Red Hat OpenStack Platform 13

Red Hat OpenStack Platform 업그레이드

Red Hat OpenStack Platform 환경 업그레이드

Last Updated: 2023-05-25

Red Hat OpenStack Platform 13 Red Hat OpenStack Platform 업그레이드

Red Hat OpenStack Platform 환경 업그레이드

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 문서에서는 사용자가 Red Hat OpenStack Platform 12(Pike)에서 13(Queens)으로 업그레이드할 수 있는 다양한 방법을 설명합니다. 이러한 방법은 Red Hat Enterprise Linux 7에 설치된 OpenStack 배포에서 업그레이드할 것으로 가정합니다.

차례

1장. 소개

이 문서에서는 Red Hat OpenStack Platform 환경을 최신 주요 버전으로 업그레이드하고 해당 버전의 마이너 릴리스에서 계속 업데이트할 수 있는 워크플로를 제공합니다.

이 가이드에서는 다음 버전을 통해 업그레이드 경로를 제공합니다.

이전 오버클라우드 버전	새로운 오버클라우드 버전
Red Hat OpenStack Platform 12	Red Hat OpenStack Platform 13

1.1. 높은 수준의 워크플로

다음 표에서는 업그레이드 프로세스에 필요한 단계를 간략하게 설명합니다.

Step	설명
환경 준비	언더클라우드 및 오버클라우드 컨트롤러 노드의 데이터베이스 백업 및 구성을 수행합니다. 최신 마이너 릴리스로 업데이트합니다. 환경을 검증합니다.
언더클라우드 업그레이드	OpenStack Platform 12에서 OpenStack Platform 13으로 언더클라우드를 업그레이드합니다.
컨테이너 이미지 가져오기	OpenStack Platform 13 서비스의 컨테이너 이미지 위치가 포함된 환경 파일을 생성합니다.
오버클라우드 준비	오버클라우드 구성 파일을 OpenStack Platform 13으로 전환하려면 관련 단계를 수행합니다.
컨트롤러 노드 업그레이드	모든 컨트롤러 노드를 OpenStack Platform 13으로 동시에 업그레이드합니다.
컴퓨팅 노드 업그레이드	선택한 컴퓨팅 노드에서 업그레이드를 테스트합니다. 테스트에 성공하면 모든 컴퓨팅 노드를 업그레이드합니다.
Ceph Storage 노드 업그레이드	모든 Ceph Storage 노드를 업그레이드합니다. 여기에는 컨테이너화된 Red Hat Ceph Storage 3 버전으로의 업그레이드가 포함됩니다.
업그레이드 종료	통합 명령을 실행하여 오버클라우드 스택을 새로 고칩니다.

1.2. 리포지토리

언더클라우드와 오버클라우드 모두 Red Hat Content Delivery Network 또는 Red Hat Satellite 6을 통해 Red Hat 리포지토리에 액세스해야 합니다. Red Hat Satellite Server를 사용하는 경우 필요한 리포지토리를 OpenStack Platform 환경과 동기화합니다. 다음 CDN 채널 이름 목록을 가이드로 사용하십시오.

표 1.1. OpenStack Platform 리포지토리

이름	리포지토리	요구 사항 설명
Red Hat Enterprise Linux 7 Server(RPMs)	**rhel-7-server-rpms**	x86_64 시스템용 기본 운영 체제 리포지토리입니다.
Red Hat Enterprise Linux 7 Server - Extras(RPM)	**rhel-7-server-extras-rpms**	Red Hat OpenStack Platform 종속성을 포함합니다.
Red Hat Enterprise Linux 7 Server - RH Common(RPM)	**rhel-7-server-rh-common-rpms**	Red Hat OpenStack Platform 배포 및 구성을 위한 툴을 포함합니다.
Red Hat Satellite Tools 6.3 (for RHEL 7 Server) x86_64	**rhel-7-server-satellite-tools-6.3-rpms**	Red Hat Satellite Server 6으로 호스트를 관리하는 툴입니다. 이후 버전의 Satellite Tools 리포지토리를 사용하면 언더클라우드 설치에 실패할 수 있습니다.
Red Hat Enterprise Linux High Availability(for RHEL 7 Server) (RPM)	**rhel-ha-for-rhel-7-server-rpms**	Red Hat Enterprise Linux용 고가용성 툴입니다. 컨트롤러 노드 고가용성에 사용됩니다.
Red Hat OpenStack Platform 13 for RHEL 7(RPM)	**rhel-7-server-openstack-13-rpms**	코어 Red Hat OpenStack Platform 리포지토리입니다. Red Hat OpenStack Platform director용 패키지도 포함되어 있습니다.
Red Hat Ceph Storage OSD 3 for Red Hat Enterprise Linux 7 Server(RPMs)	**rhel-7-server-rhceph-3-osd-rpms**	(Ceph Storage 노드용) Ceph Storage Object Storage 데몬용 리포지토리입니다. Ceph Storage 노드에 설치됩니다.
Red Hat Ceph Storage MON 3 for Red Hat Enterprise Linux 7 Server(RPMs)	**rhel-7-server-rhceph-3-mon-rpms**	(Ceph Storage 노드용) Ceph Storage 모니터 데몬용 리포지토리입니다. Ceph Storage 노드를 사용하는 OpenStack 환경의 Controller 노드에 설치됩니다.
Red Hat Ceph Storage Tools 3 for Red Hat Enterprise Linux 7 Server(RPMs)	**rhel-7-server-rhceph-3-tools-rpms**	노드가 Ceph Storage 클러스터와 통신할 수 있는 툴을 제공합니다. Ceph Storage 클러스터와 함께 오버클라우드를 배포하거나 오버클라우드를 기존 Ceph Storage 클러스터와 통합할 때 모든 노드에 대해 이 리포지토리를 활성화합니다.

이름	리포지토리	요구 사항 설명
Red Hat OpenStack 13 Director Deployment Tools for RHEL 7(RPM)	**rhel-7-server-openstack-13-deployment-tools-rpms**	(Ceph Storage 노드용) 현재 Red Hat OpenStack Platform director와 호환되는 배포 툴 세트를 제공합니다. 활성 Red Hat OpenStack Platform 서브스크립션이 없는 Ceph 노드에 설치됩니다.
Enterprise Linux for Real Time for NFV (RHEL 7 Server)(RPM)	**rhel-7-server-nfv-rpms**	NFV용 실시간 KVM(RT-KVM) 리포지토리로, 실시간 커널을 활성화하는 패키지가 포함되어 있습니다. 이 리포지토리는 RT-KVM을 대상으로 하는 모든 컴퓨팅 노드에 대해 활성화되어야 합니다. 참고: 이 리포지토리에 액세스하려면 별도의 **Red Hat OpenStack Platform for Real Time** SKU 서브스크립션이 필요합니다.
Red Hat OpenStack Platform 13 Extended Life Cycle Support for RHEL 7(RPM)	**rhel-7-server-openstack-13-els-rpms**	2021년 6월 26일에 시작된 연장된 라이프 사이클 지원에 대한 업데이트가 포함되어 있습니다. 이 리포지토리를 사용하려면 "OpenStack 13 Platform Extended Life Cycle Support"(MCT3637)가 필요합니다.

참고

오프라인 네트워크에서 Red Hat OpenStack Platform 환경에 대한 리포지토리를 구성하려면 Red Hat Customer Portal의 "오프라인 환경에서 Red Hat OpenStack Platform Director 구성" 을 참조하십시오.

1.3. 업그레이드를 시작하기 전

- 업그레이드를 수행하기 전에 하드웨어에 펌웨어 업데이트를 적용합니다.
- 업데이트 중에 OVS(Open vSwitch) 메이저 버전이 변경되는 경우(예: 2.9에서 2.11), director는 사용자 정의 설정 파일의 이름을 .rpmsave 확장자로 변경하고 기본 OVS 구성을 설치합니다. 이전 OVS 사용자 정의를 유지하려면 이름이 변경된 파일에 포함된 수정 사항을 수동으로 다시 적용해야 합니다(예: /etc/logrotate.d/openvswitch의 logrotate 구성). 이 2 단계 업데이트 방법을 사용하면 자동 RPM 패키지 업데이트로 트리거되는 데이터 플레인 중단을 방지할 수 있습니다.

2장. OPENSTACK PLATFORM 업그레이드 준비

이 프로세스를 통해 전체 업데이트를 위해 OpenStack Platform 환경이 준비됩니다. 다음 프로세스에 따라 이 작업을 수행합니다.

- 언더클라우드 패키지를 업데이트하고 업그레이드 명령을 실행합니다.
- 최신 커널 또는 최신 시스템 패키지가 설치된 경우 언더클라우드 재부팅
- overcloud upgrade 명령을 사용하여 오버클라우드 업데이트
- 최신 커널 또는 최신 시스템 패키지가 설치된 경우 오버클라우드 노드를 재부팅합니다.
- 언더클라우드 및 오버클라우드에서 모두 검증을 수행합니다.

이러한 절차를 수행하면 업그레이드를 계속하기 전에 OpenStack Platform 환경이 최상의 상태가 되도록 합니다.

2.1. 현재 언더클라우드 및 오버클라우드의 마이너 업데이트 수행

OpenStack Platform 13으로 업그레이드하기 전에 기존 환경을 최신 마이너 버전(이 경우 12)으로 업데이트해야 합니다. 기존 OpenStack Platform 12 환경을 업데이트하려면 " OpenStack 플랫폼 업데이트" 을 참조하십시오.

2.2. 컨트롤러 및 구성 가능한 노드 재부팅

다음 절차에서는 구성 가능 역할을 기반으로 컨트롤러 노드 및 독립 실행형 노드를 재부팅합니다. 이렇게 하면 Compute 노드와 Ceph Storage 노드가 제외됩니다.

절차

1. 재부팅하려는 노드에 로그인합니다.
2. 선택 사항: 노드에서 Pacemaker 리소스를 사용하는 경우 클러스터를 중지합니다.

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs cluster stop
```

3. 노드를 재부팅합니다.

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

4. 노드가 부팅될 때까지 기다립니다.
5. 서비스를 확인합니다. 예를 들면 다음과 같습니다.
   a. 노드에서 Pacemaker 서비스를 사용하는 경우 노드가 클러스터에 다시 가입했는지 확인합니다.

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

   b. 노드에서 Systemd 서비스를 사용하는 경우 모든 서비스가 활성화되었는지 확인합니다.

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

c. 모든 컨트롤러 및 구성 가능한 노드에 대해 이 단계를 반복합니다.

2.3. CEPH STORAGE(OSD) 클러스터 재부팅

다음 절차에 따라 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 **pgs**를 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_OK**를 보고하는지 확인합니다.

```
$ sudo ceph status
```

2.4. 컴퓨팅 노드 재부팅

컴퓨팅 노드를 재부팅하려면 다음 워크플로우가 필요합니다.

- 새 인스턴스를 프로비저닝하지 않도록 재부팅할 컴퓨팅 노드를 선택하고 비활성화합니다.
- 인스턴스를 다른 컴퓨팅 노드로 마이그레이션하여 인스턴스 다운타임을 최소화합니다.
- 빈 컴퓨팅 노드를 재부팅하고 활성화합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.

2. 재부팅할 컴퓨팅 노드를 식별하려면 모든 컴퓨팅 노드를 나열합니다.

