Red Hat OpenStack Platform 13

Fast Forward Upgrades

Red Hat OpenStack Platform 10에서 13으로 긴 라이프 버전에서 업그레이드

Last Updated: 2023-05-25

Red Hat OpenStack Platform 13 Fast Forward Upgrades

Red Hat OpenStack Platform 10에서 13으로 긴 라이프 버전에서 업그레이드

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 가이드에서는 빠른 전달 업그레이드 프로세스를 제공합니다. 이 프로세스는 OpenStack Platform 환경을 긴 수명 버전에서 다음 장기 라이프 버전으로 업그레이드합니다. 이 경우 이 가이드에서는 Red Hat OpenStack Platform 10(Newton)에서 13(Queens)으로의 업그레이드에 중점을 둡니다.

차례

1장. 소개 **4**
1.1. 시작하기 전 4
1.2. FAST FORWARD UPGRADE 4
1.3. 상위 수준의 워크플로 4
1.4. 업그레이드 전에 CEPH 클러스터 상태 확인 6

2장. OPENSTACK PLATFORM 업그레이드 준비 **7**
2.1. 베어 메탈 UNDERCLOUD 백업 생성 7
2.2. 오버클라우드 컨트롤 플레인 서비스 백업 8
2.3. OPENSTACK PLATFORM 10.Z의 현재 언더클라우드 패키지 업데이트 11
2.4. NFV 지원 환경에 대한 업데이트 준비 12
2.5. OPENSTACK PLATFORM 10.Z의 현재 오버클라우드 이미지 업데이트 13
2.6. OPENSTACK PLATFORM 10.Z의 현재 오버클라우드 패키지 업데이트 14
2.7. 컨트롤러 및 구성 가능한 노드 재부팅 17
2.8. CEPH STORAGE(OSD) 클러스터 재부팅 17
2.9. 컴퓨팅 노드 재부팅 18
2.10. 시스템 패키지 확인 19
2.11. OPENSTACK PLATFORM 10 언더클라우드 검증 20
2.12. OPENSTACK PLATFORM 10 오버클라우드 검증 20
2.13. NFV 지원 환경의 업데이트 완료 23
2.14. YUM 이력 유지 24
2.15. 다음 단계 24

3장. 언더클라우드 업그레이드 **25**
3.1. 언더클라우드를 OPENSTACK PLATFORM 11으로 업그레이드 25
3.2. 언더클라우드를 OPENSTACK PLATFORM 12로 업그레이드 26
3.3. 언더클라우드를 OPENSTACK PLATFORM 13으로 업그레이드 27
3.4. 언더클라우드에서 더 이상 사용되지 않는 서비스 비활성화 28
3.5. 다음 단계 28

4장. 컨테이너 이미지 소스 구성 **29**
4.1. 레지스트리 방법 29
4.2. 컨테이너 이미지 준비 명령 사용 29
4.3. 추가 서비스의 컨테이너 이미지 31
4.4. RED HAT 레지스트리를 원격 레지스트리 소스로 사용 34
4.5. 언더클라우드를 로컬 레지스트리로 사용 35
4.6. SATELLITE 서버를 레지스트리로 사용 37
4.7. 다음 단계 40

5장. 오버클라우드 업그레이드 준비 **41**
5.1. 오버클라우드 서비스 다운타임 준비 41
5.2. 업그레이드 테스트용 컴퓨팅 노드 선택 41
5.3. 새로운 구성 가능 서비스 42
5.4. 사용되지 않는 구성 가능 서비스 43
5.5. 컨테이너화된 서비스로 전환 44
5.6. 더 이상 사용되지 않는 매개변수 45
5.7. 더 이상 사용되지 않는 CLI 옵션 47
5.8. 구성 가능 네트워크 50
5.9. CEPH STORAGE 또는 HCI 노드 업그레이드 준비 51
5.10. 동종 디스크가 아닌 CEPH 또는 HCI 노드의 환경 변수 업데이트 54
5.11. 대규모 CEPH 클러스터의 재시작 지연 증가 55
5.12. 스토리지 백엔드 준비 55

5.13. SSL/TLS를 통한 UNDERCLOUD의 공용 API에 대한 액세스 준비 56
5.14. 빠른 전달 업그레이드를 위한 등록 구성 57
5.15. 사용자 정의 PUPPET 매개변수 확인 59
5.16. 네트워크 인터페이스 템플릿을 새 구조로 변환 60
5.17. DPDK 및 SR-IOV 구성 확인 61
5.18. 사전 프로비저닝된 노드 업그레이드 준비 64
5.19. 다음 단계 65

6장. 오버클라우드 업그레이드 66
6.1. FAST FORWARD UPGRADE 명령 66
6.2. 오버클라우드의 빠른 전달 업그레이드 수행 67
6.3. 컨트롤러 및 사용자 정의 역할 노드 업그레이드 71
6.4. 테스트 컴퓨팅 노드 업그레이드 74
6.5. 모든 컴퓨팅 노드 업그레이드 74
6.6. 모든 CEPH STORAGE 노드 업그레이드 76
6.7. 하이퍼컨버지드 노드 업그레이드 78
6.8. 혼합 하이퍼컨버지드 노드 업그레이드 80
6.9. FAST FORWARD 업그레이드 종료 82
6.10. 다음 단계 83

7장. 업그레이드 후 오버클라우드 재부팅 84
7.1. 컨트롤러 및 구성 가능한 노드 재부팅 84
7.2. CEPH STORAGE(OSD) 클러스터 재부팅 85
7.3. 컴퓨팅 노드 재부팅 86
7.4. 컴퓨팅 HCI 노드 재부팅 87

8장. 업그레이드 후 단계 실행 91
8.1. 언더클라우드 검증 91
8.2. 컨테이너화된 오버클라우드 검증 92
8.3. 오버클라우드 이미지 업그레이드 95
8.4. 배포 테스트 97
8.5. CONCLUSION 97

부록 A. 언더클라우드 복원 98

부록 B. 오버클라우드 복원 105
B.1. 오버클라우드 컨트롤 플레인 서비스 복원 105
B.2. 복원된 고가용성 서비스 113
B.3. 복원된 컨트롤러 서비스 114
B.4. 복원된 오버클라우드 계산 서비스 116

1장. 소개

이 문서에서는 Red Hat OpenStack Platform 환경을 최신 장기 버전으로 업그레이드하는 데 도움이 되는 워크플로를 제공합니다.

1.1. 시작하기 전

다음을 확인합니다.

- 버전 7 또는 8을 사용하여 Red Hat OpenStack Platform 환경을 처음 배포한 경우, 컨테이너화된 서비스의 업그레이드 경로 및 배포에 문제가 발생할 수 있는 이전 버전의 XFS 파일 시스템에 문제가 있습니다. 문제 및 문제 해결 방법에 대한 자세한 내용은 "XFS ftype=0이 컨테이너를 사용하는 OpenStack Director 버전으로 업그레이드하지 못하도록 하는 문서를 참조하십시오.

- 배포에 RHCS(Red Hat Ceph Storage) 노드가 포함된 경우 각 Ceph 개체 스토리지 데몬(OSD)에 대한 배치 그룹(PG) 수가 기본적으로 250개를 초과해서는 안 됩니다. OSD당 더 많은 PG로 Ceph 노드를 업그레이드하면 경고 상태가 되며 업그레이드 프로세스가 실패할 수 있습니다. 업그레이드 프로세스를 시작하기 전에 OSD당 PG 수를 늘릴 수 있습니다. 이 문제를 진단하고 해결하는 방법에 대한 자세한 내용은 하나 이상의 OSD에 할당된 Ceph PG가 250보다 높기 때문에 OpenStack FFU 문서를 10에서 13번으로 참조하십시오.

- **prevent_arp_spoofing** 이 False로 설정된 모든 포트를 찾습니다. 포트 보안이 비활성화된 포트에 대해 을 확인합니다. 업그레이드의 일부로 **prevent_arp_spoofing** 옵션이 제거되고 해당 기능은 포트 보안에 의해 제어됩니다.

- 업그레이드를 수행하기 전에 펌웨어 업데이트를 하드웨어에 적용합니다.

- 애플리케이션 암호를 포함하여 배포된 오버클라우드를 수동으로 변경한 경우 이러한 변경 사항으로 director 배포 템플릿을 업데이트하여 업그레이드에 실패하지 않도록 해야 합니다. 문의 사항이 있으시면 Red Hat 기술 지원에 문의하십시오.

1.2. FAST FORWARD UPGRADE

Red Hat OpenStack Platform은 **신속한 업그레이드** 기능을 제공합니다. 이 기능은 여러 버전의 오버클라우드를 통해 업그레이드 경로를 제공합니다. 목표는 사용자에게 **긴 수명 버전으로 간주되는 특정 OpenStack 버전을 유지하고 차후의 라이프사이클 버전을** 사용할 수 있을 때 업그레이드할 수 있는 기회를 사용자에게 제공하는 것입니다.

이 가이드에서는 다음 버전을 통해 빠른 전달 업그레이드 경로를 제공합니다.

이전 버전	새 버전
Red Hat OpenStack Platform 10	Red Hat OpenStack Platform 13

1.3. 상위 수준의 워크플로

다음 표에서는 빠른 전달 업그레이드 프로세스에 필요한 단계와 각 업그레이드 프로세스 단계의 기간 및 영향에 대한 추정치에 대해 간략히 설명합니다.

참고

이 표의 기간은 내부 테스트를 기반으로 하는 최소 추정치이며 모든 프로덕션 환경에 적용되지 않을 수 있습니다. 각 작업의 업그레이드 기간을 정확하게 측정하려면 프로덕션 환경과 유사한 하드웨어를 사용하여 테스트 환경에서 다음 절차를 수행합니다.

표 1.1. Fast Forward Upgrade 프로세스 단계 개요 및 영향

Step	설명	기간
환경 준비	언더클라우드 노드 및 오버클라우드 컨트롤러 노드에 대한 데이터베이스 및 구성을 백업합니다. 최신 마이너 릴리스로 업데이트한 후 재부팅합니다. 환경을 검증합니다.	이 단계의 기간은 배포 크기에 따라 다를 수 있습니다.
언더클라우드 업그레이드	OpenStack Platform 10에서 OpenStack Platform 13으로 연속되는 각 버전의 언더클라우드로 업그레이드합니다.	언더클라우드 업그레이드의 예상 시간은 약 60분이며 업그레이드 중에 언더클라우드 다운타임이 발생합니다. Undercloud 업그레이드 단계에서는 Overcloud가 계속 작동합니다.
컨테이너 이미지 가져오기	다양한 OpenStack 서비스에 대한 컨테이너 이미지의 위치가 포함된 환경 파일을 만듭니다.	컨테이너 이미지 소스를 구성하는 데 필요한 기간은 약 10분입니다.
오버클라우드 준비	오버클라우드 구성 파일을 OpenStack Platform 13으로 전환하려면 관련 단계를 수행합니다.	업그레이드를 위해 Overcloud를 준비하는 데 필요한 예상 기간은 약 20분입니다.
fast forward 업그레이드 수행	최신 OpenStack Platform director 템플릿 세트를 사용하여 오버클라우드 계획을 업그레이드합니다. 데이터베이스 스키마가 OpenStack Platform 13으로 업그레이드할 준비가 되도록 각 순차적 버전을 통해 패키지 및 데이터베이스 업그레이드를 실행합니다.	오버클라우드 업그레이드 실행의 예상 기간은 약 30분이며 업그레이드 중에 오버클라우드 서비스 중단이 발생합니다. 중단하는 동안 OpenStack 작업을 수행할 수 없습니다.
컨트롤러 노드 업그레이드	모든 컨트롤러 노드를 OpenStack Platform 13으로 동시에 업그레이드합니다.	컨트롤러 노드 업그레이드의 예상 기간은 약 50분입니다. 컨트롤러 노드를 업그레이드하는 동안 짧은 오버클라우드 서비스 다운타임이 발생할 수 있습니다.

Step	설명	기간
컴퓨팅 노드 업그레이드	선택한 컴퓨팅 노드에서 업그레이드를 테스트합니다. 테스트에 성공하면 모든 컴퓨팅 노드를 업그레이드합니다.	Compute 노드 업그레이드의 예상 기간은 노드당 약 25분입니다. 컴퓨팅 노드를 업그레이드하는 동안 워크로드에 대한 가동 중단이 예상되지 않습니다.
Ceph Storage 노드 업그레이드	모든 Ceph Storage 노드를 업그레이드합니다. 여기에는 컨테이너화된 Red Hat Ceph Storage 3 버전으로의 업그레이드가 포함됩니다.	Ceph Storage 노드 업그레이드의 예상 기간은 노드당 약 25분입니다. Ceph Storage 노드를 업그레이드하는 동안 중단 시간이 예상되지 않습니다.
업그레이드 종료	통합 명령을 실행하여 오버클라우드 스택을 새로 고칩니다.	오버클라우드 수렴 실행의 예상 기간은 1시간 이상 소요되지만 환경에 따라 시간이 더 오래 걸릴 수 있습니다.

1.4. 업그레이드 전에 CEPH 클러스터 상태 확인

환경을 업그레이드하기 전에 Ceph 클러스터가 활성화되어 예상대로 작동하는지 확인해야 합니다.

절차

1. **ceph-mon** 서비스를 실행 중인 노드에 로그인합니다. 이 노드는 일반적으로 컨트롤러 노드 또는 독립 실행형 Ceph 모니터 노드입니다.
2. Ceph 클러스터의 상태 보기:

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

3. 클러스터의 상태가 **HEALTH_OK** 이고 모든 OSD(오브젝트 스토리지 데몬)가 활성화되어 있는지 확인합니다.

2장. OPENSTACK PLATFORM 업그레이드 준비

이 프로세스는 OpenStack Platform 환경을 준비합니다. 여기에는 다음 단계가 포함됩니다.

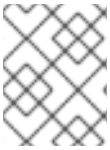
- Undercloud와 Overcloud를 둘 다 백업합니다.
- 최신 Open vSwitch를 포함하여 언더클라우드를 OpenStack Platform 10의 최신 마이너 버전으로 업데이트합니다.
- 최신 커널 또는 최신 시스템 패키지가 설치된 경우 언더클라우드를 재부팅합니다.
- 최신 Open vSwitch를 포함하여 오버클라우드를 OpenStack Platform 10의 최신 마이너 버전으로 업데이트합니다.
- 최신 커널 또는 최신 시스템 패키지가 설치된 경우 오버클라우드 노드를 재부팅합니다.
- 언더클라우드 및 오버클라우드 모두에서 검증을 수행합니다.

