show_bug.cgi?id=1670734 을 참조하십시오. 해결 방법에 대한 자세한 내용은 https://access.redhat.com/solutions/3702681 을 참조하십시오.

5.2.1. Ceph 3.2 이상에서 BlueStore 사용

참고

OpenStack Platform 13의 새로운 배포는 bluestore를 사용해야 합니다. filestore 를 사용하는 현재 배포는 Ceph 3.1 이하에서 FileStore 사용에 설명된 대로 filestore를 계속 사용해야 합니다. filestore에서 bluestore 로 의 마이그레이션 은 RHCS 3.x에서 기본적으로 지원되지 않습니다.

절차

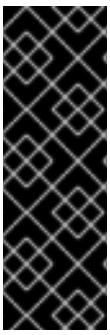
1. **Ceph OSD**로 사용할 블록 장치를 지정하려면 다음과 같은 변형을 사용합니다.

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

2. **/dev/nvme0n1** 은 더 높은 성능의 장치 클래스입니다. 다른 장치는 **SSD**이며, 예제 매개 변수 기본값은 **/dev/sdb,/dev/sdc** 및 **/dev/sdd** 에서 실행되는 **OSD 3**개를 생성합니다. **3**개의 **OSD**는 **/dev/nvme0n1** 을 **BlueStore WAL** 장치로 사용합니다. **ceph-volume** 툴은 **batch** 하위 명령을 사용하여 이 작업을 수행합니다. 각 **Ceph** 스토리지 노드에 대해 동일한 구성이 복제되며 균일한 하드웨어라고 가정합니다. **BlueStore WAL** 데이터가 **OSD**와 동일한 디스크에 있는 경우 다음과 같은 방법으로 매개변수 기본값을 변경합니다.

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

5.2.2. Ceph 3.1 이하에서 FileStore 사용



중요

기본 저널링 시나리오는 대부분의 테스트 환경과 일치하는 하드웨어 요구 사항이 낮은 **osd_scenario=co l** 배치로 설정됩니다. 그러나 일반적인 프로덕션 환경에서는 저널은 **heavier I/O** 워크로드를 수용하기 위해 **osd_scenario=non-co l** 배치된 전용 장치에 저장됩니다. 자세 한 내용은 **Red Hat Ceph Storage Hardware Selection Guide** 의 성능 사용 사례 식별 을 참조하십시오.

절차

1. **OSD**에서 사용할 각 블록 장치를 **devices** 변수 아래의 간단한 목록으로 나열하십시오. 예를 들면 다음과 같습니다.

```
devices:
  - /dev/sda
  - /dev/sdb
```

```
  - /dev/sdc
  - /dev/sdd
```

2. *선택 사항: **osd_scenario=non-co l placed**된 경우 장치의 각 항목을 **dedicated_ devices** 의 해당 항목에 매핑해야 합니다. 예를 들어 **/home/stack/templates/ceph-config.yaml** 의 다음 코드 조각:*

```
osd_scenario: non-collocated
devices:
  - /dev/sda
  - /dev/sdb
  - /dev/sdc
  - /dev/sdd

dedicated_devices:
  - /dev/sdf
  - /dev/sdf
  - /dev/sdg
  - /dev/sdg
```

결과

*결과 **Ceph** 클러스터의 각 **Ceph Storage** 노드에는 다음과 같은 특징이 있습니다.*

- */**dev**/**sda** 에 저널로 /**dev**/**sdf1** 이 있습니다.*
- */**dev**/**sdb** 에 저널로 /**dev**/**sdf2** 가 있습니다.*
- */**dev**/**sdc** 가 저널로 /**dev**/**sdg1** 을 갖습니다.*
- */**dev**/**sdd** 는 저널로 /**dev**/**sdg2** 를 갖습니다.*