```
$ source ~/stackrc
(undercloud) $ openstack server list --name compute
```

3. 오버클라우드에서 컴퓨팅 노드를 선택하고 비활성화합니다.

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service list
(overcloud) $ openstack compute service set <hostname> nova-compute --disable
```

4. 컴퓨팅 노드에 모든 인스턴스를 나열합니다.

```
(overcloud) $ openstack server list --host <hostname> --all-projects
```

5. 인스턴스를 마이그레이션합니다. 마이그레이션 전략에 대한 자세한 내용은 Compute 노드 간에 가상 머신 마이그레이션을 참조하십시오.

6. Compute 노드에 로그인하고 재부팅합니다.

```
[heat-admin@overcloud-compute-0 ~]$ sudo reboot
```

7. 노드가 부팅될 때까지 기다립니다.

8. 컴퓨팅 노드를 활성화합니다.

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service set <hostname> nova-compute --enable
```

9. 컴퓨팅 노드가 활성화되었는지 확인합니다.

```
(overcloud) $ openstack compute service list
```

2.5. 언더클라우드 검증

다음은 언더클라우드의 기능을 확인하는 일련의 단계입니다.

절차

1. 언더클라우드 액세스 세부 정보를 가져옵니다.

```
$ source ~/stackrc
```

2. 실패한 Systemd 서비스가 있는지 확인합니다.

```
(undercloud) $ sudo systemctl list-units --state=failed 'openstack*' 'neutron*' 'httpd' 'docker'
```

3. 언더클라우드의 사용 가능한 공간을 확인합니다.

```
(undercloud) $ df -h
```

적절한 여유 공간이 있는지 확인하려면 "Undercloud Requirements" 를 기반으로 사용하십시오.

4. 언더클라우드에 NTP가 설치되어 있는 경우 해당 클럭이 동기화되었는지 확인합니다.

```
(undercloud) $ sudo ntpstat
```

5. 언더클라우드 네트워크 서비스를 확인합니다.

```
(undercloud) $ openstack network agent list
```

모든 에이전트가 **Alive** 여야 하며 해당 상태는 **UP** 여야 합니다.

6. 언더클라우드 컴퓨팅 서비스를 확인합니다.

```
(undercloud) $ openstack compute service list
```

모든 에이전트의 상태가 **활성화되어** 있어야 하며 해당 상태는 **up**이어야 합니다.

관련 정보

- 다음 솔루션 문서에서는 OpenStack Orchestration(heat) 데이터베이스에서 삭제된 스택 항목을 제거하는 방법을 보여줍니다. https://access.redhat.com/solutions/2215131

2.6. 컨테이너화된 오버클라우드 검증

다음은 컨테이너화된 오버클라우드 기능을 확인하는 일련의 단계입니다.

절차

1. 언더클라우드 액세스 세부 정보를 가져옵니다.

```
$ source ~/stackrc
```

2. 베어 메탈 노드의 상태를 확인합니다.

```
(undercloud) $ openstack baremetal node list
```

모든 노드에 유효한 전원 상태(**on**)가 있어야 하며 유지 관리 모드는 **false** 여야 합니다.

3. 실패한 Systemd 서비스가 있는지 확인합니다.

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo systemctl list-units --state=failed
'openstack*' 'neutron*' 'httpd' 'docker' 'ceph*'" ; done
```

4. 컨테이너화된 서비스가 실패했는지 확인합니다.

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'exited=1' --all" ; done
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'status=dead' -f
'status=restarting'" ; done
```

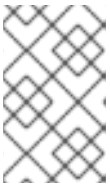
5. 모든 서비스에 대한 HAProxy 연결을 확인합니다. **haproxy.stats** 서비스의 컨트롤 플레인 VIP 주소 및 인증 세부 정보를 가져옵니다.

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE sudo 'grep "listen haproxy.stats" -A 6 /var/lib/config-data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg'
```

다음 cURL 요청에서 다음 세부 정보를 사용하십시오.

```
(undercloud) $ curl -s -u admin:<PASSWORD> "http://<IP ADDRESS>:1993/;csv" | egrep -vi "(frontend|backend)" | cut -d, -f 1,2,18,37,57 | column -s, -t
```

&lt **;PASSWORD** > 및 < **IP ADDRESS** >를 **haproxy.stats** 서비스의 실제 세부 정보로 바꿉니다. 결과 목록은 각 노드의 OpenStack Platform 서비스와 해당 연결 상태를 보여줍니다.



참고

노드가 Redis 서비스를 실행하는 경우 한 노드만 해당 서비스의 **ON** 상태를 표시합니다. 이는 Redis가 active-passive 서비스이므로 한 번에 하나의 노드에서만 실행됩니다.

6. 오버클라우드 데이터베이스 복제 상태를 확인합니다.

```
(undercloud) $ for NODE in $(openstack server list --name controller -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker exec clustercheck clustercheck" ; done
```

7. RabbitMQ 클러스터 상태를 확인합니다.

```
(undercloud) $ for NODE in $(openstack server list --name controller -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker exec $(ssh heat-admin@$NODE "sudo docker ps -f 'name=.*rabbitmq.*' -q") rabbitmqctl node_health_check" ; done
```

8. Pacemaker 리소스 상태를 확인합니다.

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo pcs status"
```

제품 상세 정보:

- 모든 클러스터 노드 **온라인**.
- 클러스터 노드에서 **중지된** 리소스가 없습니다.
- **실패한** pacemaker 작업이 없습니다.

9. 각 오버클라우드 노드의 디스크 공간을 확인합니다.

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo df -h --output=source,fstype,avail -x overlay -x tmpfs -x devtmpfs" ; done
```

10. 오버클라우드 Ceph Storage 클러스터 상태를 확인합니다. 다음 명령은 컨트롤러 노드에서 **ceph** 툴을 실행하여 클러스터를 확인합니다.

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph -s"
```

11. Ceph Storage OSD에서 사용 가능한 공간을 확인합니다. 다음 명령은 컨트롤러 노드에서 **ceph** 툴을 실행하여 사용 가능한 공간을 확인합니다.

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -d= -f2); ssh heat-admin@$NODE "sudo ceph df"
```

12. 오버클라우드 노드에서 클럭이 동기화되었는지 확인합니다.

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo ntpstat" ; done
```

13. source 명령으로 오버클라우드 액세스 세부 정보를 로드합니다.

```
(undercloud) $ source ~/overcloudrc
```

14. 오버클라우드 네트워크 서비스를 확인합니다.

```
(overcloud) $ openstack network agent list
```

모든 에이전트가 **Alive** 여야 하며 해당 상태는 **UP** 여야 합니다.

15. 오버클라우드 컴퓨팅 서비스를 확인합니다.

```
(overcloud) $ openstack compute service list
```

모든 에이전트의 상태가 **활성화되어** 있어야 하며 해당 상태는 **up**이어야 합니다.

16. 오버클라우드 볼륨 서비스를 확인합니다.

```
(overcloud) $ openstack volume service list
```

모든 에이전트의 상태는 **활성화되어야** 하며 해당 상태는 **up** 이어야 합니다.

관련 정보

- "Red Hat 권장 구성으로 OpenStack 환경이 배포되었는지 어떻게 확인할 수 있습니까?" . 이 문서에서는 Red Hat OpenStack Platform 환경을 확인하고 구성을 Red Hat의 권장 사항에 맞게 조정하는 방법에 대한 몇 가지 정보를 제공합니다.

3장. 언더클라우드 업그레이드

이 프로세스는 언더클라우드 및 오버클라우드 이미지를 **Red Hat OpenStack Platform 13**으로 업그레이드합니다.

3.1. 언더클라우드를 OPENSTACK PLATFORM 13으로 업그레이드

이 절차에서는 언더클라우드 툴 세트 및 코어 Heat 템플릿 컬렉션을 **OpenStack Platform 13** 릴리스로 업그레이드합니다.

절차

1. director에 **stack** 사용자로 로그인합니다.
2. 현재 OpenStack Platform 리포지토리를 비활성화합니다.

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-12-rpms
```

3. RHEL 버전을 RHEL 7.9로 설정합니다.

```
$ sudo subscription-manager release --set=7.9
```

4. 새 OpenStack Platform 리포지토리를 활성화합니다.

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
```

5. 오버클라우드 기본 이미지에 대한 업데이트를 다시 활성화합니다.

```
$ sudo yum-config-manager --setopt=exclude= --save
```

6. **yum** 을 실행하여 director의 주요 패키지를 업그레이드합니다.

```
$ sudo yum update -y python-tripleoclient
```

7. 다음 명령을 실행하여 언더클라우드를 업그레이드합니다.

```
$ openstack undercloud upgrade
```

8. 언더클라우드 업그레이드 프로세스가 완료될 때까지 기다립니다.
9. 언더클라우드를 재부팅하여 운영 체제의 커널 및 기타 시스템 패키지를 업데이트합니다.

```
$ sudo reboot
```

10. 노드가 부팅될 때까지 기다립니다.

언더클라우드를 **OpenStack Platform 13** 릴리스로 업그레이드했습니다.

3.2. 오버클라우드 이미지 업그레이드

현재 오버클라우드 이미지를 새 버전으로 교체해야 합니다. 새 이미지를 통해 director가 최신 버전의 OpenStack Platform 소프트웨어를 사용하여 노드를 세부 검사 및 프로비저닝할 수 있습니다.

사전 요구 사항

- 언더클라우드를 최신 버전으로 업그레이드했습니다.

절차

1. 언더클라우드 액세스 세부 정보를 가져옵니다.

```
$ source ~/stackrc
```

2. **stack** 사용자 홈의 **images** 디렉터리에서 기존 이미지를 제거합니다(**/home/stack/images**).

```
$ rm -rf ~/images/*
```

3. 아카이브의 압축을 풉니다.

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-13.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-13.0.tar; do tar -xvf $i; done
$ cd ~
```

4. 최신 이미지를 director로 가져옵니다.

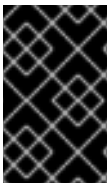
```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

5. 새 이미지를 사용하도록 노드를 설정합니다.

```
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f value)
```

6. 새 이미지가 있는지 확인합니다.

```
$ openstack image list
$ ls -l /httpboot
```



중요

오버클라우드 노드를 배포할 때 오버클라우드 이미지 버전이 해당 heat 템플릿 버전에 해당하는지 확인합니다. 예를 들어 OpenStack Platform 13 heat 템플릿이 있는 OpenStack Platform 13 이미지만 사용합니다.

중요

새 **overcloud-full** 이미지는 이전 **overcloud-full** 이미지를 대체합니다. 이전 이미지를 변경한 경우 특히 향후 새 노드를 배포하려는 경우 새 이미지의 변경 사항을 반복해야 합니다.

3.3. 이전 템플릿 버전 비교

업그레이드 프로세스에서는 최신 오버클라우드 버전에 해당하는 새로운 코어 Heat 템플릿 세트를 설치합니다. Red Hat OpenStack Platform의 리포지토리는 **openstack-tripleo-heat-templates-compat** 패키지에 코어 템플릿 컬렉션의 이전 버전을 유지합니다. 다음 절차에서는 오버클라우드 업그레이드에 영향을 미칠 수 있는 변경 사항을 식별할 수 있도록 이러한 버전을 비교하는 방법을 설명합니다.

절차

1. **openstack-tripleo-heat-templates-compat** 패키지를 설치합니다.

```
$ sudo yum install openstack-tripleo-heat-templates-compat
```

그러면 Heat 템플릿 컬렉션의 **compat** 디렉터리에 이전 템플릿을 설치하고 (**/usr/share/openstack-tripleo-heat-templates/compat**) 이전 버전(**pike**) 다음에 이름이 지정된 링크를 생성합니다. 이러한 템플릿은 업그레이드된 director와 이전 버전과 호환됩니다. 즉, 최신 director 버전을 사용하여 이전 버전의 오버클라우드를 설치할 수 있습니다.

2. 코어 Heat 템플릿의 임시 사본을 생성합니다.