다음 절차에서는 업그레이드를 진행하기 전에 OpenStack Platform 환경이 최상의 상태인지 확인합니다.

2.1. 베어 메탈 UNDERCLOUD 백업 생성

전체 언더클라우드 백업에는 다음과 같은 데이터베이스 및 파일이 포함됩니다.

- 언더클라우드 노드의 모든 MariaDB 데이터베이스
- 언더클라우드의 MariaDB 구성 파일(데이터베이스를 정확하게 복원할 수 있음)
- 설정 데이터: **/etc**
- 로그 데이터: **/var/log**
- 이미지 데이터: **/var/lib/glance**
- SSL을 사용하는 경우 인증서 생성 데이터: **/var/lib/certmonger**
- 모든 컨테이너 이미지 데이터: **/var/lib/docker** 및 **/var/lib/registry**
- 모든 swift 데이터: **/srv/node**
- stack 사용자 홈 디렉토리의 모든 데이터: **/home/stack**



참고

백업 프로세스를 수행하기 전에 언더클라우드에서 사용할 수 있는 디스크 공간이 충분한지 확인합니다. 더 크지 않은 경우 아카이브 파일이 3.5GB 이상이어야 합니다.

절차

1. **root** 사용자로 언더클라우드에 로그인합니다.
2. 데이터베이스를 백업합니다.

```
[root@director ~]# mysqldump --opt --all-databases > /root/undercloud-all-databases.sql
```

3. **백업** 디렉터리를 생성하고 디렉터리의 사용자 소유권을 **stack** 사용자로 변경합니다.

```
[root@director ~]# mkdir /backup
[root@director ~]# chown stack: /backup
```

이 디렉터리를 사용하여 Undercloud 데이터베이스 및 파일 시스템이 포함된 아카이브를 저장합니다.

4. **백업** 디렉터리로 변경합니다.

```
[root@director ~]# cd /backup
```

5. 데이터베이스 백업 및 구성 파일을 보관합니다.

```
[root@director ~]# tar --xattrs --xattrs-include='*.*' --ignore-failed-read -cf \
  undercloud-backup-$(date +%F).tar \
  /root/undercloud-all-databases.sql \
  /etc \
  /var/log \
  /var/lib/glance \
  /var/lib/certmonger \
  /var/lib/docker \
  /var/lib/registry \
  /srv/node \
  /root \
  /home/stack
```

- **--ignore-failed-read** 옵션은 언더클라우드에 적용되지 않는 디렉터리를 건너뜁니다.
- **--xattrs** 및 **--xattrs-include='.'** 옵션에는 오브젝트 스토리지(swift) 및 SELinux에 대한 메타데이터를 저장하는 데 필요한 확장 속성이 포함되어 있습니다.

그러면 **undercloud-backup-<date>.tar.gz라는 파일이 생성됩니다.** 여기서 **<date>** 는 시스템 날짜입니다. 이 **tar** 파일을 안전한 위치에 복사합니다.

관련 정보

- 언더클라우드 백업을 복원해야 하는 경우 부록 A. *언더클라우드 복원* 을 참조하십시오.

2.2. 오버클라우드 컨트롤 플레인 서비스 백업

다음 절차에서는 오버클라우드 데이터베이스 및 구성의 백업을 생성합니다. 오버클라우드 데이터베이스 및 서비스의 백업에서는 작업 환경의 스냅샷을 보유합니다. 이 스냅샷을 사용하면 작동 중단 시 오버클라우드를 원래 상태로 복원해야 합니다.

중요

이 절차에는 중요한 컨트롤 플레인 서비스만 포함됩니다. 컴퓨팅 노드 워크로드 백업, Ceph Storage 노드의 데이터, 추가 서비스는 포함되지 않습니다.

절차

1. 데이터베이스 백업을 수행합니다.

a. 컨트롤러 노드에 로그인합니다. 언더클라우드에서 오버클라우드에 액세스할 수 있습니다.

```
$ ssh heat-admin@192.0.2.100
```

b. **root** 사용자로 변경합니다.

```
$ sudo -i
```

c. 백업을 저장할 임시 디렉터리를 생성합니다.

```
# mkdir -p /var/tmp/mysql_backup/
```

d. 데이터베이스 암호를 가져와서 **MYSQLDBPASS** 환경 변수에 저장합니다. 암호는 **/etc/puppet/hieradata/service _configs.json 파일의 mysql::server::root_ password** 변수에 저장됩니다. 다음 명령을 사용하여 암호를 저장합니다.

```
# MYSQLDBPASS=$(sudo hiera -c /etc/puppet/hiera.yaml mysql::server::root_password)
```

e. 데이터베이스를 백업합니다.

```
# mysql -uroot -p$MYSQLDBPASS -s -N -e "select distinct table_schema from
information_schema.tables where engine='innodb' and table_schema != 'mysql';" | xargs
mysqldump -uroot -p$MYSQLDBPASS --single-transaction --databases >
/var/tmp/mysql_backup/openstack_databases-$(date +%F)-$(date +%T).sql
```

이렇게 하면 **/var/tmp/mysql_backup/openstack_databases-<date>.sql이라는 데이터베이스 백업을 덤프합니다.** 여기서 **<date>** 는 시스템 날짜와 시간입니다. 이 데이터베이스 덤프를 안전한 위치에 복사합니다.

f. 모든 사용자 및 권한 정보를 백업합니다.

```
# mysql -uroot -p$MYSQLDBPASS -s -N -e "SELECT CONCAT('\"SHOW GRANTS FOR
'',user,''@'',host,'';\"') FROM mysql.user where (length(user) > 0 and user NOT LIKE
'root')" | xargs -n1 mysql -uroot -p$MYSQLDBPASS -s -N -e | sed 's/$/;/' >
/var/tmp/mysql_backup/openstack_databases_grants-$(date +%F)-$(date +%T).sql
```

이렇게 하면 **/var/tmp/mysql_backup/openstack_databases_grants-<date>.sql이라는 데이터베이스 백업을 덤프합니다.** 여기서 **<date>** 는 시스템 날짜와 시간입니다. 이 데이터베이스 덤프를 안전한 위치에 복사합니다.

2. Pacemaker 구성을 백업합니다.

a. 컨트롤러 노드에 로그인합니다.

b. 다음 명령을 실행하여 현재 Pacemaker 구성의 아카이브를 생성합니다.

```
# sudo pcs config backup pacemaker_controller_backup
```

c. 결과 아카이브(**pacemaker_controller_backup.tar.bz2)**를안전한 위치에 복사합니다.

3. OpenStack Telemetry 데이터베이스를 백업합니다.

a. 컨트롤러에 연결하고 MongoDB 기본 인스턴스의 IP를 가져옵니다.

```
# MONGOIP=$(sudo hiera -c /etc/puppet/hiera.yaml mongodb::server::bind_ip)
```

b. 백업을 생성합니다.

```
# mkdir -p /var/tmp/mongo_backup/
# mongodump --oplog --host $MONGOIP --out /var/tmp/mongo_backup/
```

c. **/var/tmp/mongo_backup/** 의 데이터베이스 덤프를 안전한 위치로 복사합니다.

4. Redis 클러스터를 백업합니다.

a. HAProxy에서 Redis 끝점을 가져옵니다.

```
# REDISIP=$(sudo hiera -c /etc/puppet/hiera.yaml redis_vip)
```

b. Redis 클러스터의 마스터 암호를 가져옵니다.

```
# REDISPASS=$(sudo hiera -c /etc/puppet/hiera.yaml redis::masterauth)
```

c. Redis 클러스터에 대한 연결을 확인합니다.

```
# redis-cli -a $REDISPASS -h $REDISIP ping
```

d. Redis 데이터베이스를 덤프합니다.

```
# redis-cli -a $REDISPASS -h $REDISIP bgsave
```

데이터베이스 백업을 기본 **/var/lib/redis/** 디렉토리에 저장합니다. 이 데이터베이스 덤프를 안전한 위치에 복사합니다.

5. 각 컨트롤러 노드에서 파일 시스템을 백업합니다.

a. 백업을 위한 디렉터리를 생성합니다.

```
# mkdir -p /var/tmp/filesystem_backup/
```

b. 다음 **tar** 명령을 실행합니다.

```
# tar --acls --ignore-failed-read --xattrs --xattrs-include='*.*' \
   -zcvf /var/tmp/filesystem_backup/`hostname`-filesystem-`date '+%Y-%m-%d-%H-%M-
%S'`.tar \
   /etc \
   /srv/node \
   /var/log \
   /var/lib/nova \
   --exclude /var/lib/nova/instances \
   /var/lib/glance \
   /var/lib/keystone \
   /var/lib/cinder \
   /var/lib/heat \
   /var/lib/heat-config \
   /var/lib/heat-cfntools \
   /var/lib/rabbitmq \
   /var/lib/neutron \
```

```
/var/lib/haproxy \
/var/lib/openvswitch \
/var/lib/redis \
/var/lib/os-collect-config \
/usr/libexec/os-apply-config \
/usr/libexec/os-refresh-config \
/home/heat-admin
```

--ignore-failed-read 옵션은 누락된 디렉터리를 무시합니다. 이는 특정 서비스가 자체 사용자 지정 역할에서 사용하거나 구분되지 않은 경우 유용합니다.

c. 결과 **tar** 파일을 안전한 위치에 복사합니다.

6. 오버클라우드에서 삭제된 행을 보관합니다.

a. 삭제된 아카이브 인스턴스를 확인합니다.

```
$ source ~/overcloudrc
$ nova list --all-tenants --deleted
```

b. 아카이브에 보관된 인스턴스가 없는 경우 오버클라우드 컨트롤러 노드 중 하나에서 다음 명령을 입력하여 삭제된 인스턴스를 보관합니다.

```
# su - nova -s /bin/bash -c "nova-manage --debug db archive_deleted_rows --max_rows 1000"
```

삭제된 모든 인스턴스를 보관할 때까지 이 명령을 다시 실행합니다.

c. 오버클라우드 컨트롤러 노드 중 하나에 다음 명령을 입력하여 아카이브에서 삭제된 모든 인스턴스를 삭제합니다.

```
# su - nova -s /bin/bash -c "nova-manage --debug db purge --all --all-cells"
```

d. 보관된 나머지 인스턴스가 없는지 확인합니다.

```
$ nova list --all-tenants --deleted
```

관련 정보

- 오버클라우드 백업을 복원해야 하는 경우 부록 B. *오버클라우드 복원* 을 참조하십시오.

2.3. OPENSTACK PLATFORM 10.Z의 현재 언더클라우드 패키지 업데이트

director는 언더클라우드 노드에서 패키지를 업데이트하는 명령을 제공합니다. 이를 통해 현재 버전의 OpenStack Platform 환경에서 마이너 업데이트를 수행할 수 있습니다. 이는 **OpenStack Platform 10** 의 마이너 업데이트입니다.

참고

이 단계에서는 언더클라우드 운영 체제를 최신 버전의 Red Hat Enterprise Linux 7 및 Open vSwitch로 업데이트합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. 기본 OpenStack Platform 서비스를 중지합니다.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

참고

이로 인해 언더클라우드에 대해 짧은 다운타임이 발생합니다. Undercloud 업그레이드 중에 Overcloud가 계속 작동합니다.

3. RHEL 버전을 RHEL 7.7로 설정합니다.

```
$ sudo subscription-manager release --set=7.7
```

4. **python-tripleoclient** 패키지 및 해당 종속성을 업데이트하여 마이너 버전 업데이트에 대한 최신 스크립트가 있는지 확인합니다.

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

5. **openstack undercloud upgrade** 명령을 실행합니다.

```
$ openstack undercloud upgrade
```

6. 명령이 실행을 완료할 때까지 기다립니다.
7. 언더클라우드를 재부팅하여 운영 체제의 커널 및 기타 시스템 패키지를 업데이트합니다.

```
$ sudo reboot
```

8. 노드가 부팅될 때까지 기다립니다.
9. **stack** 사용자로 언더클라우드에 로그인합니다.

언더클라우드 패키지 업데이트 외에도 오버클라우드 이미지를 최신 상태로 유지하여 이미지 구성을 최신 **openstack-tripleo-heat-template** 패키지와 동기화하는 것이 좋습니다. 이렇게 하면 현재 준비 단계와 실제 빠른 전달 업그레이드 사이에 배포 및 확장 작업이 성공적으로 수행됩니다. 다음 섹션에서는 이 시나리오에서 이미지를 업데이트하는 방법을 보여줍니다. 환경을 준비한 후 즉시 환경을 업그레이드하려는 경우 다음 섹션을 건너뛸 수 있습니다.

2.4. NFV 지원 환경에 대한 업데이트 준비

환경에 NFV(네트워크 기능 가상화)가 활성화된 경우 언더클라우드를 업데이트한 후 오버클라우드를 업데이트하기 전에 다음 단계를 따르십시오.

절차

1. 사용자 지정 환경 파일에서 vhost 사용자 소켓 디렉터리를 변경합니다(예: **network-environment.yaml**).

```
parameter_defaults:
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
```

2. **ovs-dpdk-permissions.yaml** 파일을 **openstack overcloud deploy** 명령에 추가하여 OVS-DPDK의 qemu 그룹 설정을 **hugetlbfs** 로 구성합니다.

```
-e environments/ovs-dpdk-permissions.yaml
```

3. 모든 인스턴스의 vHost 사용자 포트가 in **dpdkvhostuserclient** 모드인지 확인합니다. 자세한 내용은 수동으로 vhost 사용자 포트 모드 변경을 참조하십시오.

2.5. OPENSTACK PLATFORM 10.Z의 현재 오버클라우드 이미지 업데이트

Undercloud 업데이트 프로세스는 rhosp-director- **images 및 rhosp-director-images- ipa** 패키지에서 새 이미지 아카이브를 다운로드할 수 있습니다. 이 프로세스는 **Red Hat OpenStack Platform 10** 에서 언더클라우드에서 이러한 이미지를 업데이트합니다.

사전 요구 사항

- 현재 언더클라우드 버전의 최신 마이너 릴리스로 업데이트되었습니다.