5.2.3. 영구 이름이 있는 장치 참조

절차

1. *일부 노드에서 **/dev/sdb** 및 **/dev/sdc** 와 같은 디스크 경로는 재부팅 시 동일한 블록 장치를 가리키지 않을 수 있습니다. **CephStorage** 노드의 경우 각 디스크를 **/dev/disk/by-path/ symlink** 로 지정하여 배포 전반에서 블록 장치 매핑이 일관되게 유지되도록 합니다.*

```
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: 오버클라우드 배포 전에 **OSD** 장치 목록을 설정해야 하므로 디스크 장치의 **PCI** 경로를 식별하고 설정할 수 없습니다. 이 경우 인트로스펙션 중에 블록 장치에 대한 **/dev/disk/by-path/symlink** 데이터를 수집합니다.*

*다음 예제에서는 첫 번째 명령을 실행하여 서버 **b08-h03-r620-hci** 의 언더클라우드 **Object Storage** 서비스**(swift)**에서 인트로스펙션 데이터를 다운로드하고 해당 데이터를 **b08-h03-r620-hci.json** 이라는 파일에 저장합니다. 두 번째 명령을 실행하여 **"by-path"**에 대한 **grep**에 대해 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 명명 규칙에 대한 자세한 내용은 **RHEL(Red Hat Enterprise Linux)** 관리 스토리지 장치 관리 가이드의 영구 이름 지정 가이드를 참조하십시오.*

주의

***osd_scenario: lvm** 은 **ceph-volume** 에서 구성한 대로 **bluestore** 에 대한 새 배포를 기본값으로 사용합니다. **ceph-ansible 3.2** 이상 및 **Ceph Luminous** 이상에서만 사용할 수 있습니다. **ceph-ansible 3.2**로 **filestore** 를 지원하는 매개변수는 이전 버전과 호환됩니다. 따라서 기존 **FileStore** 배포에서 **osd_objectstore** 또는 **osd_scenario** 매개변수를 변경하지 마십시오.*

5.2.4. Bare Metal 서비스 인트로스펙션 데이터에서 자동으로 유효한 JSON 파일 생성

노드별 재정의를 포함하여 **Ceph Storage** 배포에서 장치를 수동으로 사용자 지정하는 경우 실수로 오류가 발생할 수 있습니다. **director** 툴 디렉터리에는 **Bare Metal** 서비스(**ironic**) 인트로스펙션 데이터에서 유효한 **JSON** 환경 파일을 자동으로 생성하는 데 사용할 수 있는 **make_ceph_disk_list.py** 라는 유틸리티가 포함되어 있습니다.

절차

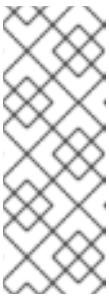
1. 배포하려는 **Ceph Storage** 노드의 **Bare Metal** 서비스 데이터베이스에서 인트로스펙션 데이터를 내보냅니다.

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

2. 유틸리티를 언더클라우드의 **stack** 사용자의 홈 디렉터리에 복사한 다음, 이를 사용하여 **openstack overcloud deploy** 명령에 전달할 수 있는 **node_data_lookup.json** 파일을 생성합니다.

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

- **i** 옵션은 ***.json** 또는 파일 목록과 같은 식을 입력할 수 있습니다.
- **k** 옵션은 **OSD** 디스크를 식별하는 데 사용되는 **ironic** 디스크 데이터 구조의 키를 정의합니다.



참고

Red Hat은 **/dev/sdd** 와 같은 장치 목록을 생성하기 때문에 **name** 을 사용하지 않는 것이 좋습니다. 이 목록이 재부팅 시 항상 동일한 장치를 가리키지는 않습니다. 대신 **by_path** 를 사용하도록 권장합니다. 이 옵션은 **-k** 가 지정되지 않은 경우 기본 옵션입니다.

참고

배포 중에 **NodeDataLookup** 만 정의할 수 있으므로 인트로스펙션 데이터 파일을 **Ceph OSD**를 호스팅하는 모든 노드에 전달할 수 있습니다. 베어 메탈 서비스는 시스템에서 사용 가능한 디스크 중 하나를 루트 디스크로 예약합니다. 유틸리티는 항상 생성된 장치 목록에서 **root** 디스크를 전사합니다.

3. *./make_ceph_disk_list.py -help 명령을 실행하여 사용 가능한 다른 옵션을 확인합니다.*

5.2.5. 디스크 레이아웃을 비Homogeneous Ceph Storage 노드에 매핑

중요

비정상적인 Ceph Storage 노드는 예측할 수 없는 성능 손실 위험과 같은 성능 문제를 일으킬 수 있습니다. Red Hat OpenStack Platform 환경에서 동종이 아닌 Ceph Storage 노드를 구성할 수 있지만 Red Hat은 권장하지 않습니다.

기본적으로 Ceph OSD를 호스팅하는 모든 노드는 5.2절. "Ceph Storage 노드 디스크 레이아웃 매핑" 에서 설정한 글로벌 장치 와 dedicated_devices 목록을 사용합니다.

이 기본 구성은 모든 Ceph OSD 노드에 동종 하드웨어가 있는 경우 적합합니다. 그러나 이러한 서버의 하위 집합에 동종 하드웨어가 없는 경우 director에서 노드별 디스크 구성을 정의해야 합니다.

참고

Ceph OSD를 호스팅하는 노드를 식별하려면 roles_data.yaml 파일을 검사하고 OS::TripleO::Services::CephOSD 서비스가 포함된 모든 역할을 식별합니다.

노드별 구성을 정의하려면 각 서버를 식별하고 전역 변수를 재정의하는 로컬 변수 목록을 포함하는 사용자 지정 환경 파일을 생성하고 openstack overcloud deploy 명령에 환경 파일을 포함합니다. 예를 들어 node-spec-overrides.yaml 이라는 노드별 구성 파일을 생성합니다.

개별 서버 또는 Ironic 데이터베이스에 대해 시스템의 고유한 UUID를 추출할 수 있습니다.

개별 서버의 UUID를 찾으려면 서버에 로그인하고 다음 명령을 실행합니다.

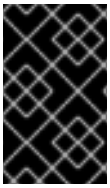
```
dmidecode -s system-uuid
```

Ironic 데이터베이스에서 UUID를 추출하려면 언더클라우드에서 다음 명령을 실행합니다.

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

주의

***undercloud.conf** 파일에 언더클라우드 설치 또는 업그레이드 및 인트로스펙션 전에 **inspection_extras = true** 가 없는 경우 시스템의 고유 **UUID**는 **Ironic** 데이터베이스에 없습니다.*



중요

*시스템의 고유 **UUID**는 **Ironic UUID**가 아닙니다.*

*유효한 **node-spec-overrides.yaml** 파일은 다음과 같습니다.*

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

*처음 두 행 이후의 모든 행은 유효한 **JSON**이어야 합니다. **JSON**이 유효한지 확인하는 쉬운 방법은 **jq** 명령을 사용하는 것입니다. 예를 들면 다음과 같습니다.*

1. *파일에서 처음 두 행(**parameter_defaults:** 및 **NodeDataLookup: |**)을 일시적으로 제거합니다.*

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

***node-spec-overrides.yaml** 파일이 증가함에 따라 **jq** 를 사용하여 포함된 **JSON**이 유효한지 확인할 수도 있습니다. 예를 들어 **devices** 및 **dedicated_devices** 목록은 길이가 동일해야 하므로 다음을 사용하여 배포를 시작하기 전에 동일한 길이인지 확인합니다.*

```
(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]$
```

*이 예에서 **node-spec-c05-h17-h21-h25-6048r.yaml** 에는 랙 **c05**에 세 개의 서버가 있으며 이 슬롯 **h17, h21, h25**는 디스크가 누락되어 있습니다. 이 섹션의 끝에 더 복잡한 예제가 포함되어 있습니다.*

***JSON** 구문을 검증한 후 환경 파일의 처음 두 줄을 다시 채우고 **-e** 옵션을 사용하여 배포 명령에 파일을 포함해야 합니다.*

*다음 예에서 업데이트된 환경 파일은 **Ceph** 배포에 **NodeDataLookup**을 사용합니다. 모든 서버에는 하나의 서버가 누락된 디스크를 제외하고 **35**개의 디스크가 있는 장치 목록이 있었습니다.*

*다음 예제 환경 파일을 사용하여 글로벌 목록 대신 사용해야 하는 디스크 목록과 **34**개의 디스크 목록이 있는 노드의 기본 장치 목록을 재정의합니다.*

```
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"
    ]
  }
}
```

5.3. CEPH STORAGE 컨테이너에서 사용 가능한 리소스 제어

Ceph Storage 컨테이너와 Red Hat OpenStack Platform 컨테이너를 동일한 서버에서 함께 배치할 때 컨테이너는 메모리 및 CPU 리소스와 경쟁할 수 있습니다.

Ceph Storage 컨테이너가 사용할 수 있는 메모리 또는 **CPU** 양을 제어하려면 다음 예와 같이 **CPU** 및 메모리 제한을 정의합니다.