```
$ cp -a /usr/share/openstack-tripleo-heat-templates /tmp/osp13
```

3. 이전 버전을 자체 디렉터리로 이동합니다.

```
$ mv /tmp/osp13/compat /tmp/osp12
```

4. 두 디렉터리의 콘텐츠에 대해 **diff** 를 수행합니다.

```
$ diff -urN /tmp/osp12 /tmp/osp13
```

이는 한 버전에서 다음 버전으로의 코어 템플릿 변경 사항을 보여줍니다. 이러한 변경으로 인해 오버클라우드 업그레이드 중에 어떤 일이 발생하는지 이해할 수 있습니다.

3.4. 언더클라우드 업그레이드 후 노트

- **스택** 사용자 홈 디렉터리에 있는 로컬 코어 템플릿 세트를 사용하는 경우 "사용자 지정 코어 Heat 템플릿 사용"에서 권장 워크플로를 사용하여 템플릿 을 업데이트해야 합니다. 오버클라우드를 업그레이드하기 전에 로컬 사본을 업데이트해야 합니다.

3.5. 다음 단계

언더클라우드 업그레이드가 완료되었습니다. 이제 업그레이드를 위해 오버클라우드를 준비할 수 있습니다.

4장. 컨테이너 이미지 소스 구성

컨테이너화된 오버클라우드는 필요한 컨테이너 이미지가 있는 레지스트리에 액세스해야 합니다. 이 장에서는 Red Hat OpenStack Platform용 컨테이너 이미지를 사용하도록 레지스트리 및 오버클라우드 구성을 준비하는 방법을 설명합니다.

이 가이드에서는 레지스트리를 사용하도록 오버클라우드를 구성하는 몇 가지 사용 사례를 제공합니다. 이러한 사용 사례 중 하나를 시도하기 전에 이미지 준비 명령을 사용하는 방법을 숙지하는 것이 좋습니다. 자세한 내용은 4.2절. "컨테이너 이미지 준비 명령 사용" 를 참조하십시오.

4.1. 레지스트리 방법

Red Hat OpenStack Platform은 다음과 같은 레지스트리 유형을 지원합니다.

원격 레지스트리

오버클라우드는 **registry.redhat.io** 에서 직접 컨테이너 이미지를 가져옵니다. 이 방법은 초기 구성을 생성하는 가장 쉬운 방법입니다. 그러나 각 오버클라우드 노드는 Red Hat Container Catalog에서 직접 각 이미지를 가져오므로 네트워크 혼잡과 느린 배포가 발생할 수 있습니다. 또한 모든 오버클라우드 노드에는 Red Hat Container Catalog에 인터넷 액세스가 필요합니다.

로컬 레지스트리

언더클라우드는 **docker-distribution** 서비스를 사용하여 레지스트리 역할을 합니다. 이를 통해 director는 **registry.redhat.io** 의 이미지를 동기화하고 **docker-distribution** 레지스트리로 푸시할 수 있습니다. 오버클라우드를 생성할 때 오버클라우드는 언더클라우드의 **docker-distribution** 레지스트리에서 컨테이너 이미지를 가져옵니다. 이 방법을 사용하면 내부적으로 레지스트리를 저장할 수 있으므로 배포 속도가 빨라지고 네트워크 정체를 줄일 수 있습니다. 그러나 언더클라우드는 기본 레지스트리 역할을 하며 컨테이너 이미지에 대한 제한된 라이프사이클 관리를 제공합니다.

참고

docker-distribution 서비스는 **docker** 와 별도로 작동합니다. Docker는 **docker-distribution** 레지스트리로 이미지를 가져와서 푸시하는 데 사용되며 오버클라우드에 이미지를 제공하지 않습니다. 오버클라우드는 **docker-distribution** 레지스트리에서 이미지를 가져옵니다.

Satellite Server

컨테이너 이미지의 전체 애플리케이션 라이프사이클을 관리하고 Red Hat Satellite 6 서버를 통해 게시합니다. 오버클라우드는 Satellite 서버에서 이미지를 가져옵니다. 이 방법을 사용하면 Red Hat OpenStack Platform 컨테이너를 저장, 관리 및 배포할 수 있는 엔터프라이즈급 솔루션을 제공합니다.

목록에서 방법을 선택하고 레지스트리 세부 정보를 계속 구성합니다.

참고

다중 아키텍처 클라우드를 빌드하는 경우 로컬 레지스트리 옵션이 지원되지 않습니다.

4.2. 컨테이너 이미지 준비 명령 사용

이 섹션에서는 명령의 다양한 옵션에 대한 개념적 정보를 포함하여 **openstack overcloud container image prepare** 명령을 사용하는 방법에 대한 개요를 제공합니다.

오버클라우드용 컨테이너 이미지 환경 파일 생성

openstack overcloud container image prepare 명령의 주요 용도 중 하나는 오버클라우드에서 사용하

는 이미지 목록이 포함된 환경 파일을 생성하는 것입니다. **openstack overcloud deploy** 와 같은 오버클라우드 배포 명령을 사용하여 이 파일을 포함합니다. **openstack overcloud container image prepare** 명령은 이 함수에 다음 옵션을 사용합니다.

--output-env-file

결과 환경 파일 이름을 정의합니다.

다음 스니펫은 이 파일의 내용의 예입니다.

```
parameter_defaults:
  DockerAodhApiImage: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
  DockerAodhConfigImage: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
...
```

환경 파일에는 언더클라우드 레지스트리의 IP 주소 및 포트로 설정된 **DockerInsecureRegistryAddress** 매개변수도 포함되어 있습니다. 이 매개변수는 오버클라우드 노드에서 SSL/TLS 인증 없이 언더클라우드 레지스트리에서 이미지에 액세스하도록 설정합니다.

가져오기 방법의 컨테이너 이미지 목록 생성

OpenStack Platform 컨테이너 이미지를 다른 레지스트리 소스로 가져오려는 경우 이미지 목록을 생성할 수 있습니다. 목록의 구문은 주로 컨테이너 이미지를 언더클라우드의 컨테이너 레지스트리로 가져오는 데 사용되지만, 이 목록의 형식을 Red Hat Satellite 6과 같은 다른 가져오기 방법에 맞게 수정할 수 있습니다.

openstack overcloud container image prepare 명령은 이 함수에 다음 옵션을 사용합니다.

--output-images-file

가져오기 목록에 대한 결과 파일 이름을 정의합니다.

다음은 이 파일의 내용의 예입니다.

```
container_images:
- imagename: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
- imagename: registry.redhat.io/rhosp13/openstack-aodh-evaluator:13.0-34
...
```

컨테이너 이미지의 네임스페이스 설정

--output-env-file 및 **--output-images-file** 옵션 모두 생성된 이미지 위치를 생성하려면 네임스페이스가 필요합니다. **openstack overcloud container image prepare** 명령은 다음 옵션을 사용하여 가져올 컨테이너 이미지의 소스 위치를 설정합니다.

--namespace

컨테이너 이미지의 네임스페이스를 정의합니다. 일반적으로 디렉터리가 있는 호스트 이름 또는 IP 주소입니다.

--prefix

이미지 이름 앞에 추가할 접두사를 정의합니다.

결과적으로 director는 다음 형식을 사용하여 이미지 이름을 생성합니다.

- **[NAMESPACE]/[PREFIX][IMAGE NAME]**

컨테이너 이미지 태그 설정

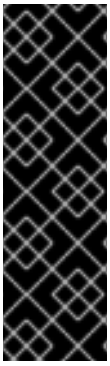
--tag 및 **--tag-from-label** 옵션을 함께 사용하여 각 컨테이너 이미지의 태그를 설정합니다.

--tag

소스의 모든 이미지에 대한 특정 태그를 설정합니다. 이 옵션만 사용하는 경우 director는 이 태그를 사용하여 모든 컨테이너 이미지를 가져옵니다. 그러나 **--tag-from-label** 과 함께 이 옵션을 사용하는 경우 director는 **--tag** 를 소스 이미지로 사용하여 레이블을 기반으로 특정 버전 태그를 식별합니다. **--tag** 옵션은 기본적으로 **latest** 로 설정됩니다.

--tag-from-label

지정된 컨테이너 이미지 레이블의 값을 사용하여 모든 이미지의 버전 지정된 태그를 검색하고 가져옵니다. director는 **--tag** 용으로 설정한 값으로 태그된 각 컨테이너 이미지를 검사한 다음, 컨테이너 이미지 레이블을 사용하여 director가 레지스트리에서 가져온 새 태그를 구성합니다. 예를 들어 **--tag-from-label {version}-{release}** 를 설정하면 director는 **version** 및 **release** 레이블을 사용하여 새 태그를 구성합니다. 한 컨테이너의 경우 **version** 을 **13.0** 으로 설정하고 **release** 를 **34** 로 설정할 수 있으므로 태그 **13.0-34** 가 발생합니다.



중요

Red Hat Container Registry는 특정 버전 형식을 사용하여 모든 Red Hat OpenStack Platform 컨테이너 이미지에 태그를 지정합니다. 이 버전 형식은 **{version}-{release}** 이며, 각 컨테이너 이미지는 컨테이너 메타데이터에 레이블로 저장됩니다. 이 버전 형식은 하나의 **{release}** 에서 다음 버전으로의 업데이트를 원활하게 수행할 수 있습니다. 따라서 **openstack overcloud container image prepare** 명령을 실행할 때 항상 **--tag-from-label {version}-{release}** 를 사용해야 합니다. 컨테이너 이미지를 가져오기 위해 자체적으로 **--tag** 를 사용하지 마십시오. 예를 들어 **--tag latest** 를 사용하면 컨테이너 이미지를 업데이트하기 위해 director에서 태그를 변경해야 하므로 업데이트를 수행할 때 문제가 발생합니다.

4.3. 추가 서비스를 위한 컨테이너 이미지

director는 핵심 OpenStack Platform 서비스에 대한 컨테이너 이미지만 준비합니다. 일부 추가 기능은 추가 컨테이너 이미지가 필요한 서비스를 사용합니다. 환경 파일을 사용하여 이러한 서비스를 활성화합니다. **openstack overcloud container image prepare** 명령은 다음 옵션을 사용하여 환경 파일과 해당 컨테이너 이미지를 포함합니다.

-e

추가 컨테이너 이미지를 활성화하려면 환경 파일을 포함합니다.

다음 표에서는 **/usr/share/openstack-tripleo-heat-templates** 디렉터리 내의 컨테이너 이미지와 해당 환경 파일 위치를 사용하는 추가 서비스의 샘플 목록을 제공합니다.

서비스	환경 파일
Ceph Storage	**environments/ceph-ansible/ceph-ansible.yaml**
collectd	**environments/services-docker/collectd.yaml**
< 매커>	**environments/services-docker/congress.yaml**
fluentd	**environments/services-docker/fluentd.yaml**
OpenStack Bare Metal(ironic)	**environments/services-docker/ironic.yaml**

서비스	환경 파일
OpenStack Data Processing(sahara)	**environments/services-docker/sahara.yaml**
OpenStack EC2-API	**environments/services-docker/ec2-api.yaml**
OpenStack Key Manager(barbican)	**environments/services-docker/barbican.yaml**
OpenStack Load Balancing-as-a-Service(octavia)	**environments/services-docker/octavia.yaml**
OpenStack Shared File System Storage(manila)	**environments/manila-{backend-name}-config.yaml** 참고: 자세한 내용은 OpenStack Shared File System(manila)을 참조하십시오.
OVN(Open Virtual Network)	**environments/services-docker/neutron-ovn-dvr-ha.yaml**
Sensu	**environments/services-docker/sensu-client.yaml**

다음 부분에서는 추가 서비스를 포함하는 예제를 제공합니다.

Ceph Storage

오버클라우드와 함께 Red Hat Ceph Storage 클러스터를 배포하는 경우 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 환경 파일을 포함해야 합니다. 이 파일을 사용하면 오버클라우드에서 구성 가능한 컨테이너화된 서비스를 사용할 수 있으며 director에서 이미지를 준비하려면 이러한 서비스가 활성화되었는지 확인해야 합니다.

이 환경 파일 외에도 OpenStack Platform 서비스와 다른 Ceph Storage 컨테이너 위치도 정의해야 합니다. **--set** 옵션을 사용하여 Ceph Storage와 관련된 다음 매개변수를 설정합니다.

--set ceph_namespace

Ceph Storage 컨테이너 이미지의 네임스페이스를 정의합니다. 이 기능은 **--namespace** 옵션과 유사합니다.

--set ceph_image

Ceph Storage 컨테이너 이미지의 이름을 정의합니다. 일반적으로 **rhceph-3-rhel7** 입니다.

--set ceph_tag

Ceph Storage 컨테이너 이미지에 사용할 태그를 정의합니다. 이 기능은 **--tag** 옵션과 유사합니다. **--tag-from-label** 이 지정되면 이 태그에서 버전 지정된 태그가 검색됩니다.

다음 스니펫은 컨테이너 이미지 파일에 Ceph Storage를 포함하는 예입니다.