절차

1. **yum** 로그를 확인하여 새 이미지 아카이브를 사용할 수 있는지 확인합니다.

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

2. 새 아카이브를 사용할 수 있는 경우 현재 이미지를 새 이미지로 바꿉니다. 새 이미지를 설치하려면 먼저 **stack** 사용자 홈(**/home/stack/images**)의 **images** 디렉터리에서 기존 이미지를 제거합니다.

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

3. 언더클라우드 노드에서 언더클라우드 인증 정보를 가져옵니다.

```
$ source ~/stackrc
```

4. 아카이브를 추출합니다.

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

5. 의 최신 이미지를 director로 가져오고 새 이미지를 사용하도록 노드를 구성합니다.

```
$ cd ~/images
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f csv --quote none | sed "1d" | paste -s -d " ")
```

6. 이미지 업데이트를 완료하려면 새 이미지가 있는지 확인합니다.

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

또한 director는 이전 이미지를 유지하고 업데이트 시의 타임스탬프를 사용하여 이름을 바꿉니다. 이러한 이미지가 더 이상 필요하지 않은 경우 삭제합니다.

director가 최신 이미지를 사용하여 업데이트되고 최신 이미지를 사용합니다. 업데이트 후 서비스를 다시 시작하지 않아도 됩니다.

이제 언더클라우드에서 업데이트된 OpenStack Platform 10 패키지를 사용하고 있습니다. 다음으로 오버클라우드를 최신 마이너 릴리스로 업데이트합니다.

2.6. OPENSTACK PLATFORM 10.Z의 현재 오버클라우드 패키지 업데이트

director는 모든 오버클라우드 노드에서 패키지를 업데이트하는 명령을 제공합니다. 이를 통해 현재 버전의 OpenStack Platform 환경에서 마이너 업데이트를 수행할 수 있습니다. **Red Hat OpenStack Platform 10** 의 마이너 업데이트입니다.

참고

이 단계에서는 Overcloud 노드의 운영 체제를 최신 버전의 Red Hat Enterprise Linux 7 및 Open vSwitch로 업데이트합니다.

사전 요구 사항

- 현재 언더클라우드 버전의 최신 마이너 릴리스로 업데이트되었습니다.
- 오버클라우드 백업을 수행했습니다.

절차

1. **rhel_reg_release** 매개변수의 서브스크립션 관리 구성을 확인합니다. 이 매개변수가 설정되지 않은 경우 포함시키고 버전 7.7을 설정해야 합니다.

```
parameter_defaults:
  ...
  rhel_reg_release: "7.7"
```

   오버클라우드 서브스크립션 관리 환경 파일에 변경 사항을 저장해야 합니다.

2. 원래의 **openstack overcloud deploy** 명령 및 **--update-plan-only** 옵션을 포함하여 현재 계획을 업데이트합니다. 예를 들면 다음과 같습니다.

```
$ openstack overcloud deploy --update-plan-only \
  --templates  \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -e /home/stack/templates/storage-environment.yaml \
  -e /home/stack/templates/rhel-registration/environment-rhel-registration.yaml \
  [-e <environment_file>|...]
```

   --update-plan-only 는 director에 저장된 Overcloud 계획만 업데이트합니다. 오버클라우드와 관련된 환경 파일과 해당 업데이트 경로를 포함하려면 **-e** 옵션을 사용합니다. 후속 환경 파일에 정의된 매개 변수와 리소스가 우선하므로 환경 파일의 순서가 중요합니다. 다음 목록은 환경 파일 순서의 예입니다.

   - heat 템플릿 컬렉션의 초기화 파일(**environments/network-isolation.yaml**)을 포함한 네트워크 분리 파일 및 사용자 정의 NIC 구성 파일입니다.

- 모든 외부 로드 밸런싱 환경 파일.
- 모든 스토리지 환경 파일.
- Red Hat CDN 또는 Satellite 등록의 환경 파일
- 기타 사용자 지정 환경 파일

3. 오버클라우드의 정적 인벤토리 파일을 생성합니다.

```
$ tripleo-ansible-inventory --ansible_ssh_user heat-admin --static-yaml-inventory
~/inventory.yaml
```

오버클라우드 이름을 오버클라우드의 기본 오버클라우드 이름에 다르게 사용하는 경우 **--plan** 옵션을 사용하여 오버클라우드 이름을 **설정합니다**.

4. 모든 노드에서 운영 체제 버전을 Red Hat Enterprise Linux 7.7로 설정하는 작업이 포함된 플레이북을 생성합니다.

```
$ cat > ~/set_release.yaml <<'EOF'
- hosts: all
  gather_facts: false
  tasks:
    - name: set release to 7.7
      command: subscription-manager release --set=7.7
      become: true
EOF
```

5. set_release.yaml 플레이북을 실행합니다.

```
$ ansible-playbook -i ~/inventory.yaml -f 25 ~/set_release.yaml --limit
undercloud,Controller,Compute
```

--limit 옵션을 사용하여 콘텐츠를 모든 Red Hat OpenStack Platform 노드에 적용합니다.

6. **openstack overcloud update 명령을 사용하여 모든 노드에서 패키지 업데이트를** 수행합니다.

```
$ openstack overcloud update stack -i overcloud
```

i는 각 노드를 순차적으로 업데이트하기 위해 대화형 모드를 **실행합니다**. 업데이트 프로세스가 노드 업데이트를 완료하면 스크립트에서 확인할 중단점을 제공합니다. i **옵션을** 사용하지 않으면 업데이트가 첫 번째 중단점에서 일시 중지된 상태로 유지됩니다. 따라서 **-i** 옵션을 포함해야 합니다.

스크립트는 다음 기능을 수행합니다.

a. 스크립트는 노드에서 하나씩 실행됩니다.

   i. 컨트롤러 노드의 경우 전체 패키지 업데이트가 표시됩니다.

   ii. 다른 노드의 경우 Puppet 모듈만 업데이트됩니다.

b. Puppet은 한 번에 모든 노드에서 실행됩니다.

   i. 컨트롤러 노드의 경우 Puppet 실행은 구성을 동기화합니다.

   ii. 다른 노드의 경우 Puppet에서 나머지 패키지를 업데이트하고 구성을 동기화합니다.

7. 업데이트 프로세스가 시작됩니다. 이 프로세스 중에 director는 **IN_PROGRESS** 상태를 보고하고 중단 지점을 지우도록 정기적으로 프롬프트를 표시합니다. 예를 들면 다음과 같습니다.

```
starting package update on stack overcloud
IN_PROGRESS
IN_PROGRESS
WAITING
on_breakpoint: [u'overcloud-compute-0', u'overcloud-controller-2', u'overcloud-controller-1',
u'overcloud-controller-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear 49913767-e2dd-4772-
b648-81e198f5ed00), no=cancel update, C-c=quit interactive mode:
```

Enter를 눌러 **on_breakpoint** 목록의 마지막 노드에서 중단 지점을 지웁니다. 그러면 해당 노드의 업데이트가 시작됩니다.

8. 스크립트는 노드의 업데이트 순서를 자동으로 미리 정의합니다.

   - 개별적으로 각 컨트롤러 노드
   - 개별 컴퓨팅 노드 각각
   - 각 Ceph Storage 노드 개별적으로
   - 다른 모든 노드 개별적으로

   이 주문을 사용하여 업데이트가 성공적으로 수행되도록 특히 다음 명령을 사용하는 것이 좋습니다.

   a. 각 컨트롤러 노드의 중단 지점을 개별적으로 지웁니다. 업데이트 후 노드의 서비스를 다시 시작해야 하는 경우 각 컨트롤러 노드에 개별 패키지 업데이트가 필요합니다. 이로 인해 다른 컨트롤러 노드의 고가용성 서비스로 중단이 줄어듭니다.

   b. 컨트롤러 노드를 업데이트한 후 각 컴퓨팅 노드의 중단 지점을 지웁니다. 컴퓨팅 노드 이름을 입력하여 특정 노드에서 중단 지점을 지우거나 Python 기반 정규 표현식을 사용하여 여러 컴퓨팅 노드에서 한 번에 중단 지점을 지울 수 있습니다.

   c. 각 Ceph Storage 노드의 중단 지점을 지웁니다. Ceph Storage 노드 이름을 입력하여 특정 노드에서 중단 지점을 지우거나 Python 기반 정규 표현식을 사용하여 여러 Ceph Storage 노드에서 한 번에 중단 지점을 지울 수 있습니다.

   d. 나머지 중단 지점을 지웁니다. 나머지 노드를 업데이트합니다. 노드 이름을 입력하여 특정 노드에서 중단 지점을 지우거나 Python 기반 정규 표현식을 사용하여 한 번에 여러 노드에서 중단 지점을 지울 수 있습니다.

   e. 모든 노드가 업데이트가 완료될 때까지 기다립니다.

9. update 명령은 업데이트가 완료되면 **COMPLETE** 상태를 보고합니다.

```
...
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

10. 컨트롤러 노드에 펜싱을 구성한 경우 업데이트 프로세스에서 해당 노드를 비활성화할 수 있습니다. 업데이트 프로세스가 완료되면 컨트롤러 노드 중 하나에서 다음 명령을 사용하여 펜싱을 다시 활성화합니다.

```
$ sudo pcs property set stonith-enabled=true
```

업데이트 프로세스는 오버클라우드의 노드를 자동으로 재부팅하지 않습니다. 커널 및 기타 시스템 패키지를 업데이트하려면 재부팅해야 합니다. 각 노드에서 **/var/log/yum.log** 파일을 확인하여 **kernel** 또는 **openvswitch** 패키지가 해당 주 버전 또는 부 버전을 업데이트했는지 확인합니다. 있는 경우 다음 절차를 사용하여 각 노드를 재부팅합니다.

2.7. 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.

2.8. 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 **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 노드를 재부팅할 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
```

2.9. 컴퓨팅 노드 재부팅

컴퓨팅 노드를 재부팅하려면 다음 워크플로가 포함됩니다.

- 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. 재부팅하려는 컴퓨팅 노드를 확인하려면 모든 컴퓨팅 노드를 나열합니다.

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

3. 오버클라우드에서 컴퓨팅 노드를 선택하여 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 컴퓨팅 노드 간 가상 머신 마이그레이션을 참조하십시오.

6. Compute 노드에 로그인하여 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 활성화되어 있는지 확인합니다.

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

2.10. 시스템 패키지 확인

업그레이드하기 전에 언더클라우드 노드 및 모든 오버클라우드 노드는 다음 패키지의 최신 버전을 사용해야 합니다.

패키지	버전
openvswitch	2.9 이상
qemu-img-rhev	최소 2.10
qemu-kvm-common-rhev	최소 2.10
qemu-kvm-rhev	최소 2.10
qemu-kvm-tools-rhev	최소 2.10

절차

1. 노드에 로그인합니다.

2. **yum** 을 실행하여 시스템 패키지를 확인합니다.

```
$ sudo yum list qemu-img-rhev qemu-kvm-common-rhev qemu-kvm-rhev qemu-kvm-tools-rhev openvswitch
```

3. **ovs-vsctl** 을 실행하여 현재 실행 중인 버전을 확인합니다.

```
$ sudo ovs-vsctl --version
```

4. 모든 노드에 대해 이 단계를 반복합니다.

이제 언더클라우드에서 업데이트된 OpenStack Platform 10 패키지를 사용합니다. 다음 몇 가지 절차에 따라 시스템이 작동 중인지 확인합니다.

2.11. OPENSTACK PLATFORM 10 언더클라우드 검증

다음은 업그레이드하기 전에 **Red Hat OpenStack Platform 10** 언더클라우드의 기능을 확인하는 일련의 단계입니다.

절차

1. 언더클라우드 액세스 세부 정보를 가져옵니다.

```
$ source ~/stackrc
```

2. 실패한 Systemd 서비스를 확인합니다.

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

3. 언더클라우드의 사용 가능한 공간을 확인합니다.

```
$ df -h
```

"Undercloud Requirements(Undercloud 요구 사항)" 를 기준으로 사용하여 사용 가능한 공간이 적절한지 확인합니다.

4. 언더클라우드에 NTP가 설치되어 있으면 클록이 동기화되었는지 확인합니다.

```
$ sudo ntpstat
```

5. 언더클라우드 네트워크 서비스를 확인합니다.

```
$ openstack network agent list
```

모든 에이전트는 **Alive** 이고 해당 상태는 **UP** 이어야 합니다.

6. 언더클라우드 계산 서비스를 확인합니다.

```
$ openstack compute service list
```

모든 에이전트의 상태가 **활성화되고** 상태가 **up**이어야 합니다

관련 정보

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

2.12. OPENSTACK PLATFORM 10 오버클라우드 검증

다음은 업그레이드하기 전에 **Red Hat OpenStack Platform 10** 오버클라우드의 기능을 확인하는 일련의 단계입니다.

절차

1. 언더클라우드 액세스 세부 정보를 가져옵니다.

```
$ source ~/stackrc
```

2. 베어 메탈 노드의 상태를 확인합니다.

```
$ openstack baremetal node list
```

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

3. 실패한 Systemd 서비스를 확인합니다.

```
$ 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. 모든 서비스에 대한 HAProxy 연결을 확인합니다. **haproxy.stats** 서비스에 대한 컨트롤 플레인 VIP 주소 및 인증 정보를 가져옵니다.

```
$ 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 /etc/haproxy/haproxy.cfg'
```

5. 이전 단계에서 얻은 연결 및 인증 정보를 사용하여 RHOSP 서비스의 연결 상태를 확인합니다. SSL이 활성화되지 않은 경우 다음 cURL 요청에서 다음 세부 정보를 사용합니다.

```
$ curl -s -u admin:<PASSWORD> "http://<IP ADDRESS>:1993/;csv" | egrep -vi "
(frontend|backend)" | awk -F',' '{ print $1" "$2" "$18 }'
```

SSL이 활성화된 경우 다음 cURL 요청에서 다음 세부 정보를 사용합니다.