```
parameter_defaults:
  CephAnsibleExtraConfig:
    ceph_mds_docker_cpu_limit: 4
    ceph_mgr_docker_cpu_limit: 1
    ceph_mon_docker_cpu_limit: 1
    ceph_osd_docker_cpu_limit: 4
    ceph_mds_docker_memory_limit: 64438m
    ceph_mgr_docker_memory_limit: 64438m
    ceph_mon_docker_memory_limit: 64438m
```

참고

표시된 제한은 예를 들어만 해당됩니다. 실제 값은 환경에 따라 달라질 수 있습니다.

주의

예제에 지정된 모든 메모리 제한의 기본값은 시스템의 총 호스트 메모리입니다. 예를 들어 **ceph-ansible** 은 **"{{ ansible_memtotal_mb }}m"** 을 사용합니다.

주의

ceph_osd_docker_memory_limit 매개변수는 예에서 의도적으로 제외됩니다. **ceph_osd_docker_memory_limit** 매개변수를 사용하지 마십시오. 자세한 내용은 **Hyper-Converged Infrastructure Guide** 에서 **Reserving Memory Resources for Ceph** 를 참조하십시오.

컨테이너가 있는 서버에 충분한 메모리 또는 **CPU**가 없거나 설계에 물리적 분리가 필요한 경우 구성 가능 서비스를 사용하여 **Ceph Storage** 컨테이너를 추가 노드에 배포할 수 있습니다. 자세한 내용은 **Advanced Overcloud Customization** 가이드의 **Composable Services and Custom Roles** 를 참조하십시오.

5.4. ANSIBLE 환경 변수 덮어쓰기

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

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

이 예제 구성으로 인해 포크 수 및 **SSH** 재시도 횟수가 증가합니다.

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

Ansible 환경 변수에 대한 자세한 내용은 **Ansible** 구성 설정 을 참조하십시오.

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

6장. RED HAT OPENSTACK PLATFORM에 2계층 CEPH 스토리지 배포

OpenStack director를 사용하면 Ceph 클러스터에서 특정 계층에 전용 새 Ceph 노드를 추가하여 다양한 Red Hat Ceph Storage 성능 계층을 배포할 수 있습니다.

예를 들어, SSD 드라이브가 있는 새로운 OSD(오브젝트 스토리지 데몬) 노드를 기존 Ceph 클러스터에 추가하여 이러한 노드에 데이터를 저장하기 위한 Block Storage(cinder) 백엔드를 독점적으로 생성할 수 있습니다. 그런 다음 새 블록 스토리지 볼륨을 생성한 사용자는 원하는 성능 계층인 HDD 또는 새 SSD를 선택할 수 있습니다.

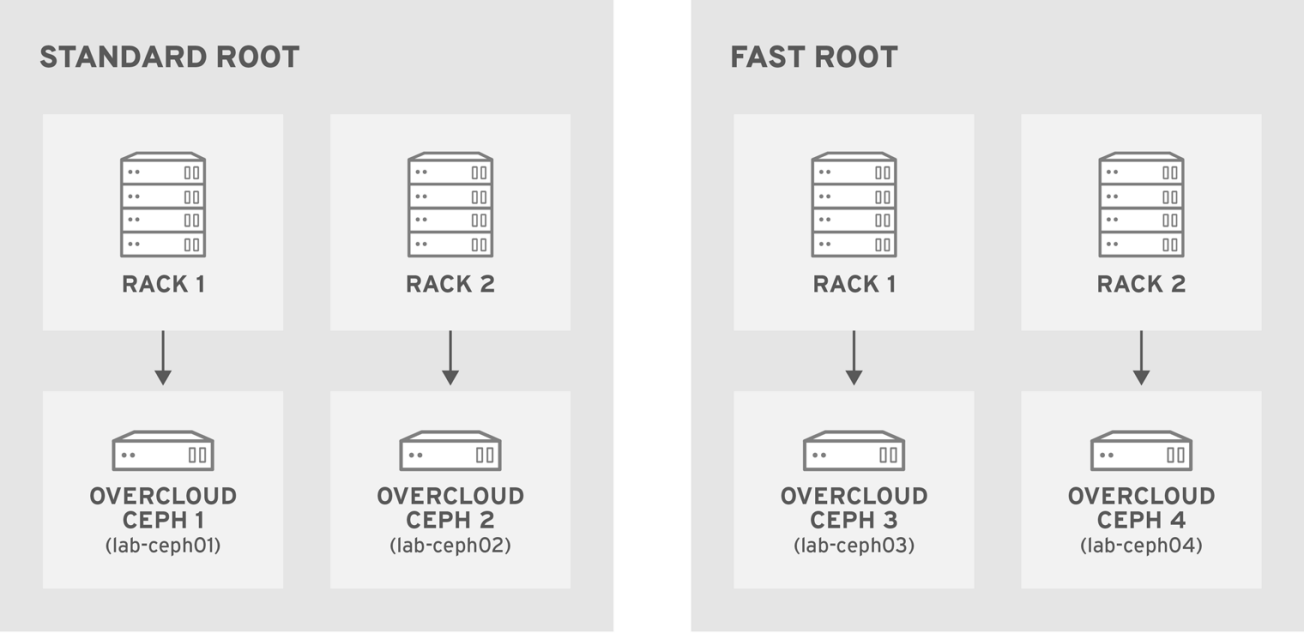
이러한 유형의 배포에는 Red Hat OpenStack Platform director가 사용자 지정된 CRUSH 맵을 ceph-ansible에 전달해야 합니다. CRUSH 맵을 사용하면 디스크 성능을 기반으로 OSD 노드를 분할할 수 있지만 이 기능을 사용하여 물리적 인프라 레이아웃을 매핑할 수도 있습니다.

다음 섹션에서는 노드 중 두 노드가 SSD를 사용하고 다른 두 개의 HDD를 사용하는 4개의 노드를 배포하는 방법을 설명합니다. 이 예제는 반복 가능한 패턴을 쉽게 전달할 수 있습니다. 그러나 프로덕션 배포에서는 Red Hat Ceph Storage 하드웨어 선택 가이드에 따라 지원되도록 더 많은 노드와 더 많은 OSD를 사용해야 합니다.

6.1. CRUSH 맵 만들기

CRUSH 맵을 사용하면 OSD 노드를 CRUSH 루트에 배치할 수 있습니다. 기본적으로 "default" root가 생성되고 모든 OSD 노드가 포함됩니다.

지정된 루트 내에서 물리적 토폴로지, 랙, 방 등을 정의한 다음 OSD 노드를 원하는 계층 구조(또는 버킷)에 배치합니다. 기본적으로 물리적 토폴로지는 정의되어 있지 않습니다. 플랫 설계는 모든 노드가 동일한 랙에 있는 것처럼 가정됩니다.



*사용자 지정 **CRUSH** 맵을 만드는 방법에 대한 자세한 내용은 스토리지 전략 가이드의 **Crush Administration** **in the Storage Strategies Guide** 를 참조하십시오.*

6.2. OSD 매핑

***OSD**를 매핑하려면 다음 단계를 완료합니다.*

절차

1. ***OSD/journal** 매핑을 선언합니다.*

```
parameter_defaults:
  CephAnsibleDisksConfig:
   devices:
   - /dev/sda
   - /dev/sdb
   dedicated_devices:
   - /dev/sdc
   - /dev/sdc
   osd_scenario: non-collocated
   journal_size: 8192
```

6.3. 복제 요소 설정

복제 요소를 설정하려면 다음 단계를 완료합니다.

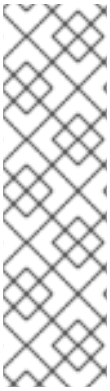
참고

이는 일반적으로 전체 **SSD** 배포에서만 지원됩니다. **Red Hat Ceph Storage:** 지원 구성을 참조하십시오.

절차

1. 기본 복제 요소를 두 개로 설정합니다. 이 예제에서는 **4**개의 노드를 두 개의 다른 루트로 분할합니다.