```
$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  --set ceph_namespace=registry.redhat.io/rhceph \
```

```
  --set ceph_image=rhceph-3-rhel7 \
  --tag-from-label {version}-{release} \
  ...
```

OpenStack Bare Metal(ironic)

오버클라우드에 OpenStack Bare Metal(ironic)을 배포하는 경우 director가 이미지를 준비하도록 **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml** 환경 파일을 포함해야 합니다. 다음 스니펫은 이 환경 파일을 포함하는 방법에 대한 예입니다.

```
$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml \
  ...
```

OpenStack Data Processing(sahara)

오버클라우드에 OpenStack Data Processing(sahara)을 배포하는 경우 director가 이미지를 준비하도록 **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/sahara.yaml** 환경 파일을 포함해야 합니다. 다음 스니펫은 이 환경 파일을 포함하는 방법에 대한 예입니다.

```
$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/sahara.yaml \
  ...
```

OpenStack Neutron SR-IOV

오버클라우드에 OpenStack Neutron SR-IOV를 배포하는 경우 director가 이미지를 준비하도록 **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml** 환경 파일을 포함합니다. 기본 컨트롤러 및 컴퓨팅 역할은 SR-IOV 서비스를 지원하지 않으므로 SR-IOV 서비스가 포함된 사용자 지정 역할 파일을 포함하려면 **-r** 옵션도 사용해야 합니다. 다음 스니펫은 이 환경 파일을 포함하는 방법에 대한 예입니다.

```
$ openstack overcloud container image prepare \
  ...
  -r ~/custom_roles_data.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml \
  ...
```

OpenStack Load Balancing-as-a-Service(octavia)

오버클라우드에 OpenStack Load Balancing-as-a-Service를 배포하는 경우 director가 이미지를 준비하도록 **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml** 환경 파일을 포함합니다. 다음 스니펫은 이 환경 파일을 포함하는 방법에 대한 예입니다.

```
$ openstack overcloud container image prepare \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml
\
  ...
```

OpenStack Shared File System(manila)

manila-{backend-name}-config.yaml 형식을 사용하여 지원되는 백엔드를 선택하여 해당 백엔드와 함께 공유 파일 시스템을 배포할 수 있습니다. 공유 파일 시스템 서비스 컨테이너는 다음 환경 파일을 포함하여 준비할 수 있습니다.

```
environments/manila-isilon-config.yaml
environments/manila-netapp-config.yaml
environments/manila-vmax-config.yaml
environments/manila-cephfsnative-config.yaml
environments/manila-cephfsganesha-config.yaml
environments/manila-unity-config.yaml
environments/manila-vnx-config.yaml
```

환경 파일 사용자 지정 및 배포에 대한 자세한 내용은 다음 리소스를 참조하십시오.

- *공유 파일 시스템 서비스의 NFS 백엔드 가이드를 통해 CephFS에 업데이트된 환경 배포*
- *공유 파일 시스템 서비스 용 NetApp 백엔드에서 NetApp 백엔드 가이드를 사용하여 공유 파일 시스템 서비스배포*
- *공유 파일 시스템 서비스 용 CephFS 백엔드에서 CephFS 백엔드를 사용하여 공유 파일 시스템 서비스배포*

4.4. RED HAT 레지스트리를 원격 레지스트리 소스로 사용

Red Hat은 **registry.redhat.io** 에서 오버클라우드 컨테이너 이미지를 호스팅합니다. 원격 레지스트리에서 이미지를 가져오는 것이 레지스트리가 이미 구성되어 있고 필요한 모든 것은 가져온 이미지의 URL과 네임스페이스이기 때문에 가장 간단한 방법입니다. 그러나 오버클라우드 생성 중에 오버클라우드 노드는 모두 원격 리포지토리에서 이미지를 가져옵니다. 이 이미지는 외부 연결을 유지할 수 있습니다. 따라서 이 방법은 프로덕션 환경에 권장되지 않습니다. 프로덕션 환경의 경우 다음 방법 중 하나를 사용합니다.

- 로컬 레지스트리 설정
- Red Hat Satellite 6에서 이미지 호스트

절차

1. 오버클라우드 배포의 **registry.redhat.io** 에서 직접 이미지를 가져오려면 이미지 매개변수를 지정하는 환경 파일이 필요합니다. 다음 명령을 실행하여 컨테이너 이미지 환경 파일을 생성합니다.

```
(undercloud) $ sudo openstack overcloud container image prepare \
  --namespace=registry.redhat.io/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

   - **-e** 옵션을 사용하여 선택적 서비스에 대한 환경 파일을 포함합니다.
   - 사용자 지정 역할 파일을 포함하려면 **-r** 옵션을 사용합니다.
   - Ceph Storage를 사용하는 경우 추가 매개 변수를 포함하여 Ceph Storage 컨테이너 이미지 위치(**--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag**)를 정의합니다.

2. **overcloud_images.yaml** 파일을 수정하고 다음 매개변수를 포함하여 배포 중에 **registry.redhat.io** 로 인증합니다.

```
ContainerImageRegistryLogin: true
ContainerImageRegistryCredentials:
  registry.redhat.io:
    <USERNAME>: <PASSWORD>
```

- &lt **;USERNAME** > 및 < **PASSWORD** >를 **registry.redhat.io** 의 인증 정보로 바꿉니다. **overcloud_images.yaml** 파일에는 언더클라우드의 이미지 위치가 포함되어 있습니다. 배포를 통해 이 파일을 포함합니다.

참고

openstack overcloud deploy 명령을 실행하기 전에 원격 레지스트리에 로그인해야 합니다.

```
(undercloud) $ sudo docker login registry.redhat.io
```

레지스트리 구성이 준비되었습니다.

4.5. 언더클라우드를 로컬 레지스트리로 사용

오버클라우드 컨테이너 이미지를 저장하도록 언더클라우드에서 로컬 레지스트리를 구성할 수 있습니다.

director를 사용하여 **registry.redhat.io** 에서 각 이미지를 가져온 후 언더클라우드에서 실행되는 **docker-distribution** 레지스트리로 각 이미지를 푸시할 수 있습니다. director를 사용하여 오버클라우드를 생성할 때 오버클라우드 생성 프로세스 중에 노드는 언더클라우드 **docker-distribution** 레지스트리에서 관련 이미지를 가져옵니다.

이렇게 하면 컨테이너 이미지의 네트워크 트래픽이 내부 네트워크 내에 유지되므로 외부 네트워크 연결을 가장하지 않고 배포 프로세스를 가속화할 수 있습니다.

절차

1. 로컬 언더클라우드 레지스트리의 주소를 찾습니다. 주소는 다음 패턴을 사용합니다.

```
<REGISTRY_IP_ADDRESS>:8787
```

이전에는 **undercloud.conf** 파일에서 **local_ip** 매개변수로 설정한 언더클라우드의 IP 주소를 사용합니다. 아래 명령의 경우 주소가 **192.168.24.1:8787** 로 가정합니다.

2. **registry.redhat.io** 에 로그인합니다.

```
(undercloud) $ docker login registry.redhat.io --username $RH_USER --password
$RH_PASSWD
```

3. 이미지를 로컬 레지스트리에 업로드하는 템플릿을 생성하고 해당 이미지를 참조하는 환경 파일을 생성합니다.

```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=registry.redhat.io/rhosp13 \
  --push-destination=192.168.24.1:8787 \
  --prefix=openstack- \
```

```
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml \
  --output-images-file /home/stack/local_registry_images.yaml
```

- **-e** 옵션을 사용하여 선택적 서비스에 대한 환경 파일을 포함합니다.
- 사용자 지정 역할 파일을 포함하려면 **-r** 옵션을 사용합니다.
- Ceph Storage를 사용하는 경우 추가 매개 변수를 포함하여 Ceph Storage 컨테이너 이미지 위치(**--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag**)를 정의합니다.

4. 다음 두 파일이 생성되었는지 확인합니다.
   - **local_registry_images.yaml** - 원격 소스의 컨테이너 이미지 정보가 포함됩니다. 이 파일을 사용하여 Red Hat Container Registry(**registry.redhat.io**)에서 언더클라우드로 이미지를 가져옵니다.
   - **overcloud_images.yaml**: 언더클라우드에 이벤트 이미지 위치가 포함됩니다. 이 파일을 배치와 함께 포함합니다.
5. 원격 레지스트리에서 컨테이너 이미지를 가져와서 언더클라우드 레지스트리로 푸시합니다.

```
(undercloud) $ openstack overcloud container image upload \
 --config-file  /home/stack/local_registry_images.yaml \
 --verbose
```

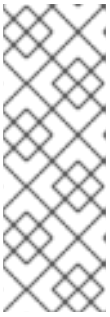
네트워크 및 언더클라우드 디스크에 따라 필요한 이미지를 가져오는 데 시간이 걸릴 수 있습니다.

참고

컨테이너 이미지는 약 10GB의 디스크 공간을 사용합니다.

6. 이제 이미지가 언더클라우드의 **docker-distribution** 레지스트리에 저장됩니다. 언더클라우드의 **docker-distribution** 레지스트리에서 이미지 목록을 보려면 다음 명령을 실행합니다.

```
(undercloud) $  curl http://192.168.24.1:8787/v2/_catalog | jq .repositories[]
```



참고

_catalog 리소스는 자체적으로 100개의 이미지만 표시합니다. 더 많은 이미지를 표시하려면 **_catalog** 리소스와 함께 **?n=<interger** > 쿼리 문자열을 사용하여 더 많은 수의 이미지를 표시합니다.

```
(undercloud) $  curl http://192.168.24.1:8787/v2/_catalog?n=150 | jq
.repositories[]
```

특정 이미지의 태그 목록을 보려면 **skopeo** 명령을 사용합니다.

```
(undercloud) $ curl -s http://192.168.24.1:8787/v2/rhosp13/openstack-keystone/tags/list | jq
.tags
```

태그된 이미지를 확인하려면 **skopeo** 명령을 사용합니다.

```
(undercloud) $ skopeo inspect --tls-verify=false
docker://192.168.24.1:8787/rhosp13/openstack-keystone:13.0-44
```

레지스트리 구성이 준비되었습니다.