```
curl -s -u admin:<PASSWORD> "https://<HOSTNAME>:1993/;csv" | egrep -vi "
(frontend|backend)" | awk -F',' '{ print $1" "$2" "$18 }'
```

< **PASSWORD** > 및 < **IP ADDRESS** > 또는 < **HOSTNAME** > 값을 **haproxy.stats** 서비스의 각 정보로 바꿉니다. 결과 목록은 각 노드의 OpenStack Platform 서비스와 해당 연결 상태를 표시합니다.

6. 오버클라우드 데이터베이스 복제 상태를 확인합니다.

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

7. RabbitMQ 클러스터 상태를 확인합니다.

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

8. Pacemaker 리소스 상태를 확인합니다.

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

다음 항목 검색:

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

9. 각 오버클라우드 노드의 디스크 공간을 확인합니다.

```
$ 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. Overcloud Ceph Storage 클러스터 상태를 확인합니다. 다음 명령은 컨트롤러 노드에서 **ceph** 툴을 실행하여 클러스터를 확인합니다.

```
$ 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 공간이 있는지 확인합니다. 다음 명령은 컨트롤러 노드에서 **ceph** 툴을 실행하여 사용 가능한 공간을 확인합니다.

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

중요

각 Ceph 개체 스토리지 데몬(OSD)에 대한 PG(배치 그룹) 수는 기본적으로 250개를 초과해서는 안 됩니다. OSD당 더 많은 PG로 Ceph 노드를 업그레이드하면 경고 상태가 되며 업그레이드 프로세스가 실패할 수 있습니다. 업그레이드 프로세스를 시작하기 전에 OSD당 PG 수를 늘릴 수 있습니다. 이 문제를 진단하고 해결하는 방법에 대한 자세한 내용은 하나 이상의 OSD에 할당된 Ceph PG가 250보다 높기 때문에 OpenStack FFU 문서를 10에서 13번으로 참조하십시오.

12. 오버클라우드 노드에서 클록이 동기화되었는지 확인합니다.

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

13. 오버클라우드 액세스 세부 정보를 소싱합니다.

```
$ source ~/overcloudrc
```

14. 오버클라우드 네트워크 서비스를 확인합니다.

```
$ openstack network agent list
```

모든 에이전트는 **Alive** 이고 해당 상태는 **UP** 이어야 합니다.

15. 오버클라우드 계산 서비스를 확인합니다.

```
$ openstack compute service list
```

모든 에이전트의 상태가 **활성화되고** 상태가 **up**이어야 합니다

16. 오버클라우드 볼륨 서비스를 확인합니다.

```
$ openstack volume service list
```

모든 에이전트의 상태가 **활성화되고** 상태가 **up** 이어야 합니다.

관련 정보

- "Red Hat 권장 구성으로 OpenStack 환경이 배포되었는지 어떻게 확인할 수 있습니까?"라는 문서를 검토하십시오. 이 문서에서는 Red Hat OpenStack Platform 환경을 확인하고 Red Hat의 권장 사항에 맞게 구성을 조정하는 방법에 대해 설명합니다.
- "데이터베이스 크기 관리 Red Hat Enterprise Linux OpenStack Platform에 대한 데이터베이스 관리" 문서를 검토하여 Overcloud에서 OpenStack Platform 서비스에 대해 사용하지 않는 데이터베이스 레코드를 확인하고 정리합니다.

2.13. NFV 지원 환경의 업데이트 완료

환경에 NFV(네트워크 기능 가상화)가 활성화된 경우 언더클라우드 및 오버클라우드를 업데이트한 후 다음 단계를 수행해야 합니다.

절차

기존 OVS-DPDK 인스턴스를 마이그레이션하여 vhost 소켓 모드가 OVS 포트의 **dkdpvhostuser** client 모드에서 **dkdpvhostuserclient** 모드로 변경되었는지 확인해야 합니다. 기존 인스턴스를 스냅숏하고 해당 스냅샷 이미지를 기반으로 새 인스턴스를 다시 빌드하는 것이 좋습니다. 인스턴스 스냅샷에 대한 자세한 내용은 인스턴스 스냅샷 관리를 참조하십시오.

인스턴스를 스냅샷하고 스냅숏에서 새 인스턴스를 부팅하려면 다음을 수행합니다.

1. 오버클라우드 액세스 세부 정보를 소싱합니다.

```
$ source ~/overcloudrc
```

2. 스냅샷을 만들 인스턴스의 서버 ID를 찾습니다.

```
$ openstack server list
```

3. 스냅샷을 생성하기 전에 소스 인스턴스를 종료하여 모든 데이터가 디스크로 플러시되었는지 확인합니다.

```
$ openstack server stop SERVER_ID
```

4. 인스턴스의 스냅샷 이미지를 생성합니다.

```
$ openstack image create --id SERVER_ID SNAPSHOT_NAME
```

5. 이 스냅샷 이미지를 사용하여 새 인스턴스를 부팅합니다.

```
$ openstack server create --flavor DPDK_FLAVOR --nic net-id=DPDK_NET_ID--image
SNAPSHOT_NAME INSTANCE_NAME
```

6. 필요한 경우 새 인스턴스 상태가 **ACTIVE인지 확인합니다**.

```
$ openstack server list
```

스냅샷 및 다시 시작하는 데 필요한 모든 인스턴스에 대해 이 절차를 반복합니다.

2.14. YUM 이력 유지

Overcloud의 마이너 업데이트를 완료한 후 **yum 기록을** 유지합니다. 이 정보는 가능한 롤백 작업에 대해 yum 트랜잭션을 실행 취소해야 하는 경우에 유용합니다.

절차

1. 각 노드에서 다음 명령을 실행하여 노드의 전체 **yum** 기록을 파일에 저장합니다.

```
$ sudo yum history list all > /home/heat-admin/$(hostname)-yum-history-all
```

2. 각 노드에서 다음 명령을 실행하여 마지막 **yum** 히스토리 항목의 ID를 저장합니다.

```
$ sudo yum history list all | head -n 5 | tail -n 1 | awk '{print $1}' > /home/heat-
admin/$(hostname)-yum-history-all-last-id
```

3. 이러한 파일을 안전한 위치에 복사합니다.

2.15. 다음 단계

준비 단계가 완료되면 3장. *언더클라우드 업그레이드* 의 단계를 사용하여 언더클라우드를 10에서 13으로 업그레이드할 수 있습니다.

3장. 언더클라우드 업그레이드

다음 절차에서는 언더클라우드를 **Red Hat OpenStack Platform 13으로 업그레이드합니다**. 이를 위해 OpenStack Platform 10에서 OpenStack Platform 13으로의 후속 버전의 언더클라우드를 통해 업그레이드를 수행합니다.

3.1. 언더클라우드를 OPENSTACK PLATFORM 11으로 업그레이드

다음 절차에서는 언더클라우드 툴 세트 및 코어 Heat 템플릿 컬렉션을 **OpenStack Platform 11** 릴리스로 업그레이드합니다.

절차

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

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
```

3. 새 OpenStack Platform 리포지토리를 활성화합니다.

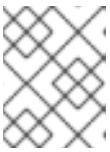
```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
```

4. 오버클라우드 기본 이미지에 대한 업데이트를 비활성화합니다.

```
$ sudo yum-config-manager --setopt=exclude=rhosp-director-images* --save
```

5. 기본 OpenStack Platform 서비스를 중지합니다.

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```



참고

이로 인해 언더클라우드에 대해 짧은 다운타임이 발생합니다. Undercloud 업그레이드 중에 Overcloud가 계속 작동합니다.

6. 기본 프로비저닝/제어 플레인 네트워크가 **192.0.2.0/24** 에서 **192.168.24.0/24** 로 변경되었습니다. 이전 **undercloud.conf** 파일에서 기본 네트워크 값을 사용한 경우 프로비저닝/컨트롤 플레인 네트워크가 **192.0.2.0/24** 로 설정됩니다. 즉, **192.0.2.0/24** 네트워크를 계속 사용하려면 **undercloud.conf** 파일에서 특정 매개변수를 설정해야 합니다. 이러한 매개변수는 다음과 같습니다.
   - **local_ip**
   - **network_gateway**
   - **undercloud_public_vip**
   - **undercloud_admin_vip**
   - **network_cidr**
   - **masquerade_network**

- **dhcp_start**
- **dhcp_end**

향후 업그레이드 중에 **192 .0.2.0/24 CIDR을 계속 사용하도록 undercloud. conf** 에서 네트워크 값을 설정합니다. **openstack undercloud upgrade** 명령을 실행하기 전에 네트워크 구성이 올바르게 설정되었는지 확인합니다.

7. **yum** 을 실행하여 director의 기본 패키지를 업그레이드합니다.

```
$ sudo yum update -y instack-undercloud openstack-puppet-modules openstack-tripleo-common python-tripleoclient
```

8. 다음 명령을 실행하여 언더클라우드를 업그레이드합니다.

```
$ openstack undercloud upgrade
```

9. 언더클라우드 업그레이드 프로세스가 완료될 때까지 기다립니다.

언더클라우드를 **OpenStack Platform 11** 릴리스로 업그레이드했습니다.

3.2. 언더클라우드를 OPENSTACK PLATFORM 12로 업그레이드

다음 절차에서는 언더클라우드 툴 세트 및 코어 Heat 템플릿 컬렉션을 **OpenStack Platform 12** 릴리스로 업그레이드합니다.

절차

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

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-11-rpms
```

3. 새 OpenStack Platform 리포지토리를 활성화합니다.

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

4. 오버클라우드 기본 이미지에 대한 업데이트를 비활성화합니다.

```
$ sudo yum-config-manager --setopt=exclude=rhosp-director-images* --save
```

5. **yum** 을 실행하여 director의 기본 패키지를 업그레이드합니다.

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

6. **/home/stack/undercloud.conf** 파일을 편집하고 **enabled_drivers 매개 변수에 pxe_ ssh** 드라이버가 포함되어 있지 않은지 확인합니다. 이 드라이버는 더 이상 사용되지 않으며 Red Hat OpenStack Platform에서 제거된 VBMC(Virtual Baseboard Management Controller)를 사용합니다. 이 새로운 드라이버 및 마이그레이션 지침에 대한 자세한 내용은 *Director 설치 및 사용 가이드의 부록 "Virtual Baseboard Management Controller (Virtual Baseboard Management Controller)" 를 참조하십시오.*

7. 다음 명령을 실행하여 언더클라우드를 업그레이드합니다.

```
$ openstack undercloud upgrade
```

8. 언더클라우드 업그레이드 프로세스가 완료될 때까지 기다립니다.

언더클라우드를 **OpenStack Platform 12** 릴리스로 업그레이드했습니다.

3.3. 언더클라우드를 OPENSTACK PLATFORM 13으로 업그레이드

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

절차

1. **stack** 사용자로 director에 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 기본 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.

3.4. 언더클라우드에서 더 이상 사용되지 않는 서비스 비활성화

언더클라우드를 업그레이드한 후 더 이상 사용되지 않는 **openstack-glance-registry** 및 **mongod** 서비스를 비활성화해야 합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. **openstack-glance-registry** 서비스를 중지하고 비활성화합니다.

```
$ sudo systemctl stop openstack-glance-registry
$ sudo systemctl disable openstack-glance-registry
```

3. **mongod** 서비스를 중지하고 비활성화합니다.