```
parameter_defaults:
  CephPoolDefaultSize: 2
```



참고

gnocchi를 백엔드로 사용하는 배포를 업그레이드하는 경우 배포 타임아웃이 발생할 수 있습니다. 이 타임아웃을 방지하려면 다음 **CephPool** 정의를 사용하여 **gnocchi** 풀을 사용자 지정합니다.

```
parameter_defaults
   CephPools:  {"name": metrics, "pg_num": 128, "pgp_num": 128, "size": 1}
```

6.4. CRUSH 계층 구조 정의

director는 **CRUSH** 계층 구조에 대한 데이터를 제공하지만 **ceph-ansible**은 **Ansible** 인벤토리 파일을 통해 **CRUSH** 매핑을 통해 해당 데이터를 전달합니다. 기본 루트를 유지하지 않는 경우 각 노드에 대한 루트 위치를 지정해야 합니다.

예를 들어 **lab-ceph01(**프로비저닝 **IP 172.16.0.26)** 노드가 **fast_root** 내의 **rack1** 에 배치된 경우 **Ansible** 인벤토리는 다음과 같아야 합니다.

```
172.16.0.26:
osd_crush_location: {host: lab-ceph01, rack: rack1, root: fast_root}
```

director를 사용하여 **Ceph**를 배포할 때 실제로 **Ansible** 인벤토리를 작성하지 않습니다. 이 인벤토리는 사용자를 위해 생성됩니다. 따라서 **NodeDataLookup** 을 사용하여 데이터를 추가해야 합니다.

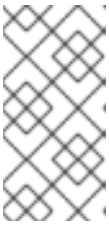
NodeDataLookup은 시스템의 마더보드에 저장된 시스템 제품 **UUID**를 지정하여 작동합니다. **Bare Metal** 서비스**(ironic)**는 인트로스펙션 단계 후에도 이 정보를 저장합니다.

*보조 계층 스토리지를 지원하는 **CRUSH** 맵을 만들려면 다음 단계를 완료합니다.*

절차

1. *다음 명령을 실행하여 **4**개의 노드의 **UUID**를 검색합니다.*

```
for ((x=1; x<=4; x++)); \
{ echo "Node overcloud-ceph0${x}"; \
openstack baremetal introspection data save overcloud-ceph0${x} | jq
.extra.system.product.uuid; }
Node overcloud-ceph01
"32C2BC31-F6BB-49AA-971A-377EFDFDB111"
Node overcloud-ceph02
"76B4C69C-6915-4D30-AFFD-D16DB74F64ED"
Node overcloud-ceph03
"FECF7B20-5984-469F-872C-732E3FEF99BF"
Node overcloud-ceph04
"5FFEFA5F-69E4-4A88-B9EA-62811C61C8B3"
```



참고

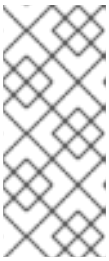
*이 예제에서 **overcloud-ceph0[1-4]**는 **Ironic** 노드 이름입니다. 이 노드 이름은 **lab-ceph0[1-4]** 으로 배포됩니다(**HostnameMap.yaml**을 통해).*

2. *다음과 같이 노드 배치를 지정합니다.*

루트	Rack	노드
standard_root	rack1_std	overcloud-ceph01(lab-ceph01)
	rack2_std	overcloud-ceph02(lab-ceph02)
fast_root	rack1_fast	overcloud-ceph03 (lab-ceph03)
	rack2_fast	overcloud-ceph04(lab-ceph04)

참고

*동일한 이름을 가진 두 개의 버킷을 가질 수 없습니다. **lab-ceph01** 및 **lab-ceph03** 이 동일한 물리적 랙에 있더라도 **rack1** 이라는 두 개의 버킷을 가질 수 없습니다. 따라서 **rack1_std** 및 **rack1_fast** 의 이름을 지정합니다.*



참고

*이 예제에서는 여러 사용자 지정 루트를 설명하기 위해 **"standard_root"**라는 특정 경로를 생성하는 방법을 보여줍니다. 그러나 **HDD**의 **OSD** 노드를 기본 루트에 보관할 수 있습니다.*

3. *다음 **NodeDataLookup** 구문을 사용합니다.*

```
NodeDataLookup: {"SYSTEM_UUID": {"osd_crush_location": {"root": "$MY_ROOT", "rack": "$MY_RACK", "host": "$OVERCLOUD_NODE_HOSTNAME"}}}
```

참고

*시스템 **UUID**를 지정한 다음 **CRUSH** 계층 구조를 위에서 아래로 지정해야 합니다. 또한 **host** 매개변수는 **Bare Metal** 서비스**(ironic)** 노드 이름이 아닌 노드의 오버클라우드 호스트 이름을 가리켜야 합니다. 예제 구성과 일치하려면 다음을 입력합니다.*

```
parameter_defaults:
  NodeDataLookup: {"32C2BC31-F6BB-49AA-971A-377EFDFDB111": {"osd_crush_location": {"root": "standard_root", "rack": "rack1_std", "host": "lab-ceph01"}},
    "76B4C69C-6915-4D30-AFFD-D16DB74F64ED": {"osd_crush_location": {"root": "standard_root", "rack": "rack2_std", "host": "lab-ceph02"}},
    "FECF7B20-5984-469F-872C-732E3FEF99BF": {"osd_crush_location": {"root": "fast_root", "rack": "rack1_fast", "host": "lab-ceph03"}},
    "5FFEFA5F-69E4-4A88-B9EA-62811C61C8B3": {"osd_crush_location": {"root": "fast_root", "rack": "rack2_fast", "host": "lab-ceph04"}}}
```

4. ***ceph-ansible** 수준에서 **CRUSH** 맵 관리를 활성화합니다.*

```
parameter_defaults:
  CephAnsibleExtraConfig:
    create_crush_tree: true
```

5. *스케줄러 힌트를 사용하여 베어 메탈 서비스 노드 **UUID**가 호스트 이름에 올바르게 매핑되도록 합니다.*

```
parameter_defaults:
  CephStorageCount: 4
  OvercloudCephStorageFlavor: ceph-storage
  CephStorageSchedulerHints:
    'capabilities:node': 'ceph-%index%'
```

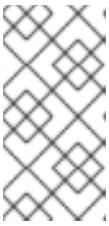
6. *해당 힌트로 베어 메탈 서비스 노드를 태그합니다.*