4.6. SATELLITE 서버를 레지스트리로 사용

Red Hat Satellite 6는 레지스트리 동기화 기능을 제공합니다. 이를 통해 여러 이미지를 Satellite 서버로 가져와 애플리케이션 라이프사이클의 일부로 관리할 수 있습니다. Satellite는 다른 컨테이너 활성화 시스템이 사용할 레지스트리 역할도 합니다. 컨테이너 이미지 관리 방법에 대한 자세한 내용은 *Red Hat Satellite 6 Content Management Guide*의 "Managing Container Images"를 참조하십시오.

다음 절차의 예제에서는 Red Hat Satellite 6용 **hammer** 명령행 툴과 **ACME**라는 조직을 사용합니다. 이 조직을 실제로 사용하는 Satellite 6 조직으로 대체하십시오.

절차

1. 이미지를 로컬 레지스트리로 가져올 템플릿을 생성합니다.

```
$ source ~/stackrc
(undercloud) $ openstack overcloud container image prepare \
  --namespace=rhosp13 \
  --prefix=openstack- \
  --output-images-file /home/stack/satellite_images
```

   - **-e** 옵션을 사용하여 선택적 서비스에 대한 환경 파일을 포함합니다.
   - 사용자 지정 역할 파일을 포함하려면 **-r** 옵션을 사용합니다.
   - Ceph Storage를 사용하는 경우 추가 매개 변수를 포함하여 Ceph Storage 컨테이너 이미지 위치(**--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag**)를 정의합니다.

참고

openstack overcloud container image prepare 명령의 이 버전은 **registry.redhat.io** 의 레지스트리를 대상으로 이미지 목록을 생성합니다. 이후 단계에서 사용한 **openstack overcloud container image prepare** 명령과 다른 값을 사용합니다.

2. 이렇게 하면 컨테이너 이미지 정보가 있는 **satellite_images** 라는 파일이 생성됩니다. 이 파일을 사용하여 컨테이너 이미지를 Satellite 6 서버에 동기화합니다.

3. **satellite_images** 파일에서 YAML 관련 정보를 제거하고 이미지 목록만 포함된 플랫 파일로 변환합니다. 다음 **sed** 명령은 다음을 수행합니다.

```
(undercloud) $ awk -F ':' '{if (NR!=1) {gsub("[[:space:]]", ""); print $2}}' ~/satellite_images >
~/satellite_images_names
```

   이는 Satellite 서버로 가져온 이미지 목록을 제공합니다.

4. **satellite_images_names** 파일을 Satellite 6 **hammer** 툴이 포함된 시스템으로 복사합니다. 또는 *Hammer CLI 가이드*의 지침을 사용하여 **hammer** 툴을 언더클라우드에 설치합니다.

5. 다음 **hammer** 명령을 실행하여 Satellite 조직에 새 제품(**OSP13 Containers**)을 생성합니다.

```
$ hammer product create \
  --organization "ACME" \
  --name "OSP13 Containers"
```

이 사용자 지정 제품에 이미지를 저장합니다.

6. 제품에 기본 컨테이너 이미지를 추가합니다.

```
$ hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
  --url https://registry.redhat.io \
  --docker-upstream-name rhosp13/openstack-base \
  --name base
```

7. **satellite_images** 파일에서 오버클라우드 컨테이너 이미지를 추가합니다.

```
$ while read IMAGE; do \
  IMAGENAME=$(echo $IMAGE | cut -d"/" -f2 | sed "s/openstack-//g" | sed "s/:.*//g") ; \
  hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
  --url https://registry.redhat.io \
  --docker-upstream-name $IMAGE \
  --name $IMAGENAME ; done < satellite_images_names
```

8. 컨테이너 이미지를 동기화합니다.

```
$ hammer product synchronize \
  --organization "ACME" \
  --name "OSP13 Containers"
```

Satellite 서버가 동기화를 완료할 때까지 기다립니다.

참고

설정에 따라 **hammer**에서 Satellite 서버 사용자 이름과 암호가 필요할 수 있습니다. **hammer**를 구성한 후 구성 파일을 사용하여 자동으로 로그인할 수 있습니다. *Hammer CLI Guide* 의 "Authentication" 섹션을 참조하십시오.

9. Satellite 6 서버에서 콘텐츠 뷰를 사용하는 경우 새 콘텐츠 뷰 버전을 생성하여 이미지를 통합합니다.

10. **base** 이미지에 사용 가능한 태그를 확인합니다.

```
$ hammer docker tag list --repository "base" \
  --organization "ACME" \
  --product "OSP13 Containers"
```

그러면 OpenStack Platform 컨테이너 이미지에 대한 태그가 표시됩니다.

11. 언더클라우드로 돌아가서 Satellite 서버의 이미지에 대한 환경 파일을 생성합니다. 다음은 환경 파일을 생성하는 예제 명령입니다.

```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=satellite6.example.com:5000 \
  --prefix=acme-osp13_containers- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

참고

openstack overcloud container image prepare 명령의 이 버전은 Satellite 서버를 대상으로 합니다. 이전 단계에서 사용한 **openstack overcloud container image prepare** 명령과 다른 값을 사용합니다.

이 명령을 실행할 때 다음 데이터를 포함합니다.

- **--namespace** - Satellite 서버에 있는 레지스트리의 URL 및 포트입니다. Red Hat Satellite의 레지스트리 포트는 5000입니다. 예를 들면 **--namespace=satellite6.example.com:5000** 입니다.

참고

Red Hat Satellite 버전 6.10을 사용하는 경우 포트를 지정할 필요가 없습니다. 기본 포트 **443** 이 사용됩니다. 자세한 내용은 " RHOSP13 배포를 Red Hat Satellite 6.10에 어떻게 적용할 수 있습니까?"에서 참조하십시오.

- **--prefix=** - 접두사는 소문자를 사용하고 밑줄에 공백을 대체하는 라벨의 Satellite 6 규칙을 기반으로 합니다. 접두사는 콘텐츠 뷰 사용 여부에 따라 달라집니다.
  - 콘텐츠 뷰를 사용하는 경우 구조는 **[org]-[environment]-[content view]-[product]-**입니다. 예: **acme-production-myosp13-osp13_containers-**.
  - 콘텐츠 뷰를 사용하지 않는 경우 구조는 **[org]-[product]-**입니다. 예: **acme-osp13_containers-**.
- **--tag-from-label {version}-{release}** - 각 이미지의 최신 태그를 식별합니다.
- **-e** - 선택적 서비스를 위한 환경 파일을 포함합니다.
- **-** R - 사용자 지정 역할 파일을 포함합니다.
- **--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag** - Ceph Storage를 사용하는 경우 추가 매개변수를 포함하여 Ceph Storage 컨테이너 이미지 위치를 정의합니다. 이제 **ceph_image**에는 Satellite별 접두사가 포함됩니다. 이 접두사는 **--prefix** 옵션과 동일한 값입니다. 예를 들어 다음과 같습니다.

```
--set ceph_image=acme-osp13_containers-rhceph-3-rhel7
```

이렇게 하면 오버클라우드에서 Satellite 이름 지정 규칙을 사용하여 Ceph 컨테이너 이미지를 사용합니다.

12. **overcloud_images.yaml** 파일에는 Satellite 서버의 이미지 위치가 포함되어 있습니다. 배포를 통해 이 파일을 포함합니다.

레지스트리 구성이 준비되었습니다.

4.7. 다음 단계

이제 컨테이너 이미지 소스 목록이 포함된 **overcloud_images.yaml** 환경 파일이 있습니다. 향후 모든 업그레이드 및 배포 작업에 이 파일을 포함합니다.

이제 업그레이드를 위해 오버클라우드를 준비할 수 있습니다.

5장. 오버클라우드 업그레이드 준비

이 프로세스에서는 업그레이드 프로세스를 위해 오버클라우드를 준비합니다.

사전 요구 사항

- 언더클라우드를 최신 버전으로 업그레이드했습니다.

5.1. 오버클라우드 등록 세부 정보 준비

오버클라우드가 업그레이드 프로세스 중에 최신 패키지를 사용하도록 오버클라우드에 최신 서브스크립션 세부 정보를 제공해야 합니다.

사전 요구 사항

- 최신 OpenStack Platform 리포지토리가 포함된 서브스크립션입니다.
- 등록에 활성화 키를 사용하는 경우 새 OpenStack Platform 리포지토리를 포함하여 새 활성화 키를 생성합니다.

절차

1. 등록 세부 정보가 포함된 환경 파일을 편집합니다. 예를 들어 다음과 같습니다.

```
$ vi ~/templates/rhel-registration/environment-rhel-registration.yaml
```

2. 다음 매개변수 값을 편집합니다.

rhel_reg_repos

Red Hat OpenStack Platform 13의 새 리포지토리를 포함하도록 업데이트되었습니다.

rhel_reg_activation_key

활성화 키를 업데이트하여 Red Hat OpenStack Platform 13 리포지토리에 액세스합니다.

rhel_reg_sat_repo

최신 버전의 Red Hat Satellite 6을 사용하는 경우 Satellite 6의 관리 툴을 포함하는 리포지토리를 업데이트합니다.

3. 환경 파일을 저장합니다.

관련 정보

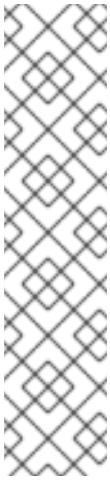
- 등록 매개변수에 대한 자세한 내용은 *Advanced Overcloud Customization 가이드의 "환경 파일을 사용하여 Overcloud 등록"*을 참조하십시오.

5.2. 더 이상 사용되지 않는 매개변수

다음 매개 변수는 더 이상 사용되지 않으며 교체되어 있습니다.

이전 매개 변수	새로운 매개변수
KeystoneNotificationDriver	**NotificationDriver**

이전 매개 변수	새로운 매개변수
controllerExtraConfig	**ControllerExtraConfig**
OvercloudControlFlavor	**OvercloudControllerFlavor**
controllerImage	**ControllerImage**
NovaImage	**ComputeImage**
NovaComputeExtraConfig	**ComputeExtraConfig**
NovaComputeServerMetadata	**ComputeServerMetadata**

NovaComputeSchedulerHints	**ComputeSchedulerHints**
	 
	참고
	역할별 Compute(컴퓨팅)SchedulerHints를 사용하려면 더 이상 사용되지 않는 **NovaCompute SchedulerHints** 매개변수가 설정되었지만 정의되지 않은 경우 환경에 다음 구성을 추가해야 합니다.
	`parameter_defaults:`
	`  NovaComputeSchedulerHints: {}`
	사용자 정의 역할을 사용할 때 역할별 **_ROLE_SchedulerHints** 매개변수를 사용하려면 이 구성을 추가해야 합니다.

NovaComputeIPs	**ComputeIPs**
SwiftStorageServerMetadata	**ObjectStorageServerMetadata**
SwiftStorageIPs	**ObjectStorageIPs**
SwiftStorageImage	**ObjectStorageImage**
OvercloudSwiftStorageFlavor	**OvercloudObjectStorageFlavor**
NeutronDpdkCoreList	**OvsPmdCoreList**
NeutronDpdkMemoryChannels	**OvsDpdkMemoryChannels**
NeutronDpdkSocketMemory	**OvsDpdkSocketMemory**
NeutronDpdkDriverType	**OvsDpdkDriverType**

이전 매개 변수	새로운 매개변수
HostCpusList	**OvsDpdkCoreList**

새 매개변수 값의 경우 다음 예와 같이 중첩된 작은따옴표가 없는 큰따옴표를 사용합니다.

이전 매개 변수 값	값이 있는 새로운 매개 변수
NeutronDpdkCoreList: "'2,3'"	**OvsPmdCoreList: "2,3"**
HostCpusList: "'0,1'"	**OvsDpdkCoreList: "0,1"**

사용자 지정 환경 파일에서 이러한 매개변수를 업데이트합니다. 다음 매개변수는 현재 동등하지 않고 더 이상 사용되지 않습니다.

NeutronL3HA

L3 고가용성은 분산 가상 라우팅(**NeutronEnableDVR**)이 있는 구성을 제외한 모든 경우에 활성화됩니다.

CeilometerWorkers

Ceilometer는 최신 구성 요소(Gnocchi, Aodh, Panko)를 사용하는 데 더 이상 사용되지 않습니다.

CinderNetappEseriesHostType

모든 E-series 지원이 더 이상 사용되지 않습니다.

ControllerEnableSwiftStorage

ControllerServices 매개변수 조작을 대신 사용해야 합니다.

OpenDaylightPort

EndpointMap을 사용하여 OpenDaylight의 기본 포트를 정의합니다.

OpenDaylightConnectionProtocol

이제 이 매개변수의 값은 TLS를 사용하여 Overcloud를 배포 중인지 여부에 따라 결정됩니다.

/home/stack 디렉토리에서 다음 **egrep** 명령을 실행하여 더 이상 사용되지 않는 매개변수가 포함된 환경 파일을 식별합니다.