```
$ sudo systemctl stop mongod
$ sudo systemctl disable mongod
```

3.5. 다음 단계

언더클라우드 업그레이드가 완료되었습니다. 이제 컨테이너 이미지의 소스를 구성할 수 있습니다.

4장. 컨테이너 이미지 소스 구성

컨테이너화된 오버클라우드는 필수 컨테이너 이미지가 있는 레지스트리에 액세스해야 합니다. 이 장에서는 Red Hat OpenStack Platform의 컨테이너 이미지를 사용하도록 레지스트리 및 오버클라우드 구성을 준비하는 방법에 대해 설명합니다.

이 가이드에서는 레지스트리를 사용하도록 오버클라우드를 구성하는 몇 가지 사용 사례를 제공합니다. 이러한 사용 사례 중 하나를 사용하기 전에 image prepare 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 메서드는 초기 구성을 생성하는 데 가장 쉽습니다. 그러나 각 오버클라우드 노드는 Red Hat Container Catalog에서 각 이미지를 직접 가져와 네트워크 정체 및 느린 배포가 발생할 수 있습니다. 또한 모든 오버클라우드 노드에는 Red Hat Container Catalog에 대한 인터넷 액세스가 필요합니다.

로컬 레지스트리

Undercloud는 **docker-distribution** 서비스를 사용하여 레지스트리 역할을 합니다. 이를 통해 director는 **registry.redhat.io** 의 이미지를 동기화하고 **docker-distribution** 레지스트리로 푸시할 수 있습니다. Overcloud를 생성할 때 Overcloud가 언더클라우드의 **docker-distribution** 레지스트리에서 컨테이너 이미지를 가져옵니다. 이 방법을 사용하면 배포를 가속화하고 네트워크 정체를 줄일 수 있는 레지스트리를 내부적으로 저장할 수 있습니다. 그러나 언더클라우드는 기본 레지스트리로만 작동하며 컨테이너 이미지에 대한 제한된 라이프사이클 관리를 제공합니다.

참고

docker-distribution 서비스는 **docker** 와 별도로 작동합니다.**docker** 는 **docker-distribution** 레지스트리로 이미지를 가져오고 내보내는 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 솔루션이 제공됩니다.

목록에서 메서드를 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 개념 정보를 포함하여 **openstack overcloud container image prepare** 명령을 사용하는 방법에 대해 설명합니다.

오버클라우드의 컨테이너 이미지 환경 파일 생성

openstack overcloud container prepare 명령의 주요 용도 중 하나는 Overcloud에서 사용하는 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 기능에 다음 옵션을 사용합니다.

--output-env-file

결과 환경 파일 이름을 정의합니다.

다음 코드 조각은 이 파일의 콘텐츠의 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 방법에 대한 컨테이너 이미지 목록 생성

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

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

--output-images-file

가져오기 목록에 대한 결과 파일 이름을 정의합니다.

다음은 이 파일의 콘텐츠의 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 결과 이미지 위치를 생성하는 네임스페이스가 필요합니다. **openstack overcloud container image prepare** 명령에서는 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][이미지 이름]**

컨테이너 이미지 태그 설정

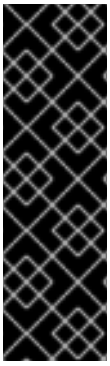
--tag 및 **--tag-from-label** 옵션을 함께 사용하여 각 컨테이너 이미지의 태그를 설정합니다.

--tag

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

--tag-from-label

지정된 컨테이너 이미지 레이블의 값을 사용하여 모든 이미지에 대해 버전이 지정된 태그를 검색하고 가져옵니다. director는 **--tag** 에 대해 설정한 값으로 태그가 지정된 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 **버전** 및 **릴리스** 레이블을 사용하여 새 태그를 구성합니다. 한 컨테이너의 경우 **version** 을 **13.0** 으로 설정하고 **릴리스가 34** 로 설정될 수 있으므로 태그 **13.0-34** 가 발생할 수 있습니다.



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 라벨로 저장합니다. 이 버전 형식을 사용하면 한 **{release}** 에서 다음 버전으로 쉽게 업데이트할 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 director는 컨테이너 이미지를 업데이트하기 위해 태그를 변경해야 하므로 업데이트를 수행할 때 **--tag latest** 를 자체적으로 사용하면 문제가 발생합니다.

4.3. 추가 서비스의 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 활성화하는 환경 파일을 포함합니다.

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

Service	환경 파일
Ceph Storage	**environments/ceph-ansible/ceph-ansible.yaml**
collectd	**environments/services-docker/collectd.yaml**
마케도니아	**environments/services-docker/congress.yaml**
fluentd	**environments/services-docker/fluentd.yaml**
OpenStack Bare Metal(ironic)	**environments/services-docker/ironic.yaml**

Service	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** 알림: 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 몇 섹션에서는 추가 서비스를 포함하는 예를 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 이미지를 준비하려면 director에서 이러한 서비스가 활성화되어 있음을 알아야 합니다.

이 환경 파일 외에도 OpenStack Platform 서비스와 다른 Ceph Storage 컨테이너 위치도 정의해야 합니다. Ceph Storage와 관련된 다음 매개 변수를 설정하려면 **--set** 옵션을 사용합니다.

--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 태그부터 버전이 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 director에서 이미지를 준비할 수 있도록 **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml** 환경 파일을 포함해야 합니다. 다음 코드 조각은 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)를 배포하는 경우 director에서 이미지를 준비할 수 있도록 **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/sahara.yaml** 환경 파일을 포함해야 합니다. 다음 코드 조각은 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 director에서 이미지를 준비할 수 있도록 **/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 정의 역할 파일을 포함하려면 **-r** 옵션도 사용해야 합니다. 다음 코드 조각은 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 코드 조각은 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 백엔드를 사용하여 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 서비스용 NetApp 백엔드 가이드에서 *NetApp 백엔드를 사용하여 공유 파일 시스템 서비스배포*
- 공유 파일 시스템 서비스용 CephFS 백엔드 가이드에서 *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. 레지스트리가 이미 구성되어 있고 필요한 모든 것이 가져올 이미지의 URL과 네임스페이스이므로 원격 레지스트리에서 이미지를 가져오는 것이 가장 간단한 방법입니다. 그러나 오버클라우드를 생성하는 동안 오버클라우드 노드는 모두 원격 리포지토리에서 이미지를 가져와 외부 연결을 제한할 수 있습니다. 따라서 이 방법은 프로덕션 환경에 권장되지 않습니다. 프로덕션 환경의 경우 다음 방법 중 하나를 사용하십시오.

- 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.

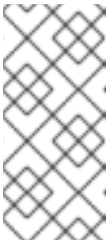
```
(undercloud) $ sudo openstack overcloud container image prepare \
  --namespace=registry.redhat.io/rhosp13 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

   - 선택적 서비스에 대한 환경 파일을 포함하려면 **-e** 옵션을 사용합니다.
   - r **옵션을** 사용하여 사용자 지정 역할 파일을 포함합니다.
   - Ceph Storage를 사용하는 경우 Ceph Storage 컨테이너 이미지 위치를 정의하는 추가 매개변수를 포함합니다. **--set ceph_namespace,--set ceph_image,--set ceph_tag**.

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 외부 네트워크 연결을 혼잡하지 않고 배포 프로세스를 가속화할 수 있는 내부 네트워크 내에서 컨테이너 이미지의 네트워크 트래픽을 유지합니다.

절차

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

```
<REGISTRY_IP_ADDRESS>:8787
```

이전에 undercloud. **conf 파일의 local_ip 매개변수로 설정한 언더클라우드** 의 IP 주소를 사용합니다. 아래 명령의 경우 주소는 **192.168.24.1:8787** 로 간주됩니다.

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
```

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

4. 다음 두 파일이 생성되었는지 확인합니다.
   - 원격 소스의 컨테이너 이미지 정보가 포함된 **local_registry_images.yaml** 이 파일을 사용하여 Red Hat Container Registry(**registry.redhat.io**)에서 언더클라우드로 이미지를 가져옵니다.
   - **overcloud_images.yaml** - 언더클라우드의 최종 이미지 위치가 포함됩니다. 배포와 함께 이 파일을 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 디스크의 속도에 따라 필요한 이미지를 가져오는 데 다소 시간이 걸릴 수 있습니다.

참고

컨테이너 이미지는 약 10GB의 디스크 공간을 사용합니다.

6. 이미지가 이제 Undercloud의 **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
```

태그가 지정된 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. 로컬 레지스트리로 이미지를 가져올 템플릿을 생성합니다.

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

   - 선택적 서비스에 대한 환경 파일을 포함하려면 **-e** 옵션을 사용합니다.
   - r **옵션을** 사용하여 사용자 지정 역할 파일을 포함합니다.
   - Ceph Storage를 사용하는 경우 Ceph Storage 컨테이너 이미지 위치를 정의하는 추가 매개변수를 포함합니다. **--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag**.

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 값이 사용됩니다.

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

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

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

   그러면 Satellite 서버로 가져온 이미지 목록이 제공됩니다.

4. **satellite_images_names** 파일을 Satellite 6 **hammer** 툴이 포함된 시스템으로 복사합니다. 또는 *Hammer CLI 가이드*의 지침에 따라 **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 통합할 수 있습니다.

10. **base** 이미지에 사용할 수 있는 태그를 확인합니다.

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

OpenStack Platform 컨테이너 이미지에 대한 태그가 표시됩니다.

11. 언더클라우드로 돌아가서 Satellite 서버에서 이미지에 대한 환경 파일을 생성합니다. 다음은 환경 파일을 생성하기 위한 예제 명령입니다.

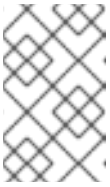
```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=satellite6.example.com:5000 \
  --prefix=acme-osp13_containers- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

참고

이 버전의 **openstack overcloud container prepare** 명령은 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. 예를 들면 **acme-production-myosp13-osp13_containers-** 입니다.
  - 콘텐츠 뷰를 사용하지 않는 경우 구조는 **[org]-[product]-**입니다. 예를 들면 **acme-osp13_containers-** 입니다.
- **--tag-from-label {version}-{release}** - 각 이미지의 최신 태그를 식별합니다.
- **-e** - 옵션 서비스의 환경 파일을 포함합니다.
- **-R -** 사용자 지정 역할 파일을 포함합니다.
- **--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag** - Ceph Storage를 사용하는 경우 추가 매개 변수를 포함하여 Ceph Storage 컨테이너 이미지 위치를 정의합니다. 이제 **ceph_image**에는 Satellite별 접두사가 포함됩니다. 이 접두사는 **--prefix** 옵션과 동일한 값입니다. 예를 들면 다음과 같습니다.

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

이렇게 하면 오버클라우드에서 Satellite 명명 규칙을 사용하여 Ceph 컨테이너 이미지를 사용합니다.

12. **overcloud_images.yaml** 파일에는 Satellite 서버에 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 작업을 통해 이 파일을 포함합니다.

이제 업그레이드할 오버클라우드를 준비할 수 있습니다.

5장. 오버클라우드 업그레이드 준비

이 섹션에서는 업그레이드 프로세스를 위해 오버클라우드를 준비합니다. 이 섹션의 모든 단계가 오버클라우드에 적용되는 것은 아닙니다. 그러나 각 단계를 단계별로 진행하고 업그레이드 프로세스를 시작하기 전에 오버클라우드에 추가 구성이 필요한지 확인하는 것이 좋습니다.

5.1. 오버클라우드 서비스 다운타임 준비

오버클라우드 업그레이드 프로세스에서는 주요 지점에서 기본 서비스를 비활성화합니다. 즉, 업그레이드 기간 동안 오버클라우드 서비스를 사용하여 새 리소스를 생성할 수 없습니다. 오버클라우드에서 실행되는 워크로드는 이 기간 동안 활성 상태로 유지됩니다. 즉, 업그레이드 기간을 통해 인스턴스가 계속 실행됩니다.

업그레이드 기간 동안 사용자가 Overcloud 서비스에 액세스할 수 없도록 유지 관리 기간을 계획하는 것이 중요합니다.

오버클라우드 업그레이드의 영향을 받습니다.

- OpenStack Platform 서비스

오버클라우드 업그레이드의 영향을 받지 않음

- 업그레이드 중에 실행되는 인스턴스
- Ceph Storage OSD(인스턴스용 백엔드 스토리지)
- Linux 네트워킹
- Open vSwitch 네트워킹
- 언더클라우드

5.2. 업그레이드 테스트용 컴퓨팅 노드 선택

오버클라우드 업그레이드 프로세스를 통해 다음 중 하나를 수행할 수 있습니다.

- 역할의 모든 노드 업그레이드
- 개별 노드 별

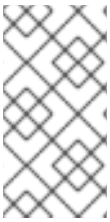
오버클라우드 업그레이드 프로세스를 원활하게 수행하려면 모든 컴퓨팅 노드를 업그레이드하기 전에 사용자 환경의 몇 가지 개별 컴퓨팅 노드에서 업그레이드를 테스트하는 것이 좋습니다. 이렇게 하면 워크로드에 대한 다운타임을 최소화하면서 업그레이드하는 동안 주요 문제가 발생하지 않습니다.

다음 권장 사항을 사용하여 업그레이드할 테스트 노드를 선택하는 데 도움이 됩니다.

- 업그레이드 테스팅을 위해 두 개 또는 세 개의 컴퓨팅 노드 선택
- 중요한 인스턴스가 실행되지 않은 노드 선택
- 필요한 경우 선택한 테스트 컴퓨팅 노드에서 다른 컴퓨팅 노드로 중요한 인스턴스를 마이그레이션합니다.

6장. *오버클라우드 업그레이드* 의 지침은 모든 컴퓨팅 노드에서 업그레이드를 실행하기 전에 **compute-0** 을 컴퓨팅 노드의 예로 사용하여 업그레이드 프로세스를 테스트합니다.

다음 단계에서는 **roles_data** 파일을 업데이트하여 새로 구성 가능한 서비스가 환경의 관련 역할에 추가되었는지 확인합니다. 기존 **roles_data** 파일을 수동으로 편집하려면 OpenStack Platform 13 역할에 대해 다음 구성 가능 서비스 목록을 사용합니다.



참고

Red Hat OpenStack Platform 12 이하에서 Compute 인스턴스의 고가용성(인스턴스 HA)을 활성화하고 버전 13 이상으로 빠른 업그레이드를 수행하려면 먼저 Instance Ha를 수동으로 비활성화해야 합니다. 자세한 내용은 이전 버전에서 인스턴스 HA 비활성화 를 참조하십시오.

5.3. 새로운 구성 가능 서비스

이 버전의 Red Hat OpenStack Platform에는 새로운 구성 가능 서비스가 포함되어 있습니다. 고유한 역할과 함께 사용자 지정 **roles_data** 파일을 사용하는 경우 해당 역할에 이러한 새 필수 서비스를 포함합니다.

모든 역할

다음과 같은 새 서비스가 모든 역할에 적용됩니다.

OS::TripleO::Services::MySQLClient

다른 구성 가능 서비스에 대한 데이터베이스 구성을 제공하는 노드에 MariaDB 클라이언트를 구성합니다. 독립 실행형 구성 가능 서비스가 있는 모든 역할에 이 서비스를 추가합니다.

OS::TripleO::Services::CertmongerUser

오버클라우드에서 Certmonger의 인증서가 필요하도록 허용합니다. TLS/SSL 통신을 활성화하는 경우에만 사용됩니다.

OS::TripleO::Services::Docker

컨테이너화된 서비스를 관리하기 위해 **docker** 를 설치합니다.

OS::TripleO::Services::ContainersLogrotateCrond

컨테이너 로그에 대한 **logrotate** 서비스를 설치합니다.

OS::TripleO::Services::Securetty

노드에서 **securetty** 를 구성할 수 있습니다. **environments/securetty.yaml** 환경 파일을 사용하여 활성화합니다.

OS::TripleO::Services::Tuned

Linux 튜닝 데몬(tuned)을 활성화하고 구성합니다**.**

OS::TripleO::Services::AuditD

auditd 데몬을 추가하고 규칙을 구성합니다. 기본적으로 비활성되어 있습니다.

OS::TripleO::Services::Collectd

collectd 데몬 추가. 기본적으로 비활성되어 있습니다.

OS::TripleO::Services::Rhsm

Ansible 기반 방법을 사용하여 서브스크립션을 구성합니다. 기본적으로 비활성되어 있습니다.

OS::TripleO::Services::RsyslogSidecar

로깅을 위한 사이드카 컨테이너를 구성합니다. 기본적으로 비활성되어 있습니다.

특정 역할

다음과 같은 새 서비스가 특정 역할에 적용됩니다.

OS::TripleO::Services::NovaPlacement

OpenStack Compute(nova) 배치 API를 구성합니다. 현재 오버클라우드에서 독립 실행형 Nova API 역할을 사용하는 경우 이 서비스를 역할에 추가합니다. 그렇지 않으면 서비스를 Controller 역할에 추가합니다.

OS::TripleO::Services::PankoApi

OpenStack Telemetry Event Storage(panko) 서비스를 구성합니다. 현재 오버클라우드에서 독립 실행형 Telemetry 역할을 사용하는 경우 이 서비스를 역할에 추가합니다. 그렇지 않으면 서비스를 Controller 역할에 추가합니다.

OS::TripleO::Services::Clustercheck

컨트롤러 또는 독립 실행형 **데이터베이스** 역할과 같은 **OS::TripleO::Services::MySQL** 서비스도 사용하는 역할에 필요합니다.

OS::TripleO::Services::Iscsid

Controller,**Compute** 및 **BlockStorage** 역할에서 **iscsid** 서비스를 구성합니다.

OS::TripleO::Services::NovaMigrationTarget

컴퓨팅 노드에서 마이그레이션 대상 서비스를 구성합니다.

OS::TripleO::Services::Ec2Api

컨트롤러 노드에서 OpenStack Compute(nova) EC2-API 서비스를 활성화합니다. 기본적으로 비활성되어 있습니다.

OS::TripleO::Services::CephMgr

컨트롤러 노드에서 Ceph Manager 서비스를 활성화합니다. **ceph-anible** 구성의 일부로 활성화됩니다.

OS::TripleO::Services::CephMds

컨트롤러 노드에서 Ceph 메타데이터 서비스(MDS)를 활성화합니다. 기본적으로 비활성되어 있습니다.

OS::TripleO::Services::CephRbdMirror

RADOS 블록 장치(RBD) 미러링 서비스를 활성화합니다. 기본적으로 비활성되어 있습니다.

또한 " 서비스 아키텍처를 참조하십시오. 독립 실행형 역할" 특정 사용자 지정 역할에 대한 업데이트된 서비스 목록에 대한 *Advanced Overcloud Customization* 가이드의 섹션입니다.

새로운 구성 가능 서비스 외에도 OpenStack Platform 13 이후의 더 이상 사용되지 않는 서비스를 기록해 두십시오.

5.4. 사용되지 않는 구성 가능 서비스

사용자 지정 **roles_data** 파일을 사용하는 경우 해당 역할에서 해당 서비스를 제거합니다.

OS::TripleO::Services::Core

이 서비스는 다른 Pacemaker 서비스의 핵심 종속성으로 작동합니다. 고가용성 구성 가능 서비스를 수용하도록 이 서비스가 제거되었습니다.

OS::TripleO::Services::VipHosts

이 서비스는 노드 호스트 이름과 IP 주소를 사용하여 /etc/hosts 파일을 구성했습니다. 이제 이 서비스가 director의 Heat 템플릿에 직접 통합되었습니다.

OS::TripleO::Services::FluentdClient

이 서비스는 **OS::TripleO::Services::Fluentd** 서비스로 교체되었습니다.

OS::TripleO::Services::ManilaBackendGeneric

Manila 일반 백엔드는 더 이상 지원되지 않습니다.

사용자 지정 **roles_data** 파일을 사용하는 경우 해당 서비스에서 해당 역할을 제거합니다.

또한 " 서비스 아키텍처를 참조하십시오. 독립 실행형 역할" 특정 사용자 지정 역할에 대한 업데이트된 서비스 목록에 대한 *Advanced Overcloud Customization* 가이드의 섹션입니다.

5.5. 컨테이너화된 서비스로 전환

빠른 전달 업그레이드 프로세스는 특정 Systemd 서비스를 컨테이너화된 서비스로 변환합니다. 이 프로세스는 **/usr/share/openstack-tripleo-heat-templates/environments/** 의 기본 환경 파일을 사용하는 경우 자동으로 수행됩니다.

사용자 지정 환경 파일을 사용하여 오버클라우드에서 서비스를 활성화하는 경우, 환경 파일에서 **resource_registry** 섹션의 환경 파일을 확인하고 등록된 구성 가능한 서비스가 구성 가능한 서비스에 매핑되는지 확인합니다.

절차

1. 사용자 정의 환경 파일을 확인합니다.