```
openstack baremetal node set --property capabilities='profile:ceph-storage,node:ceph-0,boot_option:local' overcloud-ceph01

openstack baremetal node set --property capabilities=profile:ceph-storage,'node:ceph-1,boot_option:local' overcloud-ceph02

openstack baremetal node set --property capabilities='profile:ceph-storage,node:ceph-2,boot_option:local' overcloud-ceph03

openstack baremetal node set --property capabilities='profile:ceph-storage,node:ceph-3,boot_option:local' overcloud-ceph04
```



참고

*예측 배치에 대한 자세한 내용은 **Advanced Overcloud Customization** 가이드의 특정 노드 **ID** 할당 을 참조하십시오.*

6.5. CRUSH 맵 규칙 정의

*규칙은 데이터를 클러스터에 작성하는 방법을 정의합니다. **CRUSH** 맵 노드 배치가 완료되면 **CRUSH** 규칙을 정의합니다.*

절차

1. *다음 구문을 사용하여 **CRUSH** 규칙을 정의합니다.*

```
parameter_defaults:
  CephAnsibleExtraConfig:
    crush_rules:
      - name: $RULE_NAME
        root: $ROOT_NAME
        type: $REPLICAT_DOMAIN
        default: true/false
```

참고

default 매개변수를 **true** 로 설정하면 규칙을 지정하지 않고 새 풀을 생성할 때 이 규칙이 사용됩니다. 하나의 기본 규칙만 있을 수 있습니다.

다음 예제에서 규칙 표준 은 랙당 하나의 복제본을 가진 **standard_root** 에서 호스팅되는 **OSD** 노드를 가리킵니다. 규칙은 랙당 하나의 복제본으로 **standard_root** 에서 호스팅되는 **OSD** 노드를 빠르 게 가리킵니다.

```
parameter_defaults:
  CephAnsibleExtraConfig:
    crush_rule_config: true
    crush_rules:
      - name: standard
        root: standard_root
        type: rack
        default: true
      - name: fast
        root: fast_root
        type: rack
        default: false
```

참고

crush_rule_config 를 **true** 로 설정해야 합니다.

6.6. OSP 풀 구성

Ceph 풀은 데이터 저장 방법을 정의하는 **CRUSH** 규칙으로 구성됩니다. 이 예에서는 **standard_root** (표준 규칙)와 **fast_root** (빠른 규칙)를 사용하는 새 풀을 사용하여 기본 제공 **OSP** 풀을 모두 제공합니다.

절차

1. 다음 구문을 사용하여 풀 속성을 정의하거나 변경합니다.

```
- name: $POOL_NAME
    pg_num: $PG_COUNT
    rule_name: $RULE_NAME
    application: rbd
```

2. 모든 **OSP** 풀을 나열하고 적절한 규칙(이 경우 표준)을 설정하고 빠 른 규칙을 사용하는 **tier2** 라는 새 풀을 만듭니다. 이 풀은 **Block Storage(cinder)**에서 사용됩니다.

```
parameter_defaults:
  CephPools:
    - name: tier2
      pg_num: 64
      rule_name: fast
      application: rbd

    - name: volumes
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: vms
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: backups
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: images
      pg_num: 64
      rule_name: standard
      application: rbd

    - name: metrics
      pg_num: 64
      rule_name: standard
      application: openstack_gnocchi
```

6.7. 새 풀을 사용하도록 블록 스토리지 구성

***Ceph** 풀을 **cinder.conf** 파일에 추가하여 **Block Storage(cinder)**를 활성화하여 이를 사용합니다.*

절차

1. *다음과 같이 **cinder.conf** 를 업데이트합니다.*

```
parameter_defaults:
  CinderRbdExtraPools:
    - tier2
```

*6.8. 사용자 정의된 **CRUSH** 맵 확인*

openstack overcloud deploy 명령이 오버클라우드를 생성하거나 업데이트한 후 다음 단계를 완료하여 사용자 지정된 **CRUSH** 맵이 올바르게 적용되었는지 확인합니다.

참고

한 경로에서 다른 경로로 호스트를 이동하는 경우 주의하십시오.

절차

1. **Ceph** 모니터 노드에 연결하고 다음 명령을 실행합니다.

```
# ceph osd tree
ID WEIGHT  TYPE NAME              UP/DOWN REWEIGHT PRIMARY-AFFINITY
-7 0.39996 root standard_root
-6 0.19998     rack rack1_std
-5 0.19998         host lab-ceph02
 1 0.09999             osd.1          up  1.00000          1.00000
 4 0.09999             osd.4          up  1.00000          1.00000
-9 0.19998     rack rack2_std
-8 0.19998         host lab-ceph03
 0 0.09999             osd.0          up  1.00000          1.00000
 3 0.09999             osd.3          up  1.00000          1.00000
-4 0.19998 root fast_root
-3 0.19998     rack rack1_fast
-2 0.19998         host lab-ceph01
 2 0.09999             osd.2          up  1.00000          1.00000
 5 0.09999             osd.5          up  1.00000          1.00000
```

6.9. 다양한 CEPH 풀에 사용자 정의 속성 할당

기본적으로 **director**를 통해 생성된 **Ceph** 풀에는 동일한 배치 그룹(**pg_num** 및 **pgp_num**)과 크기가 있습니다. *5장. Ceph Storage 클러스터 사용자 정의* 의 두 가지 방법을 사용하여 이러한 설정을 전역적으로 재정의할 수 있습니다. 즉, 이렇게 하면 모든 풀에 동일한 값이 적용됩니다.

각 **Ceph** 풀에 서로 다른 속성을 적용할 수도 있습니다. 이를 위해 다음과 같이 **CephPools** 매개변수를 사용합니다.

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

POOL 을 **pg_num** 설정과 함께 구성하려는 풀 이름으로 변경하여 배치 그룹 수를 나타냅니다. 이렇게

*하면 지정된 풀의 기본 **pg_num** 이 재정의됩니다.*

***CephPools** 매개변수를 사용하는 경우 애플리케이션 유형도 지정해야 합니다. 계산, 블록 스토리지 및 이미지 스토리지의 애플리케이션 유형은 예제에 표시된 대로 **rbd** 여야 하지만, 사용할 풀에 따라 다른 애플리케이션 유형을 지정해야 할 수 있습니다. 예를 들어 **gnocchi** 지표 풀의 애플리케이션 유형은 **openstack_gnocchi** 입니다. 자세한 내용은 스토리지 전략 가이드에서 애플리케이션 사용 을 참조하십시오.*

***CephPools** 매개변수를 사용하지 않는 경우 **director**는 기본 풀 목록에 대해서만 적절한 애플리케이션 유형을 자동으로 설정합니다.*

***CephPools** 매개변수를 통해 새 사용자 지정 풀을 생성할 수도 있습니다. 예를 들어 **custompool** 이라는 풀을 추가하려면 다음을 수행합니다.*

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

이렇게 하면 기본 풀 외에도 새 사용자 지정 풀이 생성됩니다.

작은 정보

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

주의

***Red Hat Ceph Storage 3(Luminous)**은 **OSD**에서 보유할 수 있는 최대 **PG** 수에 하드 제한이 있으며 기본적으로 **200**입니다. **200** 이외의 매개 변수를 재정의하지 마십시오. **Ceph PG** 번호가 최대를 초과하므로 문제가 있는 경우 **mon_max_pg_per_osd** 가 아닌 풀당 **pg_num** 을 조정하십시오.*

7장. 오버클라우드 생성

사용자 지정 환경 파일이 준비되면 각 역할이 사용하는 플레이버와 노드를 지정하고 배포를 실행할 수 있습니다.

7.1. 역할에 노드 및 플레이버 할당

오버클라우드 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. Overcloud 노드의 역할 및 플레이버

Heat 템플릿 매개변수	설명
ControllerCount	확장할 컨트롤러 노드 수
OvercloudControlFlavor	컨트롤러 노드(**컨트롤**)에 사용할 플레이버
ComputeCount	확장할 컴퓨팅 노드 수
OvercloudComputeFlavor	컴퓨팅 노드(Compute)에 사용할 플레이버
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, OvercloudCephMdsFlavor 매개변수(**ceph-mon** 및 **ceph-mds** 플레이버와 함께)는 **3장. 전용 노드에 기타 Ceph 서비스 배포** 에 설명된 대로 사용자 지정 **CephMON** 및 **CephMds** 역할을 생성한 경우에만 유효합니다.

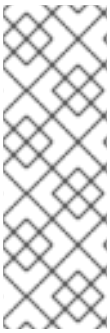
예를 들어 각 역할(**Controller, Compute, Ceph-Storage, CephMon**)에 대해 세 개의 노드를 배포하도록 오버클라우드를 구성하려면 **parameter_defaults** 에 다음을 추가합니다.