```
$ egrep -r -w
'KeystoneNotificationDriver|controllerExtraConfig|OvercloudControlFlavor|controllerImage|NovaImage|NovaComputeExtraConfig|NovaComputeServerMetadata|NovaComputeSchedulerHints|NovaComputeIPs|SwiftStorageServerMetadata|SwiftStorageIPs|SwiftStorageImage|OvercloudSwiftStorageFlavor|NeutronDpdkCoreList|NeutronDpdkMemoryChannels|NeutronDpdkSocketMemory|NeutronDpdkDriverType|HostCpusList|NeutronDpdkCoreList|HostCpusList|NeutronL3HA|CeilometerWorkers|CinderNetappEseriesHostType|ControllerEnableSwiftStorage|OpenDaylightPort|OpenDaylightConnectionProtocol' *
```

OpenStack Platform 환경에 이러한 더 이상 사용되지 않는 매개변수가 여전히 필요한 경우 기본 **roles_data** 파일에서 이 매개변수를 사용할 수 있습니다. 그러나 사용자 정의 **roles_data** 파일을 사용하고 있으며 오버클라우드에 이러한 더 이상 사용되지 않는 매개변수가 필요한 경우 **roles_data** 파일을 편집하고 각 역할에 다음을 추가하여 해당 매개변수를 액세스할 수 있습니다.

컨트롤러 역할

```
- name: Controller
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  ...
```

컴퓨팅 역할

```
- name: Compute
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  disable_upgrade_deployment: True
  ...
```

오브젝트 스토리지 역할

```
- name: ObjectStorage
  uses_deprecated_params: True
  deprecated_param_metadata: 'SwiftStorageServerMetadata'
  deprecated_param_ips: 'SwiftStorageIPs'
  deprecated_param_image: 'SwiftStorageImage'
  deprecated_param_flavor: 'OvercloudSwiftStorageFlavor'
  disable_upgrade_deployment: True
  ...
```

5.3. 더 이상 사용되지 않는 CLI 옵션

일부 명령행 옵션은 오래되었거나 더 이상 사용되지 않으며, 대신 환경 파일의 **parameter_defaults** 섹션에 포함하는 Heat 템플릿 매개변수가 사용됩니다. 다음 표에는 더 이상 사용되지 않는 옵션이 해당하는 Heat 템플릿 옵션에 매핑되어 있습니다.

표 5.1. 더 이상 사용되지 않는 CLI 옵션을 Heat 템플릿 매개변수에 매핑

옵션	설명	Heat 템플릿 매개변수
--control-scale	확장할 컨트롤러 노드 수	**ControllerCount**
--compute-scale	확장할 컴퓨팅 노드 수	**ComputeCount**
--ceph-storage-scale	확장할 Ceph Storage 노드 수	**CephStorageCount**
--block-storage-scale	확장할 Cinder 노드 수	**BlockStorageCount**
--swift-storage-scale	확장할 Swift 노드 수	**ObjectStorageCount**

옵션	설명	Heat 템플릿 매개변수
--control-flavor	컨트롤러 노드에 사용할 플레이버	**OvercloudControllerFlavor**
--compute-flavor	컴퓨팅 노드에 사용할 플레이버	**OvercloudComputeFlavor**
--ceph-storage-flavor	Ceph Storage 노드에 사용할 플레이버	**OvercloudCephStorageFlavor**
--block-storage-flavor	Cinder 노드에 사용할 플레이버	**OvercloudBlockStorageFlavor**
--swift-storage-flavor	Swift Storage 노드에 사용할 플레이버	**OvercloudSwiftStorageFlavor**
--neutron-flat-networks	neutron 플러그인에 구성할 플랫 네트워크를 정의합니다. 외부 네트워크 생성을 허용하도록 기본적으로 "datacentre"로 설정됩니다.	**NeutronFlatNetworks**
--neutron-physical-bridge	각 하이퍼바이저에 생성할 Open vSwitch 브리지입니다. 기본값은 "br-ex"입니다. 일반적으로 이 값은 변경할 필요가 없습니다.	**HypervisorNeutronPhysicalBridge**
--neutron-bridge-mappings	사용할 논리적 브리지와 물리적 브리지 매핑입니다. 기본적으로 호스트(br-ex)의 외부 브릿지를 물리적 이름(datacentre)에 매핑합니다. 기본 유동 네트워크에 이 값을 사용합니다.	**NeutronBridgeMappings**
--neutron-public-interface	네트워크 노드의 br-ex에 브리지에 인터페이스를 정의합니다.	**NeutronPublicInterface**
--neutron-network-type	Neutron의 테넌트 네트워크 유형	**NeutronNetworkType**
--neutron-tunnel-types	Neutron 테넌트 네트워크의 터널 유형입니다. 여러 값을 지정하려면 콤마로 구분된 문자열을 사용합니다.	**NeutronTunnelTypes**
--neutron-tunnel-id-ranges	테넌트 네트워크 할당에 사용할 수 있는 GRE 터널 ID 범위	**NeutronTunnelIdRanges**
--neutron-vni-ranges	테넌트 네트워크 할당에 사용할 수 있는 VXLAN VNI ID 범위	**NeutronVniRanges**

옵션	설명	Heat 템플릿 매개변수
--neutron-network-vlan-ranges	지원할 Neutron ML2 및 Open vSwitch VLAN 매핑 범위입니다. 기본적으로 'datacentre' 물리적 네트워크에서 VLAN을 허용하도록 설정되어 있습니다.	**NeutronNetworkVLANRanges**
--neutron-mechanism-drivers	neutron 테넌트 네트워크의 메커니즘 드라이버입니다. 기본값은 "openvswitch"입니다. 여러 값을 지정하려면 콤마로 구분된 문자열을 사용합니다.	**NeutronMechanismDrivers**
--neutron-disable-tunneling	VLAN 세그먼트화된 네트워크 또는 플랫 네트워크를 Neutron과 함께 사용하려는 경우 터널링을 비활성화합니다.	매개 변수 매핑 없음
--validation-errors-fatal	오버클라우드 생성 프로세스에서는 일련의 사전 배포 검사를 수행합니다. 이 옵션은 사전 배포 확인에서 치명적인 오류가 발생할 경우에만 종료됩니다. 오류가 있으면 배포에 실패할 수 있으므로 이 옵션을 사용하는 것이 좋습니다.	매개변수 매핑 없음
--ntp-server	시간 동기화에 사용할 NTP 서버 설정	**NtpServer**

이러한 매개변수는 Red Hat OpenStack Platform에서 제거되었습니다. CLI 옵션을 Heat 매개변수로 변환하여 환경 파일에 추가하는 것이 좋습니다.

5.4. 구성 가능 네트워크

이 Red Hat OpenStack Platform 버전에는 구성 가능한 네트워크용 새로운 기능이 도입되었습니다. 사용자 지정 **roles_data** 파일을 사용하는 경우 파일을 편집하여 구성 가능 네트워크를 각 역할에 추가합니다. 예를 들어 컨트롤러 노드의 경우 다음과 같습니다.

```
- name: Controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
```

기본 **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml** 파일을 확인하십시오. 또한 **/usr/share/openstack-tripleo-heat-templates/roles** 에서 예제 역할 스니펫을 확인합니다.

다음 표에서는 구성 가능한 네트워크를 사용자 지정 독립 실행형 역할에 매핑합니다.

Role	필요한 네트워크
Ceph Storage 모니터	**Storage**, **StorageMgmt**
Ceph Storage OSD	**Storage**, **StorageMgmt**
Ceph Storage RadosGW	**Storage**, **StorageMgmt**
Cinder API	**InternalApi**
Compute	**InternalApi**, **Tenant**, **Storage**
컨트롤러	**external**,**InternalApi**,스토리지,**StorageMgmt**,테넌트
데이터베이스	**InternalApi**
Glance	**InternalApi**
heat	**InternalApi**
Horizon	**InternalApi**
ironic	필요하지 않음. API에 프로비저닝/컨트롤 플레인 네트워크를 사용합니다.
Keystone	**InternalApi**
로드 밸런서	**external**,**InternalApi**,스토리지,**StorageMgmt**,테넌트
Manila	**InternalApi**
메시지 버스	**InternalApi**
Networker	**InternalApi**, **Tenant**
Neutron API	**InternalApi**
Nova	**InternalApi**
OpenDaylight	외부,**InternalApi**,테넌트
Redis	**InternalApi**
Sahara	**InternalApi**
Swift API	스토리지
Swift Storage	**StorageMgmt**

Role	필요한 네트워크
telemetry	**InternalApi**

중요

이전 버전에서는 ***NetName** 매개변수(예: **InternalApiNetName**)가 기본 네트워크의 이름을 변경했습니다. 더 이상 지원되지 않습니다. 사용자 지정 구성 가능 네트워크 파일을 사용합니다. 자세한 내용은 *Advanced Overcloud Customization* 가이드의 "Composable Networks" 를 참조하십시오.

5.5. 대규모 CEPH 클러스터의 재시작 지연 증가

업그레이드 중에 각 Ceph 모니터 및 OSD가 순차적으로 중지됩니다. 중지된 동일한 서비스가 성공적으로 재시작될 때까지 마이그레이션이 계속되지 않습니다. Ansible은 15초(분기)을 대기하고 서비스가 시작될 때까지 5배(다시 시도)를 확인합니다. 서비스가 다시 시작되지 않으면 마이그레이션이 중지되어 운영자가 개입할 수 있습니다.

Ceph 클러스터 크기에 따라 재시도 또는 지연 값을 늘려야 할 수 있습니다. 이러한 매개변수와 해당 기본값은 다음과 같습니다.

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

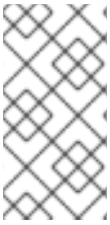
이러한 매개변수의 기본값을 업데이트할 수 있습니다. 예를 들어, Ceph OSD를 점검할 때마다 클러스터가 30초 동안 기다렸다가 Ceph MON을 점검할 때마다 10초 동안 기다린 후 **openstack overcloud deploy** 명령을 사용하여 **yaml** 파일에 다음 매개변수를 전달합니다.

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

5.6. CEPH 업그레이드 준비

OpenStack Platform 13에서는 CephMgr 서비스가 필요한 Red Hat Ceph Storage 3을 도입했습니다. OpenStack Platform 13의 기본 템플릿은 CephMgr 서비스를 포함하는 역할을 제공합니다. 그러나 구성 가능 역할 기능을 사용하여 역할을 사용자 지정하고 Overcloud가 배포된 Ceph를 사용하는 경우 CephMgr 서비스도 포함하도록 CephMon 서비스가 포함된 역할을 업데이트해야 합니다.

템플릿 비교 방법의 예는 이전 템플릿 버전 비교를 참조하십시오.



참고

역할을 사용자 정의하여 하이퍼컨버지드 오버클라우드를 배포한 경우 다음 지침을 완료해야 합니다.

절차

openstack overcloud deploy 명령에는 오버클라우드 배포에 **roles_file.yaml** 과 같은 역할 파일을 포함할 수 있습니다.

역할 파일에 **OS::TripleO::Services::CephMon** 이 포함된 경우 **CephMgr** 서비스를 역할 파일에 추가합니다.