```
$ cat ~/templates/custom_environment.yaml
```

2. 파일 콘텐츠에서 **resource_registry** 섹션이 있는지 확인합니다.

3. **resource_registry** 섹션에서 구성 가능한 서비스를 확인합니다. 다음 네임스페이스를 사용하여 구성 가능한 서비스를 구성합니다.

```
OS::TripleO::Services
```

예를 들어 다음 구성 가능 서비스는 OpenStack Bare Metal Service(ironic) API용입니다.

```
OS::TripleO::Services::IronicApi
```

4. 구성 가능 서비스가 Puppet별 Heat 템플릿에 매핑되는지 확인합니다. 예를 들면 다음과 같습니다.

```
resource_registry:
  OS::TripleO::Services::IronicApi: /usr/share/openstack-triple-heat-
template/puppet/services/ironic-api.yaml
```

5. 컨테이너화된 Heat 템플릿 버전이 **/usr/share/openstack-triple-heat-template/docker/services/** 에 있는지 확인하고 서비스를 컨테이너화된 버전으로 다시 매핑합니다.

```
resource_registry:
  OS::TripleO::Services::IronicApi: /usr/share/openstack-triple-heat-
template/docker/services/ironic-api.yaml
```

또는 **/usr/share/openstack-tripleo-heat-templates/environments/** 에 있는 서비스에 대해 업데이트된 환경 파일을 사용합니다. 예를 들어 OpenStack Bare Metal Service(ironic)를 활성화하기 위한 최신 환경 파일은 컨테이너화된 서비스 매핑을 포함하는 **/usr/share/openstack-tripleo-heat-templates/environments/services/ironic.yaml** 입니다.

사용자 지정 서비스에서 컨테이너화된 서비스를 사용하지 않는 경우 Puppet별 Heat 템플릿에 대한 매핑을 유지합니다.

5.6. 더 이상 사용되지 않는 매개변수

다음 매개 변수는 더 이상 사용되지 않으며 교체되었습니다.

이전 매개변수	새 매개변수
KeystoneNotificationDriver	**NotificationDriver**
controllerExtraConfig	**ControllerExtraConfig**
OvercloudControlFlavor	**OvercloudControllerFlavor**
controllerImage	**ControllerImage**
NovaImage	**ComputeImage**
NovaComputeExtraConfig	**ComputeExtraConfig**
NovaComputeServerMetadata	**ComputeServerMetadata**

NovaComputeSchedulerHints	**ComputeSchedulerHints**
	
	참고
	
	사용자 지정 Compute 역할을 사용하는 경우 역할별 **ComputeSchedulerHints** 를 사용하려면 더 이상 사용되지 않는 **NovaComputeSchedulerHints** 매개변수가 정의되지 않은지 확인하려면 환경에 다음 구성을 추가해야 합니다.
	
	`parameter_defaults:`
	`  NovaComputeSchedulerHints: {}`
	
	사용자 지정 역할을 사용할 때 역할별 **_ROLE_SchedulerHints** 매개변수를 사용하려면 이 구성을 추가해야 합니다.

NovaComputeIPs	**ComputeIPs**
SwiftStorageServerMetadata	**ObjectStorageServerMetadata**
SwiftStorageIPs	**ObjectStorageIPs**
SwiftStorageImage	**ObjectStorageImage**
OvercloudSwiftStorageFlavor	**OvercloudObjectStorageFlavor**
NeutronDpdkCoreList	**OvsPmdCoreList**

이전 매개변수	새 매개변수
NeutronDpdkMemoryChannels	**OvsDpdkMemoryChannels**
NeutronDpdkSocketMemory	**OvsDpdkSocketMemory**
NeutronDpdkDriverType	**OvsDpdkDriverType**
HostCpusList	**OvsDpdkCoreList**

새 매개변수 값의 경우 다음 예와 같이 중첩된 단일 따옴표 표시 없이 겹따옴표를 사용합니다.

값이 있는 이전 매개변수	값을 사용하는 새 매개변수
NeutronDpdkCoreList: "'2,3'"	**OvsPmdCoreList: "2,3"**
HostCpusList: "'0,1'"	**OvsDpdkCoreList: "0,1"**

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

NeutronL3HA

L3 고가용성은 분산 가상 라우팅(Neutron**EnableDVR**)이 있는 구성을 제외하고 모든 경우에 활성화되어 있습니다.

Ceilometer 작업자

Ceilometer는 최신 구성 요소(Gnocchi, Aodh, Panko)를 선호하지 않습니다.

CinderNetappEseriesHostType

모든 전자 시리즈 지원이 더 이상 사용되지 않습니다.

ControllerEnableSwiftStorage

대신 **ControllerServices** 매개변수를 조작해야 합니다.

OpenDaylightPort

EndpointMap을 사용하여 OpenDaylight의 기본 포트를 정의합니다.

OpenDaylightConnectionProtocol

이 매개변수의 값은 TLS를 사용하여 Overcloud를 배포하는지 여부에 따라 결정됩니다.

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

```
$ 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 더 이상 사용되지 않는 매개변수가 계속 필요한 경우 기본 **roles_data** 파일에서 사용할 수 있습니다. 그러나 사용자 지정 **roles_data** 파일을 사용하고 있고 오버클라우드에 이러한 더 이상 사용되지 않는 매개변수가 여전히 필요한 경우 **roles_data** 파일을 편집하고 각 역할에 다음을 추가하여 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
  ...
```

5.7. 더 이상 사용되지 않는 CLI 옵션

일부 명령줄 옵션은 오래되었거나 더 이상 사용되지 않으며, 대신 환경 파일의 **parameter_defaults** 섹션에 포함하는 Heat 템플릿 매개 변수를 사용합니다. 다음 표에는 더 이상 사용되지 않는 옵션이 해당하는 Heat 템플릿에 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**

옵션	설명	Heat 템플릿 매개변수
--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**
--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 변경할 필요가 없습니다.	**HypervisorNeutronPhysicalBridge**
--neutron-bridge-mappings	사용할 논리적 브릿지와 물리적 브리지 매핑. 기본적으로 호스트(br-ex)의 외부 브리지를 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 쉼표로 구분된 문자열을 사용합니다.	**NeutronTunnelTypes**
--neutron-tunnel-id-ranges	테넌트 네트워크 할당에 사용할 수 있는 GRE 터널 ID 범위	**NeutronTunnelIdRanges**

옵션	설명	Heat 템플릿 매개변수
--neutron-vni-ranges	테넌트 네트워크 할당에 사용할 수 있는 VXLAN VNI ID 범위	**NeutronVniRanges**
--neutron-network-vlan-ranges	지원할 Neutron ML2 및 Open vSwitch VLAN 매핑 범위. 기본값은 'datacentre' 물리적 네트워크에서 VLAN을 허용하도록 설정되어 있습니다	**NeutronNetworkVLANRanges**
--neutron-mechanism-drivers	neutron 테넌트 네트워크의 메커니즘 드라이버입니다. 기본값은 "openvswitch"입니다. 여러 값을 지정하려면 쉼표로 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 변환하고 환경 파일에 추가하는 것이 좋습니다.

다음은 이러한 새 매개변수 중 일부가 포함된 **deprecated_cli_options.yaml** 이라는 파일의 예입니다.

```
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 3
  CephStorageCount: 3
  ...
```

이 가이드의 다음 예제에는 이러한 새 매개변수를 포함하는 **deprecated_cli_options.yaml** 환경 파일이 포함되어 있습니다.

5.8. 구성 가능 네트워크

이 버전의 Red Hat OpenStack Platform에는 구성 가능한 네트워크에 대한 새로운 기능이 도입되었습니다. 사용자 지정 **roles_data** 파일을 사용하는 경우 파일을 편집하여 각 역할에 구성 가능한 네트워크를 추가합니다. 예를 들어 컨트롤러 노드의 경우 다음을 수행합니다.

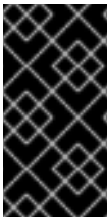
```
- name: Controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
```

구문의 추가 예는 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**, **Storage**, **StorageMgmt**, **Tenant**
데이터베이스	**InternalApi**
Glance	**InternalApi**
Heat	**InternalApi**
Horizon	**InternalApi**
Ironic	필요하지 않음. API에 프로비저닝/컨트롤 플레인 네트워크를 사용합니다.
Keystone	**InternalApi**
로드 밸런서	**External**, **InternalApi**, **Storage**, **StorageMgmt**, **Tenant**
Manila	**InternalApi**

Role	필요한 네트워크
메시지 버스	**InternalApi**
Networker	**InternalApi**, **Tenant**
Neutron API	**InternalApi**
Nova	**InternalApi**
OpenDaylight	**external**,**InternalApi**,**Tenant**
Redis	**InternalApi**
Sahara	**InternalApi**
Swift API	**스토리지**
Swift Storage	**StorageMgmt**
Telemetry	**InternalApi**



중요

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

5.9. CEPH STORAGE 또는 HCI 노드 업그레이드 준비

컨테이너화된 서비스로 업그레이드하여 Ceph Storage 노드를 설치 및 업데이트하는 방법이 변경되었습니다. Ceph Storage 구성은 이제 언더클라우드에 설치하는 **ceph-ansible** 패키지의 플레이북 세트를 사용합니다.

중요

- 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 6.7절. "하이퍼컨버지드 노드 업그레이드" 을 참조하십시오.
- 혼합 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 6.8절. "혼합 하이퍼컨버지드 노드 업그레이드" 을 참조하십시오.

절차

1. director 관리 또는 외부 Ceph Storage 클러스터를 사용하는 경우 **ceph-ansible** 패키지를 설치합니다.
   a. 언더클라우드에서 Ceph Tools 리포지토리를 활성화합니다.

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
```

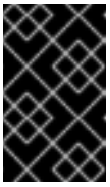
b. **ceph-ansible** 패키지를 언더클라우드에 설치합니다.

```
[stack@director ~]$ sudo yum install -y ceph-ansible
```

2. Ceph별 환경 파일을 확인하고 Ceph별 heat 리소스에서 컨테이너화된 서비스를 사용하는지 확인합니다.

- director에서 관리하는 Ceph Storage 클러스터의 경우 **resource_register** 의 리소스가 **docker/services/ceph-ansible** 의 템플릿을 가리키는지 확인합니다.

```
resource_registry:
  OS::TripleO::Services::CephMgr: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-mgr.yaml
  OS::TripleO::Services::CephMon: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-mon.yaml
  OS::TripleO::Services::CephOSD: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-osd.yaml
  OS::TripleO::Services::CephClient: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-client.yaml
```



중요

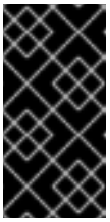
이 구성은 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 환경 파일에 포함되어 있으며, **-e** 를 사용하여 향후 모든 배포 명령에 포함할 수 있습니다.

참고

환경에서 사용하려는 환경 또는 템플릿 파일이 **/usr/share** 디렉터리에 없는 경우 파일의 절대 경로를 포함해야 합니다.