```
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

보다 완전한 **Heat** 템플릿 매개변수 목록은 **Director 설치 및 사용** 가이드에서 **CLI 툴로 Overcloud 생성** 을 참조하십시오.

7.2. 오버클라우드 배포 시작



참고

언더클라우드 설치 중에 **undercloud.conf** 파일에 **generate_service_certificate=false** 를 설정합니다. 그러지 않으면 오버클라우드를 배포할 때 신뢰 앵커를 삽입해야 합니다. 신뢰 앵커를 삽입하는 방법에 대한 자세한 내용은 **Advanced Overcloud Customization** 가이드의 **Overcloud Public Endpoints에서 SSL/TLS** 를 참조하십시오.

오버클라우드를 생성하려면 **openstack overcloud deploy** 명령에 대한 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 /usr/share/openstack-tripleo-heat-templates/environments/docker.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/docker-ha.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/docker-network.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(예: **/usr/share/openstack-tripleo-heat-templates/**)에서 **Overcloud**를 생성합니다.
- **-R /home/stack/templates/roles_data_custom.yaml** - 3장. 전용 노드에 기타 Ceph 서비스 배포 에서 사용자 지정 역할 정의 파일을 지정합니다. 그러면 **Ceph MON** 또는 **Ceph MDS** 서비스에 사용자 지정 역할이 추가됩니다. 이러한 역할을 사용하면 두 서비스 중 하나를 전용 노드에 설치할 수 있습니다.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** - **director**가 **Ceph** 클러스터를 생성하도록 설정합니다. 특히 이 환경 파일은 컨테이너화된 **Ceph Storage** 노드를 사용하여 **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** 개체 게이트웨이를 활성화합니다.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml** - 4.1절. “Ceph 메타데이터 서버 활성화” 에 설명된 대로 **Ceph** 메타데이터 서버를 활성화합니다.
- **-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 /usr/share/openstack-tripleo-heat-templates/environments/docker.yaml** - **Red Hat OpenStack Platform**에 대해 **Docker**를 구성합니다.
- **-e /usr/share/openstack-tripleo-heat-templates/environments/docker-ha.yaml** - 컨트롤러 노드의 고가용성 배포를 활성화합니다.

- *-e /usr/share/openstack-tripleo-heat-templates/environments/docker-network.yaml - 네트워크 설정을 구성합니다.*

- *-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 정의 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 템플릿 및 환경 파일 을 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 응답 파일 /home/stack/templates/answers.yaml 에는 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
```

자세한 내용은 Overcloud Creation에서 환경 파일 포함 을 참조하십시오.

전체 옵션 목록은 다음을 실행합니다.

```
$ openstack help overcloud deploy
```

*자세한 내용은 **Director** 설치 및 사용 가이드 의 **CLI** 도구를 사용하여 **Overcloud** 생성 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 다음을 실행합니다.*

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

8장. POST-DEPLOYMENT

다음 하위 섹션에서는 **Ceph** 클러스터를 관리하기 위한 몇 가지 배포 후 작업을 설명합니다.

8.1. 오버클라우드 액세스

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

```
$ source ~/overcloudrc
```

이렇게 하면 **director** 호스트의 **CLI**에서 오버클라우드와 상호 작용하는 데 필요한 환경 변수가 로드됩니다. **director** 호스트와의 상호 작용으로 돌아가려면 다음 명령을 실행합니다.

```
$ source ~/stackrc
```

8.2. 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 docker exec ceph-mon-$HOSTNAME ceph health
```

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

3. *Ceph 모니터 서비스를 실행하는 오버클라우드 노드에 로그인하고 클러스터에서 모든 OSD의 상태를 확인합니다.*

```
sudo docker exec ceph-mon-$HOSTNAME ceph osd tree
```

4. *Ceph Monitor 쿼럼의 상태를 확인합니다.*

```
$ sudo ceph quorum_status
```

이는 쿼럼에 참여하는 모니터와 리더인 모니터를 보여줍니다.

5. *모든 Ceph OSD가 실행 중인지 확인합니다.*

```
$ ceph osd stat
```

Ceph Storage 클러스터 모니터링에 대한 자세 한 내용은 Red Hat Ceph Storage 관리 가이드에서 모니터링 을 참조하십시오.

9장. 환경 재부팅

환경을 재부팅해야 하는 상황이 발생할 수 있습니다. 예를 들어 물리적 서버를 수정해야 하는 경우 또는 정전에서 복구해야 할 수 있습니다. 이 경우 **Ceph Storage** 노드가 올바르게 부팅되도록 해야 합니다.

노드를 다음 순서로 부팅합니다.

- 먼저 모든 **Ceph** 모니터 노드를 부팅 **-** 고가용성 클러스터에서 **Ceph Monitor** 서비스가 활성화됩니다. 기본적으로 **Ceph Monitor** 서비스는 컨트롤러 노드에 설치됩니다. **Ceph Monitor**가 사용자 지정 역할의 컨트롤러와 분리되어 있는 경우 이 사용자 지정 **Ceph Monitor** 역할이 활성화되어 있는지 확인합니다.

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

9.1. CEPH STORAGE(OSD) 클러스터 재부팅

다음 절차에서는 **Ceph Storage(OSD)** 노드 클러스터를 재부팅합니다.

절차

1. **Ceph MON** 또는 컨트롤러 노드에 로그인하고 **Ceph Storage** 클러스터 재조정을 일시적으로 비활성화합니다.

```
$ sudo ceph osd set noout
$ sudo ceph osd set norebalance
```

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

3. 노드를 재부팅합니다.

```
$ sudo reboot
```

4. 노드가 부팅될 때까지 기다립니다.