```
OS::TripleO::Services::CephMon
OS::TripleO::Services::CephMgr
```

참고

모든 **Ceph Storage** 노드 업그레이드에 설명된 **Ceph** 업그레이드 전에 **OS::TripleO::Services::CephMgr** 을 추가해야 합니다.

5.7. 노드별 CEPH 레이아웃 생성

ceph-ansible 을 사용하여 노드별 **Ceph** 레이아웃을 생성하려면 다음을 수행하십시오.

```
parameter_defaults:
  NodeDataLookup: |
    {"6E310C04-6186-45DB-B643-54332E76FAD1": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]},
     "1E222ACE-56B9-4F14-9DB8-929734C7CAAF": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]},
     "C6841D59-9E3E-4875-BC43-FB5F49227CE7": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]}
    }
```

5.8. 사용자 정의 PUPPET 매개변수 확인

ExtraConfig 인터페이스를 사용하여 **Puppet** 매개 변수를 사용자 정의하는 경우 **Puppet**에서 업그레이드하는 동안 중복된 선언 오류를 보고할 수 있습니다. 이는 **puppet** 모듈에서 제공하는 인터페이스가 변경

되었기 때문입니다.

다음 절차에서는 환경 파일에서 사용자 정의 **ExtraConfig hieradata** 매개 변수를 확인하는 방법을 보여줍니다.

절차

1. 환경 파일을 선택하고 **ExtraConfig** 매개변수가 있는지 확인합니다.

```
$ grep ExtraConfig ~/templates/custom-config.yaml
```

2. 선택한 파일에 모든 역할(예: **Controller ExtraConfig**)에 대한 **ExtraConfig**매개변수가 결과에 표시되면 해당 파일에서 전체 매개변수 구조를 확인합니다.

3. 매개변수에 **SECTION/parameter** 구문이 있는 **puppet Hierdata**가 포함된 경우 해당 값이 실제 **Puppet** 클래스로 교체되었을 수 있습니다. 예를 들어 다음과 같습니다.

```
parameter_defaults:
  ExtraConfig:
    neutron::config::dhcp_agent_config:
      'DEFAULT/dnsmasq_local_resolv':
        value: 'true'
```

4. **director**의 **Puppet** 모듈을 확인하여 매개 변수가 **Puppet** 클래스 내에 있는지 확인합니다. 예를 들어 다음과 같습니다.

```
$ grep dnsmasq_local_resolv
```

이 경우 새 인터페이스로 변경합니다.

5. 다음은 구문 변경을 보여주는 예입니다.

- 예 **1:**

```
parameter_defaults:
  ExtraConfig:
    neutron::config::dhcp_agent_config:
```

```
    'DEFAULT/dnsmasq_local_resolv':
      value: 'true'
```

다음으로 변경합니다.

```
parameter_defaults:
  ExtraConfig:
    neutron::agents::dhcp::dnsmasq_local_resolv: true
```

- 예 2:

```
parameter_defaults:
  ExtraConfig:
    ceilometer::config::ceilometer_config:
      'oslo_messaging_rabbit/rabbit_qos_prefetch_count':
        value: '32'
```

다음으로 변경합니다.

```
parameter_defaults:
  ExtraConfig:
    oslo::messaging::rabbit::rabbit_qos_prefetch_count: '32'
```

5.9. 네트워크 인터페이스 템플릿을 새 구조로 변환

이전에는 네트워크 인터페이스 구조에서 **OS::Heat::StructuredConfig** 리소스를 사용하여 인터페이스를 구성합니다.

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            [NETWORK INTERFACE CONFIGURATION HERE]
```

이제 템플릿에서 구성에 **OS::Heat::SoftwareConfig** 리소스를 사용합니다.

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
```

```
properties:
  group: script
  config:
    str_replace:
      template:
        get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh
      params:
        $network_config:
          network_config:
            [NETWORK INTERFACE CONFIGURATION HERE]
```

이 구성은 **$network_config** 변수에 저장된 인터페이스 구성을 사용하여 **run-os-net-config.sh** 스크립트의 일부로 삽입합니다.

주의

이 새 구조를 사용하도록 네트워크 인터페이스 템플릿을 업데이트하고 네트워크 인터페이스 템플릿이 여전히 구문을 준수하도록 해야 합니다. 이 작업을 수행하지 않으면 빠른 업그레이드 프로세스 중에 오류가 발생할 수 있습니다.

director의 **Heat** 템플릿 컬렉션에는 템플릿을 이 새 형식으로 변환하는 데 도움이 되는 스크립트가 포함되어 있습니다. 이 스크립트는 **/usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py** 에 있습니다. 사용 예는 다음과 같습니다.

```
$ /usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py \
  --script-dir /usr/share/openstack-tripleo-heat-templates/network/scripts \
  [NIC TEMPLATE] [NIC TEMPLATE] ...
```

중요

이 스크립트를 사용할 때 주석 처리된 행이 템플릿에 포함되어 있지 않은지 확인합니다. 이로 인해 이전 템플릿 구조를 구문 분석할 때 오류가 발생할 수 있습니다.

자세한 내용은 "네트워크 분리" 를 참조하십시오.

참고

Red Hat OpenStack Platform 12 이하에서 고가용성(인스턴스 **HA**)을 활성화했으며 버전 **13** 이상으로 업그레이드하려면 먼저 인스턴스 **Ha**를 수동으로 비활성화해야 합니다. 자세한 내용은 이전 버전에서 인스턴스 **HA** 비활성화를 참조하십시오.

5.10. 사용자 지정 설정 파일을 수신하도록 블록 스토리지 서비스 준비

컨테이너화된 환경으로 업그레이드할 때 **CinderVolumeOptVolumes** 매개변수를 사용하여 **docker** 볼륨 마운트를 추가합니다. 이렇게 하면 컨테이너에서 실행 중일 때 호스트의 사용자 지정 구성 파일을 **cinder-volume** 서비스에서 사용할 수 있게 됩니다.

예를 들어 다음과 같습니다.

```
parameter_defaults:
  CinderVolumeOptVolumes:
    /etc/cinder/nfs_shares1:/etc/cinder/nfs_shares1
    /etc/cinder/nfs_shares2:/etc/cinder/nfs_shares2
```

5.11. 사전 프로비저닝된 노드 업그레이드 준비

사전 프로비저닝된 노드는 **director**의 관리 외부에서 생성된 노드입니다. 사전 프로비저닝된 노드를 사용하는 오버클라우드에는 업그레이드하기 전에 몇 가지 추가 단계가 필요합니다.

사전 요구 사항

- 오버클라우드는 사전 프로비저닝된 노드를 사용합니다.

절차

1. 다음 명령을 실행하여 **OVERCLOUD_HOSTS** 환경 변수에 노드 **IP** 주소 목록을 저장합니다.

```
$ source ~/stackrc
$ export OVERCLOUD_HOSTS=$(openstack server list -f value -c Networks | cut -d "=" -f 2 |
tr '\n' ' ')
```

2. 다음 스크립트를 실행합니다.

```
$ /usr/share/openstack-tripleo-heat-templates/deployed-server/scripts/enable-ssh-admin.sh
```

3. 업그레이드를 진행합니다.

- 사전 프로비저닝된 노드와 함께 **openstack overcloud upgrade run** 명령을 사용하는 경우 **--ssh-user tripleo-admin** 매개변수를 포함합니다.

- **Compute** 또는 **Object Storage** 노드를 업그레이드할 때 다음을 사용하십시오.

  a. **upgrade-non-controller.sh** 스크립트에 **-U** 옵션을 사용하고 **stack** 사용자를 지정합니다. 이는 사전 프로비저닝된 노드의 기본 사용자가 **heat-admin** 이 아닌 **stack** 이기 때문입니다.

  b. **--upgrade** 옵션과 함께 노드의 **IP** 주소를 사용합니다. 이는 노드가 **director**의 **Compute(nova)** 및 **Bare Metal(ironic)** 서비스로 관리되지 않고 노드 이름이 없기 때문입니다.

  예를 들어 다음과 같습니다.

```
$ upgrade-non-controller.sh -U stack --upgrade 192.168.24.100
```

관련 정보

- 사전 프로비저닝된 노드에 대한 자세한 내용은 ***Director** 설치 및 사용* 가이드의 "사전 프로비저닝된 노드를 사용하여 기본 오버클라우드 구성" 을 참조하십시오.

5.12. 다음 단계

오버클라우드 준비 단계가 완료되었습니다. 이제 *6장. 오버클라우드 업그레이드* 단계를 사용하여 오버클라우드를 **13**으로 업그레이드할 수 있습니다.

6장. 오버클라우드 업그레이드

이 프로세스는 오버클라우드를 업그레이드합니다.

사전 요구 사항

- 언더클라우드를 최신 버전으로 업그레이드했습니다.
- 사용자 지정 환경 파일에서 업그레이드의 변경 사항을 수용할 수 있도록 준비했습니다.

6.1. 오버클라우드 업그레이드 준비 실행

업그레이드를 수행하려면 다음 작업을 수행하는 **openstack overcloud upgrade prepare** 명령을 실행해야 합니다.

- 오버클라우드 플랜을 **OpenStack Platform 13**으로 업데이트
- 업그레이드를 위해 노드 준비

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 준비 명령을 실행합니다.

```
$ openstack overcloud upgrade prepare \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e <ENVIRONMENT FILE>
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일(**-e**)

- 새 컨테이너 이미지 위치가 있는 환경 파일입니다(**-e**). **upgrade** 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 더 이상 사용되지 않으므로 컨테이너 이미지 환경 파일에 **-e** 를 사용하므로 이 경고를 무시할 수 있습니다.

- 해당하는 경우 **--roles-file** 을 사용하여 사용자 지정 역할(**roles_data**) 파일.

- 해당하는 경우 **--networks-file** 을 사용하여 구성 가능 네트워크(**network_data**) 파일을 사용합니다.

3. 업그레이드 준비가 완료될 때까지 기다립니다.