- 외부 Ceph Storage 클러스터의 경우 **resource_register의 리소스가 docker/services/ceph-ansible** 의 템플릿을 가리키는지 확인합니다.

```
resource_registry:
  OS::TripleO::Services::CephExternal: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-external.yaml
```

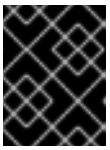


중요

이 구성은 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml** 환경 파일에 포함되어 있으며, **-e** 를 사용하여 향후 모든 배포 명령에 포함할 수 있습니다.

3. director가 관리하는 Ceph Storage 클러스터의 경우 새 **CephAnsibleDisksConfig** 매개변수를 사용하여 디스크가 매핑되는 방법을 정의합니다. 이전 버전의 Red Hat OpenStack Platform에서는 **ceph::profile::params::osds** hieradata를 사용하여 OSD 레이아웃을 정의했습니다. 이

hieradata를 **CephAnsibleDisksConfig** 매개변수 구조로 변환합니다. 다음 예제에서는 공동 배치 및 비할당 **Ceph 저널 디스크의 경우 hieradata를 CephAnsibleDisksConfig** 매개변수 구조로 변환하는 방법을 보여줍니다.



중요

osd_scenario 를 설정해야 합니다. **osd_scenario** 설정을 해제하면 배포에 실패할 수 있습니다.

- Ceph 저널 디스크를 배치하는 시나리오에서는 hieradata에 다음이 포함된 경우 다음을 실행합니다.

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_journal_size: 512
    ceph::profile::params::osds:
      '/dev/sdb': {}
      '/dev/sdc': {}
      '/dev/sdd': {}
```

CephAnsibleDisksConfig 매개변수를 사용하여 hieradata를 변환하고 **ceph::profile::params::osds** 를 **{}** 로 설정합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    journal_size: 512
    osd_scenario: collocated
  ExtraConfig:
      ceph::profile::params::osds: {}
```

- 저널이 더 빠른 전용 장치에 있고 비할당되는 시나리오에서는 hieradata에 다음이 포함된 경우 다음을 수행합니다.

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_journal_size: 512
    ceph::profile::params::osds:
      '/dev/sdb':
        journal: '/dev/sdn'
      '/dev/sdc':
        journal: '/dev/sdn'
      '/dev/sdd':
        journal: '/dev/sdn'
```

CephAnsibleDisksConfig 매개변수를 사용하여 hieradata를 변환하고 **ceph::profile::params::osds** 를 **{}** 로 설정합니다.

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
```

```
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    dedicated_devices:
    - /dev/sdn
    - /dev/sdn
    - /dev/sdn
    journal_size: 512
    osd_scenario: non-collocated
  ExtraConfig:
    ceph::profile::params::osds: {}
```

ceph-ansible 에 사용되는 OSD 디스크 레이아웃 옵션 전체 목록은 **/usr/share/ceph-ansible/group_vars/osds.yml.sample** 의 샘플 파일을 확인합니다.

4. **-e** 옵션을 사용하여 향후 배포 명령을 사용하여 새 Ceph 구성 환경 파일을 포함합니다. 여기에는 다음 파일이 포함됩니다.

   - director에서 관리하는 Ceph Storage:
     - **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml**.
     - Ansible 기반 디스크 매핑이 있는 환경 파일.
     - Ceph Storage 사용자 지정이 포함된 추가 환경 파일입니다.
   - 외부 Ceph Storage:
     - **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml**
     - Ceph Storage 사용자 지정이 포함된 추가 환경 파일입니다.

5.10. 동종 디스크가 아닌 CEPH 또는 HCI 노드의 환경 변수 업데이트

HCI 노드의 경우 계산 서비스 업그레이드 중에 디스크에 이전 구문을 사용하고 스토리지 서비스 업그레이드 중에 디스크의 새 구문을 사용합니다. 그러나 5.9절. “Ceph Storage 또는 HCI 노드 업그레이드 준비” 을 참조하십시오. 그러나 동종이 아닌 디스크의 구문을 업데이트해야 할 수도 있습니다.

업그레이드 중인 노드의 디스크가 동일하지 않으면 동종이 아닙니다. 예를 들어, HCI 노드와 Ceph Storage 노드가 혼합된 디스크는 동종이 아닐 수 있습니다.

OpenStack Platform 12 이상에서는 ceph-ansible 사용을 도입했습니다. 따라서 이기종 디스크로 혼합된 노드를 업데이트하는 방법의 구문이 변경되었습니다. 즉, OpenStack Platform 12에서 시작하여 **RoleExtraConfig** 의 구성 가능 역할 구문을 사용하여 디스크를 표시할 수 없습니다. 다음 예제를 참조하십시오.

다음 예제는 OpenStack Platform 12 이상에서는 작동하지 않습니다.

```
CephStorageExtraConfig:
  ceph::profile::params::osds:
    '/dev/sda'
    '/dev/sdb'
    '/dev/sdc'
    '/dev/sdd'
```

```
ComputeHCIExtraConfig:
  ceph::profile::params::osds:
    '/dev/sda'
    '/dev/sdb'
```

OpenStack Platform 12 이상의 경우 업그레이드하기 전에 템플릿을 업데이트해야 합니다. 동종이 아닌 디스크에 대한 템플릿을 업데이트하는 방법에 대한 자세한 내용은 *Deploy ing an Overcloud with Containerized Red Hat Ceph 가이드의 Ceph Storage Cluster setup 구성을* 참조하십시오.

5.11. 대규모 CEPH 클러스터의 재시작 지연 증가

업그레이드하는 동안 각 Ceph 모니터 및 OSD가 순차적으로 중지됩니다. 중지된 동일한 서비스가 성공적으로 다시 시작될 때까지 마이그레이션이 계속되지 않습니다. Ansible은 15초 동안 대기하고 서비스가 시작될 때까지 5번(재시도) 확인합니다. 서비스를 다시 시작하지 않으면 Operator가 개입할 수 있도록 마이그레이션이 중지됩니다.

Ceph 클러스터의 크기에 따라 재시도 또는 지연 값을 늘려야 할 수 있습니다. 이러한 매개변수 및 기본값의 정확한 이름은 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 클러스터가 30번 검사하고 Ceph OSD가 확인될 때마다 40초 동안 기다린 다음, 각 확인 시 Ceph MON이 있는지 10초 동안 기다린 후 **openstack overcloud deploy** 명령을 사용하여 **-e** 를 사용하여 **yaml** 파일에 다음 매개 변수를 전달합니다.

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

5.12. 스토리지 백엔드 준비

일부 스토리지 백엔드는 구성 후크 사용에서 자체 구성 가능 서비스로 변경되었습니다. 사용자 지정 스토리지 백엔드를 사용하는 경우 환경 디렉터리에서 새 매개 변수 및 리소스가 있는지 연결된 **환경** 파일을 확인합니다. 백엔드의 사용자 지정 환경 파일을 업데이트합니다. 예를 들면 다음과 같습니다.

- **NetApp 블록 스토리지(cinder)** 백엔드의 경우 배포에 새 **environments/cinder-netapp-config.yaml** 을 사용합니다.
- **Dell EMC 블록 스토리지(cinder)** 백엔드의 경우 배포에 새 **environments/cinder-dellsc-config.yaml** 을 사용합니다.
- **Dell EqualLogic Block Storage(cinder)** 백엔드의 경우 배포에 새 **environments/cinder-dellps-config.yaml** 을 사용합니다.

예를 들어 **NetApp Block Storage(cinder)** 백엔드는 이러한 각 버전에 대해 다음 리소스를 사용했습니다.

- OpenStack Platform 10 이하: **OS::TripleO::ControllerExtraConfigPre: ../puppet/extraconfig/pre_deploy/controller/cinder-netapp.yaml**
- OpenStack Platform 11 이상: **OS::TripleO::Services::CinderBackendNetApp: ../puppet/services/cinder-backend-netapp.yaml**

결과적으로 이 백엔드에 새로운 **OS::TripleO::Services::CinderBackendNetApp** 리소스 및 관련 서비스 템플릿을 사용합니다.

5.13. SSL/TLS를 통한 UNDERCLOUD의 공용 API에 대한 액세스 준비

오버클라우드는 업그레이드 중에 언더클라우드의 OpenStack Object Storage(swift) 공용 API에 액세스해야 합니다. 언더클라우드에서 자체 서명된 인증서를 사용하는 경우 각 오버클라우드 노드에 언더클라우드의 인증 기관을 추가해야 합니다.

사전 요구 사항

- 언더클라우드는 공용 API에 SSL/TLS 사용

절차

1. director의 동적 Ansible 스크립트가 OpenStack Platform 12 버전으로 업데이트되었으며, 이 버전에서는 Overcloud 계획에서 **RoleNetHostnameMap** Heat 매개 변수를 사용하여 인벤토리를 정의합니다. 그러나 현재 오버클라우드에서는 **RoleNetHostnameMap** 매개변수가 없는 OpenStack Platform 11 템플릿 버전을 사용합니다. 즉, 다음 명령을 사용하여 생성할 수 있는 임시 정적 인벤토리 파일을 생성해야 합니다.

```
$ openstack server list -c Networks -f value | cut -d"=" -f2 > overcloud_hosts
```

2. 다음이 포함된 Ansible 플레이북(**undercloud-ca.yml**)을 생성합니다.

```
---
- name: Add undercloud CA to overcloud nodes
  hosts: all
  user: heat-admin
  become: true
  vars:
    ca_certificate: /etc/pki/ca-trust/source/anchors/cm-local-ca.pem
  tasks:
    - name: Copy undercloud CA
      copy:
        src: "{{ ca_certificate }}"
        dest: /etc/pki/ca-trust/source/anchors/
    - name: Update trust
      command: "update-ca-trust extract"
    - name: Get the swift endpoint
      shell: |
        sudo hiera swift::keystone::auth::public_url | awk -F/ '{print $3}'
      register: swift_endpoint
      delegate_to: 127.0.0.1
      become: yes
      become_user: stack
    - name: Verify URL
      uri:
        url: https://{{ swift_endpoint.stdout }}/healthcheck
```

```
    return_content: yes
  register: verify
- name: Report output
  debug:
    msg: "{{ ansible_hostname }} can access the undercloud's Public API"
  when: verify.content == "OK"
```

이 플레이북에는 각 노드에서 다음을 수행하는 여러 작업이 포함되어 있습니다.

- Undercloud의 인증 기관 파일을 Overcloud 노드에 복사합니다. 언더클라우드에서 생성한 경우 기본 위치는 **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** 입니다.
- 명령을 실행하여 Overcloud 노드에서 인증 기관 신뢰 데이터베이스를 업데이트합니다.
- Overcloud 노드에서 언더클라우드의 Object Storage Public API를 확인하고 성공했는지 보고합니다.

3. 다음 명령을 사용하여 플레이북을 실행합니다.

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml
```

이는 임시 인벤토리를 사용하여 Ansible에 오버클라우드 노드를 제공합니다.

사용자 지정 인증 기관 파일을 사용하는 경우 **ca_certificate** 변수를 위치로 변경할 수 있습니다. 예를 들면 다음과 같습니다.

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml -e
ca_certificate=/home/stack/ssl/ca.crt.pem
```

4. 결과 Ansible 출력에 노드에 대한 디버그 메시지가 표시되어야 합니다. 예를 들면 다음과 같습니다.

```
ok: [192.168.24.100] => {
    "msg": "overcloud-controller-0 can access the undercloud's Public API"
}
```

관련 정보

- 오버클라우드에서 Ansible 자동화를 실행하는 방법에 대한 자세한 내용은 *Director 설치 및 사용* 가이드 의 "동적 인벤토리 스크립트 실행" 을 참조하십시오.

5.14. 빠른 전달 업그레이드를 위한 등록 구성

fast forward 업그레이드 프로세스는 새 방법을 사용하여 리포지토리를 전환합니다. 즉, 배포 명령에서 이전 **rhel-registration** 환경 파일을 제거해야 합니다. 예를 들면 다음과 같습니다.

- environment-rhel-registration.yaml
- rhel-registration-resource-registry.yaml

fast forward 업그레이드 프로세스는 스크립트를 사용하여 업그레이드 각 단계에서 리포지토리를 변경합니다. 이 스크립트는 **FastForwardCustomRepoScriptContent** 매개변수를 사용하여 **OS::TripleO::Services::TripleoPackages** 구성 가능한 서비스(puppet/**services/tripleo-packages.yaml**)의 일부로 포함되어 있습니다. 스크립트입니다.

-

```
#!/bin/bash
set -e
case $1 in
  ocata)
    subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
    ;;
  pike)
    subscription-manager repos --disable=rhel-7-server-openstack-11-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
    ;;
  queens)
    subscription-manager repos --disable=rhel-7-server-openstack-12-rpms
    subscription-manager release --set=7.9
    subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-osd-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-mon-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-tools-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-13-deployment-tools-rpms
    ;;
  *)
    echo "unknown release $1" >&2
    exit 1
esac
```

director는 각 OpenStack Platform 버전의 업스트림 코드 이름을 스크립트에 전달합니다.

코드명	버전
Ocata	OpenStack Platform 11
Pike	OpenStack Platform 12
마케도스	OpenStack Platform 13

를 변경하면 Ceph Storage 2 리포지토리 **도** 비활성화되고 Ceph Storage 3 MON 및 Tools 리포지토리가 활성화됩니다. 변경하면 Ceph Storage 3 OSD 리포지토리가 컨테이너화되므로 활성화되지 않습니다.

경우에 따라 사용자 지정 스크립트를 사용해야 할 수 있습니다. 예를 들면 다음과 같습니다.

- 사용자 지정 리포지토리 이름으로 Red Hat Satellite 사용.
- 사용자 지정 이름으로 연결 해제된 리포지토리 사용.
- 각 단계에서 실행할 추가 명령입니다.

이러한 경우 **FastForwardCustomRepoScriptContent** 매개변수를 설정하여 사용자 정의 스크립트를 포함합니다.

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    [INSERT UPGRADE SCRIPT HERE]
```

예를 들어 다음 스크립트를 사용하여 일련의 Satellite 6 활성화 키가 있는 리포지토리를 변경합니다.

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    set -e
    URL="satellite.example.com"
    case $1 in
      ocata)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp11 --
org=Default_Organization
        ;;
      pike)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp12 --
org=Default_Organization
        ;;
      queens)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp13 --
org=Default_Organization
        ;;
      *)
        echo "unknown release $1" >&2
        exit 1
    esac
```

이 가이드의 다음 예제에는 사용자 정의 스크립트가 포함된 **custom_repositories_script.yaml** 환경 파일이 포함되어 있습니다.