5. *Ceph MON 또는 컨트롤러 노드에 로그인하고 클러스터 상태를 확인합니다.*

```
$ sudo ceph -s
```

pgmap이 모든 pgs를 정상(active+clean)으로 보고하는지 확인합니다.

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

7. *완료되면 Ceph MON 또는 컨트롤러 노드에 로그인하고 클러스터 재조정을 다시 활성화합니다.*

```
$ sudo ceph osd unset noout
$ sudo ceph osd unset norebalance
```

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

```
$ sudo ceph status
```

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

다음 명령으로 Ceph Storage 노드 클러스터의 HEALTH_OK 상태를 확인합니다.

```
$ sudo ceph status
```

10장. CEPH 클러스터 확장

10.1. CEPH 클러스터 확장

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

이 작업을 수행하기 전에 업데이트된 배포에 사용할 노드가 충분한지 확인합니다. 이러한 노드는 **director**에 등록하고 그에 따라 태그를 지정해야 합니다.

새 Ceph Storage 노드 등록

director에 새 **Ceph** 스토리지 노드를 등록하려면 다음 단계를 따르십시오.

1. **stack** 사용자로 **director** 호스트에 로그인하고 **director** 설정을 초기화합니다.

```
$ source ~/stackrc
```

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

3. 이 파일을 **OpenStack director**로 가져옵니다.

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

노드 정의 템플릿을 가져오면 여기에 정의된 각 노드가 **director**에 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 지정

각 노드를 등록한 후 하드웨어를 검사하고 노드를 특정 프로필에 태그해야 합니다. 프로필 태그는 플레이버에 따라 노드에 일치하며, 그런 다음 플레이버가 배포 역할에 할당됩니다.

새 노드를 검사하고 태그를 지정하려면 다음 단계를 수행합니다.

1. *하드웨어 인트로스펙션을 트리거하여 각 노드의 하드웨어 속성을 검색합니다.*

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

- ***--all-manageable** 옵션은 관리 상태의 노드만 인트로스펙션합니다. 이 예제에서는 모든 것입니다.*
- ***--provide** 옵션은 인트로스펙션 이후 모든 노드를 활성 상태로 재설정합니다.*

중요

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

2. *노드 목록을 검색하여 해당 **UUID**를 확인합니다.*

```
$ openstack baremetal node list
```

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

*예를 들어 다음 명령은 **ceph-storage** 프로필을 사용하여 세 개의 추가 노드를 태그합니다.*

```
$ ironic node-update 551d81f5-4df2-4e0f-93da-6c5de0b868f7 add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update 5e735154-bd6b-42dd-9cc2-b6195c4196d7 add
properties/capabilities='profile:ceph-storage,boot_option:local'
$ ironic node-update 1a2b090c-299d-4c20-a25d-57dd21a7085b add
properties/capabilities='profile:ceph-storage,boot_option:local'
```

작은 정보

방금 태그하고 등록한 노드에서 여러 디스크를 사용하는 경우 각 노드에서 특정 루트 디스크를 사용하도록 director를 설정할 수 있습니다. 이를 수행하는 방법에 대한 지침은 2.5절. “루트 디스크 정의” 을 참조하십시오.

추가 Ceph Storage 노드를 사용하여 오버클라우드 재배포

새 노드를 등록하고 태그를 지정한 후 오버클라우드를 다시 배포하여 Ceph Storage 노드 수를 확장할 수 있습니다. 이 작업을 수행하는 경우 환경 파일의 parameter_defaults (이 경우 ~/templates/storage-config.yaml)에서 CephStorageCount 매개변수를 설정합니다. 7.1절. “역할에 노드 및 플레이버 할당” 에서 오버클라우드는 3개의 Ceph Storage 노드로 배포하도록 구성되어 있습니다. 대신 노드를 최대 6개의 노드로 확장하려면 다음을 사용합니다.

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

이 설정을 사용하여 재배포하면 오버클라우드에 3개가 아닌 6개의 Ceph Storage 노드가 있어야 합니다.

10.2. CEPH STORAGE 노드 축소 및 교체

Ceph 클러스터 축소 또는 Ceph Storage 노드에 결함이 있는 경우와 같이 Ceph Storage 노드를 교체해야 하는 경우가 있습니다. 두 경우 모두 데이터 손실을 방지하기 위해 오버클라우드에서 삭제할 Ceph Storage 노드를 비활성화하고 재조정해야 합니다.

참고

이 절차에서는 Red Hat Ceph Storage 관리 가이드의 단계를 사용하여 Ceph Storage 노드를 수동으로 삭제합니다. Ceph Storage 노드를 수동으로 제거 하는 방법에 대한 자세한 내용은 컨테이너에서 실행되는 Ceph 클러스터 관리 및 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 docker exec ceph-mon-<HOSTNAME> ceph
osd out 0
[heat-admin@overcloud-controller-0 ~]$ sudo docker exec ceph-mon-<HOSTNAME> ceph
osd out 1
```

4. *Ceph Storage 클러스터의 재조정이 시작됩니다. 이 프로세스가 완료될 때까지 기다립니다. 다음 명령을 사용하여 상태를 따릅니다.*

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

5. *Ceph 클러스터의 재조정이 완료되면 삭제 중인 Ceph Storage 노드에 로그인합니다(이 경우 overcloud-cephstorage-0, heat-admin 사용자로, 노드를 중지 및 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. *OSD를 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 CRUSH 맵에서 OSD를 제거하여 더 이상 데이터를 수신하지 못하도록 합니다.*

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

```
osd crush remove osd.1
```

8. ***OSD** 인증 키를 제거합니다.*

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

9. *클러스터에서 **OSD**를 제거합니다.*

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

10. ***CRUSH** 맵에서 스토리지 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 docker 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 항목 목록이 비어 있는지 확인합니다. 목록이 비어 있지 않으면 **7**단계로 이동합니다.*

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

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

12. ***director**가 다시 프로비저닝하지 않도록 **Ceph Storage** 노드를 비활성화합니다.*

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

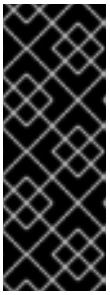
13. ***Ceph Storage** 노드를 삭제하려면 로컬 템플릿 파일을 사용하여 **director**의 오버클라우드 스택을 업데이트해야 합니다. 먼저 오버클라우드 스택의 **UUID**를 확인합니다.*

```
$ openstack stack list
```

14. *삭제하려는 **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 원하지 않는 변경을 수행하지 않도록 여기에서 다시 전달합니다. 자세한 내용은 **Director** 설치 및 사용 가이드 의 **Overcloud** 환경 수정 을 참조하십시오.*