6.2. 컨트롤러 및 사용자 정의 역할 노드 업그레이드

다음 프로세스를 사용하여 모든 컨트롤러 노드, 분할 컨트롤러 서비스 및 기타 사용자 지정 노드를 **OpenStack Platform 13**으로 업그레이드합니다. 프로세스에는 선택한 노드로만 작업을 제한하기 위한 **--nodes** 옵션을 포함하여 **openstack overcloud upgrade run** 명령을 실행해야 합니다.

```
$ openstack overcloud upgrade run --nodes [ROLE]
```

[ROLE] 을 역할 또는 쉼표로 구분된 역할 목록으로 대체합니다.

오버클라우드에서 모놀리식 컨트롤러 노드를 사용하는 경우 컨트롤러 역할에 대해 이 명령을 실행합니다.

오버클라우드에서 분할 컨트롤러 서비스를 사용하는 경우 다음 가이드를 사용하여 다음 순서로 노드 역할을 업그레이드합니다.

- **Pacemaker**를 사용하는 모든 역할입니다. 예: **ControllerOpenStack,Database,Messaging, Telemetry.**

- **Networker** 노드

- 기타 사용자 정의 역할

다음 노드를 아직 업그레이드 *하지 마십시오*.

- **DPDK** 기반 또는 **Hyper-Converged Infratructure (HCI) Compute** 노드와 같은 모든 유형의 컴퓨팅 노드

- **CephStorage** 노드

이러한 노드를 이후 단계에서 업그레이드합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 모놀리식 컨트롤러 노드를 사용하는 경우 컨트롤러 역할에 대해 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes Controller
```

- 사용자 지정 스택 이름을 사용하는 경우 **name**을 **--stack** 옵션으로 전달합니다.

3. 컨트롤러 서비스를 사용하는 경우 여러 역할로 나뉩니다.

a. **Pacemaker** 서비스를 사용하는 역할에 대해 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes ControllerOpenStack
$ openstack overcloud upgrade run --nodes Database
$ openstack overcloud upgrade run --nodes Messaging
$ openstack overcloud upgrade run --nodes Telemetry
```

- 사용자 지정 스택 이름을 사용하는 경우 **name**을 **--stack** 옵션으로 전달합니다.

b. **Networker** 역할에 대해 **upgrade** 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes Networker
```

- 사용자 지정 스택 이름을 사용하는 경우 **name**을 **--stack** 옵션으로 전달합니다.

c. **Compute** 또는 **CephStorage** 역할을 제외한 나머지 사용자 지정 역할에 대해 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes ObjectStorage
```

- 사용자 지정 스택 이름을 사용하는 경우 **name**을 **--stack** 옵션으로 전달합니다.

6.3. 모든 컴퓨팅 노드 업그레이드

중요

- 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 6.5절. “하이퍼컨버지드 노드 업그레이드” 에서 참조하십시오.

- 혼합된 하이퍼컨버지드 배포를 사용하는 경우 6.6절. “혼합 하이퍼컨버지드 노드 업그레이드” 에서 업그레이드하는 방법을 참조하십시오.

이 프로세스에서는 나머지 **Compute** 노드를 모두 **OpenStack Platform 13**으로 업그레이드합니다. 이 프로세스에는 컴퓨팅 노드로만 작업을 제한하기 위해 **openstack overcloud upgrade run** 명령을 실행하고 **--nodes Compute** 옵션을 포함하여 작업을 수행해야 합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes Compute
```

- 사용자 지정 스택 이름을 사용하는 경우 **name**을 **--stack** 옵션으로 전달합니다.
- 사용자 정의 컴퓨팅 역할을 사용하는 경우 **--nodes** 옵션이 있는 역할 이름을 포함해야 합니다.

3. 컴퓨팅 노드 업그레이드가 완료될 때까지 기다립니다.

6.4. 모든 CEPH STORAGE 노드 업그레이드

중요

- 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 **6.5절. "하이퍼컨버지드 노드 업그레이드"** 에서 참조하십시오.
- 혼합된 하이퍼컨버지드 배포를 사용하는 경우 **6.6절. "혼합 하이퍼컨버지드 노드 업그레이드"** 에서 업그레이드하는 방법을 참조하십시오.

이 프로세스는 **Ceph Storage** 노드를 업그레이드합니다. 이 과정에는 다음이 포함됩니다.

- **openstack overcloud upgrade run** 명령을 실행하고 **--nodes CephStorage** 옵션을 포함하여 작업을 **Ceph Storage** 노드로만 제한합니다.
- **openstack overcloud ceph-upgrade run** 명령을 실행하여 컨테이너화된 **Red Hat Ceph Storage 3** 클러스터에 대한 업그레이드를 수행합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes CephStorage
```

- 사용자 지정 스택 이름을 사용하는 경우 이름을 **--stack** 옵션으로 전달합니다.

3. 노드 업그레이드가 완료될 때까지 기다립니다.

4. **Ceph Storage** 업그레이드 명령을 실행합니다. 예를 들어 다음과 같습니다.

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일**(-e).** 예를 들어 다음과 같습니다.
  - 컨테이너 이미지 위치가 있는 환경 파일**(overcloud_images.yaml). upgrade** 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 더 이상 사용되지 않으므로 컨테이너 이미지 환경 파일에 **-e** 를 사용하므로 이 경고를 무시할 수 있습니다.
  - **Ceph Storage** 노드의 관련 환경 파일입니다.
  - 환경과 관련된 추가 환경 파일입니다.
- 사용자 지정 스택 이름을 사용하는 경우 이름을 **--stack** 옵션으로 전달합니다.
- 해당하는 경우 **--roles-file** 을 사용하여 사용자 지정 역할**(roles_data)** 파일.

- 해당하는 경우 **--networks-file** 을 사용하여 구성 가능 네트워크(**network_data**) 파일을 사용합니다.

5. **Ceph Storage** 노드 업그레이드가 완료될 때까지 기다립니다.

6.5. 하이퍼컨버지드 노드 업그레이드

ComputeHCI 역할의 하이퍼컨버지드 노드만 사용하고 전용 계산 노드 또는 전용 **Ceph** 노드를 사용하지 않는 경우 다음 절차를 완료하여 노드를 업그레이드하십시오.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --roles ComputeHCI
```

사용자 지정 스택 이름을 사용하는 경우 **--stack** 옵션을 사용하여 이름을 업그레이드 명령에 전달합니다.

3. **Ceph Storage** 업그레이드 명령을 실행합니다. 예를 들어 다음과 같습니다.

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일(**-e**). 예를 들어 다음과 같습니다.

- 컨테이너 이미지 위치가 있는 환경 파일(**overcloud_images.yaml**). **upgrade** 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 더 이상 사용되지 않으므로 컨테이너 이미지 환경 파일에 **-e** 를 사용하므로 이 경고를 무시할 수 있습니다.

- **Ceph Storage** 노드의 관련 환경 파일입니다.

- 사용자 지정 스택 이름을 사용하는 경우 이름을 **--stack** 옵션으로 전달합니다.

- 해당하는 경우 **--roles-file** 을 사용하여 사용자 지정 역할(**roles_data**) 파일.

- 해당하는 경우 **--networks-file** 을 사용하여 구성 가능 네트워크(**network_data**) 파일을 사용합니다.

4. **Ceph Storage** 노드 업그레이드가 완료될 때까지 기다립니다.

6.6. 혼합 하이퍼컨버지드 노드 업그레이드

ComputeHCI 역할과 같은 하이퍼컨버지드 노드 외에도 전용 컴퓨팅 노드 또는 전용 **ceph** 노드를 사용하는 경우 다음 절차를 완료하여 노드를 업그레이드합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 컴퓨팅 노드에 대해 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --roles Compute
If using a custom stack name, pass the name with the --stack option.
```

3. 노드 업그레이드가 완료될 때까지 기다립니다.

4. **ComputeHCI 노드에 대해 업그레이드 명령을 실행합니다.**

```
$ openstack overcloud upgrade run --roles ComputeHCI
If using a custom stack name, pass the name with the --stack option.
```

5. **노드 업그레이드가 완료될 때까지 기다립니다.**

6. **Ceph Storage 노드에 대해 업그레이드 명령을 실행합니다.**

```
$ openstack overcloud upgrade run --roles CephStorage
```

7. **Ceph Storage 노드 업그레이드가 완료될 때까지 기다립니다.**

8. **Ceph Storage 업그레이드 명령을 실행합니다. 예를 들어 다음과 같습니다.**

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

환경과 관련된 다음 옵션을 포함합니다.

- **사용자 지정 구성 환경 파일(-e). 예를 들어 다음과 같습니다.**
  - **컨테이너 이미지 위치가 있는 환경 파일(overcloud_images.yaml). upgrade 명령은 --container-registry-file 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 더 이상 사용되지 않으므로 컨테이너 이미지 환경 파일에 -e 를 사용하므로 이 경고를 무시할 수 있습니다.**
  - **Ceph Storage 노드의 관련 환경 파일입니다.**
  - **환경과 관련된 추가 환경 파일입니다.**

- 사용자 지정 스택 이름을 사용하는 경우 이름을 **--stack** 옵션으로 전달합니다.
- 해당하는 경우 **--roles-file** 을 사용하여 사용자 지정 역할(**roles_data**) 파일.
- 해당하는 경우 **--networks-file** 을 사용하여 구성 가능 네트워크(**network_data**) 파일을 사용합니다.

9. **Ceph Storage** 노드 업그레이드가 완료될 때까지 기다립니다.

6.7. 업그레이드 종료

업그레이드하려면 오버클라우드 스택을 업데이트하는 최종 단계가 필요합니다. 이렇게 하면 스택의 리소스 구조가 **OpenStack Platform 13**의 일반 배포와 정렬되고 향후 표준 **openstack overcloud deploy** 기능을 수행할 수 있습니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 완료 명령을 실행합니다.

```
$ openstack overcloud upgrade converge \
  --templates \
  -e <ENVIRONMENT FILE>
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일(**-e**).
- 사용자 지정 스택 이름을 사용하는 경우 이름을 **--stack** 옵션으로 전달합니다.

- 해당하는 경우 **--roles-file** 을 사용하여 사용자 지정 역할(roles_data) 파일.

- 해당하는 경우 **--networks-file** 을 사용하여 구성 가능 네트워크(network_data) 파일을 사용합니다.

3. 업그레이드가 완료될 때까지 기다립니다.

7장. 업그레이드 단계 실행

이 프로세스는 기본 업그레이드 프로세스를 완료한 후 최종 단계를 구현합니다.

사전 요구 사항

- 최신 주요 릴리스로 오버클라우드 업그레이드를 완료했습니다.

7.1. 오버클라우드 업그레이드 후 일반 고려 사항

다음 항목은 오버클라우드 업그레이드 후 일반적인 고려 사항입니다.

- 필요한 경우 오버클라우드 노드에서 결과 설정 파일을 검토합니다. 업그레이드된 패키지가 각 서비스의 업그레이드된 버전에 적합한 **.rpmnew** 파일을 설치했을 수 있습니다.

- 컴퓨팅 노드에서 **neutron-openvswitch-agent** 를 사용하여 오류를 보고할 수 있습니다. 이 경우 각 컴퓨팅 노드에 로그인하고 서비스를 다시 시작합니다. 예를 들어 다음과 같습니다.

```
$ sudo docker restart neutron_ovs_agent
```

- 경우에 따라 컨트롤러 노드를 재부팅한 후 **corosync** 서비스가 **IPv6** 환경에서 시작되지 않을 수 있습니다. 이는 컨트롤러 노드가 정적 **IPv6** 주소를 구성하기 전에 **Corosync** 시작 때문입니다. 이러한 경우 컨트롤러 노드에서 **Corosync**를 수동으로 다시 시작합니다.

```
$ sudo systemctl restart corosync
```