```
$ openstack overcloud node delete /
--stack <stack-name> /
--templates /
-e <other-environment-files> /
<node_UUID>
```

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

17. ***director** 노드 풀에 새 노드를 추가하고 이를 **Ceph Storage** 노드로 배포합니다. 환경 파일의 **parameter_defaults** (이 경우 **~/templates/storage-config.yaml**)에서 **CephStorageCount** 매개변수를 사용하여 오버클라우드의 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 다시 배포합니다.

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

director는 새 노드를 프로비저닝하고 전체 스택을 새 노드의 세부 정보로 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 클러스터의 노드 수와 일치하는지 확인합니다. 제거한 **Ceph Storage** 노드가 새 노드로 교체됩니다.

10.3. CEPH STORAGE 노드에 OSD 추가

다음 절차에서는 노드에 **OSD**를 추가하는 방법을 설명합니다. **Ceph OSD**에 대한 자세한 내용은 **Red Hat Ceph Storage Operations Guide** 의 **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 5.2절. "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 오버클라우드를 업데이트합니다.

참고

이 예에서는 **OSD**가 있는 모든 호스트에 **/dev/sde** 라는 새 장치가 있다고 가정합니다. 모든 노드에 새 장치가 있도록 하지 않도록 하려면 표시된 대로 **heat** 템플릿을 업데이트하고 다른 장치 목록으로 호스트를 정의하는 방법에 대한 정보는 5.2.5절. "디스크 레이아웃을 비Homogeneous Ceph Storage 노드에 매핑" 를 참조하십시오.

10.4. CEPH STORAGE 노드에서 OSD 제거

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

- 서버(**ceph-storage0**)에는 **/dev/sde** 에서 실행되는 **OSD**(**ceph-osd@4**)가 있습니다.
- **Ceph** 모니터 서비스(**ceph-mon**)가 **controller0** 에서 실행되고 있습니다.
- 스토리지 클러스터가 거의 완전한 비율이 아닌지 확인하는 데 사용 가능한 **OSD**가 충분히 있습니다.

Ceph OSD에 대한 자세한 내용은 **Red Hat** Ceph Storage Operations Guide 의 **Ceph OSD** 를 참조

하십시오.

절차

1. *ceph-storage0 에 SSH로 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 SSH로 연결하고 root 로 로그인합니다.*

5. *Ceph 모니터 컨테이너의 이름을 확인합니다.*

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

6. *Ceph 모니터 컨테이너에서 원하지 않는 OSD를 밖 으로 표시하도록 활성화합니다.*

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

참고

이 명령을 사용하면 Ceph가 스토리지 클러스터를 재조정하고 클러스터의 다른 OSD에 데이터를 복사합니다. 클러스터는 리밸런싱이 완료될 때까지 활성 +clean 상태를 일시적으로 유지합니다.

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

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

8. ***CRUSH** 맵에서 **OSD**를 제거하여 더 이상 데이터를 수신하지 않도록 합니다.*

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

9. ***OSD** 인증 키를 제거합니다.*

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

10. ***OSD**를 제거합니다.*

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

11. ***controller0** 에서 연결을 해제합니다.*

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

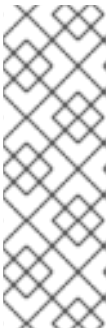
13. ***heat** 템플릿에는 **4**개의 **OSD**가 포함되어 있습니다.*

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

14. *템플릿을 수정하여 **/dev/sde** 를 제거합니다.*

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

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



참고

이 예에서는 ***OSD****가 있는 모든 호스트에서* ***/dev/sde*** *장치를 제거했다고 가정합니다. 모든 노드에서 동일한 장치를 제거하지 않으면 표시된 대로* ***heat*** *템플릿을 업데이트하고 다른 장치 목록으로 호스트를 정의하는 방법에 대한 정보는* ***5.2.5절. "디스크 레이아웃을 비Homogeneous Ceph Storage 노드에 매핑"*** *를 참조하십시오.*

부록 A. 샘플 환경 파일: CEPH 클러스터 생성

다음 사용자 지정 환경 파일은 2장. 오버클라우드 노드 준비 전체에서 설명하는 다양한 옵션을 사용합니다. 이 샘플에는 주석으로 처리된 아웃 옵션이 포함되어 있지 않습니다. 환경 파일에 대한 개요는 환경 파일 (Advanced Overcloud Customization 가이드)을 참조하십시오.

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

```
parameter_defaults: (1)
  CinderBackupBackend: ceph (2)
  CephAnsibleDisksConfig: (3)
    osd_scenario: lvm
    osd_objectstore: bluestore
    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 섹션은 모든 템플릿에서 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)

CephAnsibleDisksConfig 섹션에서는 **BlueStore** 및 **Ceph 3.2** 이상을 사용하여 배포를 위한 사용자 지정 디스크 레이아웃을 정의합니다. **FileStore** 및 **Ceph 3.1** 이하 버전을 사용하는 배포의 경우

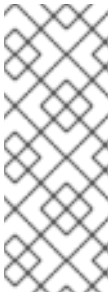
5.2절. "Ceph Storage 노드 디스크 레이아웃 매핑" 에 설명된 예제를 사용하여 CephAnsibleDisksConfig 를 수정합니다.

주의

osd_scenario: lvm 은 ceph-volume 에서 구성한 대로 bluestore 에 대한 새 배포를 기본값으로 사용합니다. ceph-ansible 3.2 이상 및 Ceph Luminous 이상에서만 사용할 수 있습니다. ceph-ansible 3.2로 filestore 를 지원하는 매개변수는 이전 버전과 호환됩니다. 따라서 기존 FileStore 배포에서 osd_objectstore 또는 osd_scenario 매개변수를 변경하지 마십시오.

4

*각 역할에 대해 *Count 매개변수는 여러 노드를 할당하고 Overcloud*Flavor 매개변수는 플레이버를 할당합니다. 예를 들어 ControllerCount: 3 은 컨트롤러 역할에 3개의 노드를 할당하고 OvercloudControlFlavor: control 은 control 플레이버를 사용하도록 각 역할을 설정합니다. 자세한 내용은 7.1절. "역할에 노드 및 플레이버 할당" 을 참조하십시오.*



참고

CephMonCount,CephMdsCount,OvercloudCephMonFlavor, OvercloudCephMdsFlavor 매개변수(ceph-mon 및 ceph-mds 플레이버와 함께)는 3장. 전용 노드에 기타 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 결합된 인터페이스

다음 템플릿은 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 의 사용자 지정 버전입니다. 두 연결 모두에 대한 중복(4.5절. “Ceph 노드당 여러 본딩 인터페이스 구성”설명된 대로 백엔드 및 프런트 엔드 스토리지 네트워크 트래픽을 격리할 수 있는 여러 개의 본딩 인터페이스가 포함되어 있습니다. 또한 사용자 지정 본딩 옵션(예: 4.5.1절. “bonding 모듈 지시문 구성”에 설명된 대로 **'mode=4 lacp_rate=1 ')**을 사용합니다.

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

```
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}
```