5.15. 사용자 정의 PUPPET 매개변수 확인

Puppet 매개 변수의 사용자 지정을 위해 **ExtraConfig** 인터페이스를 사용하는 경우 Puppet에서 업그레이드 중에 중복 선언 오류가 보고될 수 있습니다. 이는 puppet 모듈 자체에서 제공하는 인터페이스의 변경 때문입니다.

다음 절차에서는 환경 파일에서 사용자 지정 **ExtraConfig** hieradata 매개변수를 확인하는 방법을 설명합니다.

절차

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

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

2. 선택한 파일에서 모든 역할(예: **ControllerExtraConfig)에 대한 ExtraConfig 매개변수를** 표시하는 경우 해당 파일의 전체 매개변수 구조를 확인합니다.

3. 매개 변수에 **SECTION/parameter** 구문과 **값이** 오는 puppet Hierdata가 포함된 경우 매개 변수로 대체되었을 수 있습니다. 예를 들면 다음과 같습니다.

   ```
   parameter_defaults:
     ExtraConfig:
   ```

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

4. director의 Puppet 모듈을 확인하여 매개변수가 Puppet 클래스 내에 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
$ grep dnsmasq_local_resolv
```

이 경우 새 인터페이스로 변경합니다.

5. 다음은 구문의 변경 사항을 보여주는 예입니다.

- 예 1:

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

변경 사항:

```
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'
```

변경 사항:

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

5.16. 네트워크 인터페이스 템플릿을 새 구조로 변환

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

```
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]
```

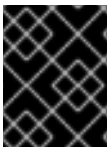
이 구성은 **$network_config** 변수에 저장된 인터페이스 구성을 가져와서 **run-os-net-config.sh** 스크립트의 일부로 삽입합니다.

주의

이 새 구조를 사용하도록 네트워크 인터페이스 템플릿을 업데이트하고 네트워크 인터페이스 템플릿이 여전히 구문을 준수하는지 확인해야 합니다. 이렇게 하지 않으면 fast forward 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. 사용법의 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] ...
```



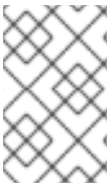
중요

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

자세한 내용은 "네트워크 격리"를 참조하십시오.

5.17. DPDK 및 SR-IOV 구성 확인

이 섹션에서는 DPDK(Data Plane Development Kit) 통합 및 SR-IOV(Single Root Input/Output Virtualization)와 같은 NFV 기술을 사용하는 오버클라우드용입니다. 오버클라우드에서 이러한 기능을 사용하지 않는 경우 이 섹션을 무시하십시오.



참고

Red Hat OpenStack Platform 10에서는 최초 부팅 스크립트 파일을 OpenStack Platform 13용 템플릿인 **host-config-and-reboot.yaml** 로 교체할 필요가 없습니다. 업그레이드 전체에서 최초 부팅 스크립트를 유지 관리하면 중단되고 추가 재부팅이 방지됩니다.

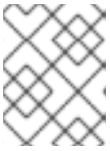
5.17.1. DPDK 환경 업그레이드

DPDK를 사용하는 환경의 경우 특정 서비스 매핑을 확인하여 컨테이너화된 환경으로 성공적으로 전환되었는지 확인합니다.

절차

1. 컨테이너화된 서비스로 전환되어 DPDK 서비스의 빠른 전달 업그레이드가 자동으로 수행됩니다. DPDK의 사용자 지정 환경 파일을 사용하는 경우 컨테이너화된 서비스에 매핑되도록 이러한 환경 파일을 수동으로 조정합니다.

```
OS::TripleO::Services::ComputeNeutronOvsDpdk:
  /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-dpdk-agent.yaml
```



참고

또는 최신 NFV 환경 파일 **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-dpdk.yaml** 을 사용합니다.

2. OpenStack Network(Neutron) 에이전트 서비스를 적절한 컨테이너화된 템플릿에 매핑합니다.

   - DPDK에 기본 **Compute** 역할을 사용하는 경우 heat 템플릿 컬렉션의 **docker/services** 디렉터리에서 **ComputeNeutronOvsAgent** 서비스를 **neutron-ovs-dpdk-agent.yaml** 파일에 매핑합니다.

```
resource_registry:
  OS::TripleO::Services::ComputeNeutronOvsAgent:
    /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-dpdk-
agent.yaml
```

   - DPDK에 사용자 지정 역할을 사용하는 경우 사용자 지정 구성 가능 서비스(예: **ComputeNeutronOvsDpdkAgentCustom**)가 있어야 합니다. docker 디렉터리의 **neutron-ovs-dpdk-agent.yaml** 파일에 이 서비스를 매핑합니다.

3. 다음 서비스와 추가 매개변수를 DPDK 역할 정의에 추가합니다.

```
RoleParametersDefault:
  VhostuserSocketGroup: "hugetlbfs"
  TunedProfileName: "cpu-paritioning"

ServicesDefault:
  - OS::TripleO::Services::ComputeNeutronOvsDPDK
```

4. 다음 서비스를 제거합니다.

```
ServicesDefault:
  - OS::TripleO::Services::NeutronLinuxbridgeAgent
  - OS::TripleO::Services::NeutronVppAgent
  - OS::TripleO::Services::Tuned
```

5.17.2. SR-IOV 환경 업그레이드

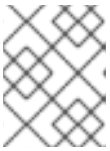
SR-IOV를 사용하는 환경의 경우 다음 서비스 매핑을 확인하여 컨테이너화된 환경으로 성공적으로 전환되었는지 확인합니다.

절차

1. SR-IOV 서비스의 빠른 전달 업그레이드는 컨테이너화된 서비스로 전환되어 자동으로 수행됩니다. SR-IOV에 사용자 지정 환경 파일을 사용하는 경우 이러한 서비스가 컨테이너화된 서비스에 올바르게 매핑되어야 합니다.

```
 OS::TripleO::Services::NeutronSriovAgent:
   /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-sriov-agent.yaml

OS::TripleO::Services::NeutronSriovHostConfig:
   /usr/share/openstack-tripleo-heat-templates/puppet/services/neutron-sriov-host-
config.yaml
```



참고

또는 lastest NFV 환경 파일 **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml** 을 사용합니다.

2. **roles_data.yaml** 파일에 필요한 SR-IOV 서비스가 포함되어 있는지 확인합니다.
SR-IOV에 *기본* **Compute** 역할을 사용하는 경우 OpenStack Platform 13의 이 역할에 적절한 서비스를 포함합니다.

   - **/usr/share/openstack-tripleo-heat-templates** 의 **roles_data.yaml** 파일을 사용자 지정 템플릿 디렉터리(예: **/home/stack/templates**)로 복사합니다.
   - 기본 컴퓨팅 역할에 다음 서비스를 추가합니다.
     - OS::TripleO::Services::NeutronSriovAgent
     - OS::TripleO::Services::NeutronSriovHostConfig
   - 기본 컴퓨팅 역할에서 다음 서비스를 제거합니다.
     - OS::TripleO::Services::NeutronLinuxbridgeAgent
     - OS::TripleO::Services::Tuned
       SR-IOV에 *사용자 지정* **Compute** 역할을 사용하는 경우 **NeutronSriovAgent** 서비스가 있어야 합니다. Red Hat OpenStack Platform 13에서 도입된 **NeutronSriovHostConfig** 서비스를 추가합니다.

참고

ffwd-upgrade 명령을 실행할 때 **roles_data.yaml** 파일을 포함하고 다음 섹션에서 통합해야 합니다.

5.18. 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 노드 **IP** 주소 목록을 **OVERCLOUD_HOSTS** 환경 변수에 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 업그레이드하는 경우 다음을 사용하십시오.

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

b.

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

예를 들면 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 Overcloud 구성" 을 참조하십시오.

5.19. 다음 단계

Overcloud 준비 단계가 완료되었습니다. 이제 *6장. 오버클라우드 업그레이드* 의 단계를 사용하여 오버클라우드를 **10**에서 **13**으로 업그레이드할 수 있습니다.

6장. 오버클라우드 업그레이드

이 섹션에서는 **Overcloud**를 업그레이드합니다. 여기에는 다음 워크플로우가 포함됩니다.

- **fast forward** 업그레이드 준비 명령 실행
- **fast forward upgrade** 명령 실행
- 컨트롤러 노드 업그레이드
- 컴퓨팅 노드 업그레이드
- **Ceph Storage** 노드 업그레이드
- **fast forward** 업그레이드 완료.

이 워크플로를 시작하면 모든 단계를 완료할 때까지 오버클라우드의 **OpenStack** 서비스를 완전히 제어할 수 없습니다. 즉, 모든 노드가 **OpenStack Platform 13**으로 성공적으로 업그레이드될 때까지 워크로드를 관리할 수 없습니다. 워크로드 자체는 영향을 받지 않고 계속 실행됩니다. 오버클라우드 워크로드의 변경 또는 추가 작업은 빠른 전달 업그레이드가 완료될 때까지 기다려야 합니다.

6.1. FAST FORWARD UPGRADE 명령

Fast Forward Upgrade 프로세스에는 특정 프로세스 단계에서 실행되는 다양한 명령이 포함됩니다. 다음 목록에는 각 명령에 대한 몇 가지 기본 정보가 나와 있습니다.

중요

이 목록에는 각 명령에 대한 정보만 포함되어 있습니다. 이러한 명령을 특정 순서로 실행하고 오버클라우드와 관련된 옵션을 제공해야 합니다. 적절한 단계에서 이러한 명령을 실행할 지침이 수신될 때까지 기다립니다.

openstack overcloud ffwd-upgrade 준비

이 명령은 오버클라우드 업그레이드를 위한 초기 준비 단계를 수행합니다. 여기에는 언더클라우드의 현재 오버클라우드 플랜을 새로운 **OpenStack Platform 13** 오버클라우드 계획 및 업데이트된 환경 파일로 교체하는 작업이 포함됩니다. 이 명령은 **openstack overcloud deploy** 명령과 유사하게 작동하며 동일한 여러 옵션을 사용합니다.

openstack overcloud ffwd-upgrade run

이 명령은 **fast forward** 업그레이드 프로세스를 수행합니다. **director**는 새로운 **OpenStack Platform 13** 오버클라우드 계획에 따라 일련의 **Ansible** 플레이북을 생성하고 전체 오버클라우드에서 빠른 전달 작업을 실행합니다. 여기에는 **10**에서 **13**으로 각 **OpenStack Platform** 버전을 통해 업그레이드 프로세스를 실행하는 작업이 포함됩니다.

OpenStack overcloud 업그레이드 실행

이 명령은 역할의 단일 노드 또는 여러 노드에 대해 노드별 업그레이드 구성을 수행합니다. **director**는 오버클라우드 계획에 따라 일련의 **Ansible** 플레이북을 생성하고 선택한 노드에 대해 작업을 실행하여 적절한 **OpenStack Platform 13** 구성으로 노드를 구성합니다. 이 명령은 역할별로 업데이트를 스테이징하는 방법도 제공합니다. 예를 들어 이 명령을 실행하여 먼저 컨트롤러 노드를 업그레이드한 다음 명령을 다시 실행하여 컴퓨팅 노드와 **Ceph Storage** 노드를 업그레이드합니다.

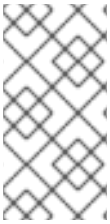
OpenStack overcloud ceph-upgrade run

이 명령은 **Ceph Storage** 버전 업그레이드를 수행합니다. **Ceph Storage** 노드에 대해 **openstack overcloud** 업그레이드를 실행한 후 이 명령을 실행합니다. **director**는 **ceph-ansible** 을 사용하여 **Ceph Storage** 버전 업그레이드를 수행합니다.

OpenStack overcloud ffwd-upgrade converge

이 명령은 오버클라우드 업그레이드의 최종 단계를 수행합니다. 이 마지막 단계에서는 오버클라우드 **Heat** 스택을 **OpenStack Platform 13** 오버클라우드 계획 및 업데이트된 환경 파일과 동기화합니다. 이렇게 하면 생성된 오버클라우드가 새 **OpenStack Platform 13** 오버클라우드의 구성과 일치합니다. 이 명령은 **openstack overcloud deploy** 명령과 유사하게 작동하며 동일한 여러 옵션을 사용합니다.

이러한 명령을 특정 순서로 실행해야 합니다. 이 장의 나머지 섹션에 따라 이러한 명령을 사용하여 빠른 업그레이드를 수행합니다.



참고

오버클라우드에 사용자 지정 이름을 사용하는 경우 각 명령에 대해 **--stack** 옵션으로 사용자 지정 이름을 설정합니다.

6.2. 오버클라우드의 빠른 전달 업그레이드 수행

fast forward 업그레이드를 수행하려면 다음 작업을 수행하는 두 가지 명령을 실행해야 합니다.

- **Overcloud** 계획을 **OpenStack Platform 13**으로 업데이트합니다.
- **fast forward** 업그레이드를 위해 노드를 준비합니다.
- 다음과 같이 **fast forward** 업그레이드 내에서 각 후속 버전의 업그레이드 단계를 통해 실행됩니다.
  - 각 **OpenStack Platform** 서비스에 대한 버전별 작업.
  - **fast forward** 업그레이드 내에서 리포지토리를 순차적으로 각 **OpenStack Platform** 버전으로 변경합니다.
  - 데이터베이스 업그레이드에 필요한 특정 패키지를 업데이트합니다.
  - 각 후속 버전에 대해 데이터베이스 업그레이드 수행.
- **OpenStack Platform 13**으로의 최종 업그레이드를 위해 오버클라우드를 준비합니다.

절차

1. **stackrc** 파일을 소싱합니다.

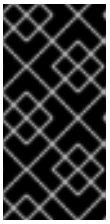
```
$ source ~/stackrc
```

2. 배포에 적합한 모든 관련 옵션 및 환경 파일을 사용하여 **fast forward upgrade prepare** 명령을 실행합니다.

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

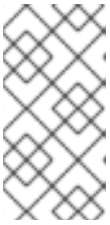
환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일(**-e**). 예를 들면 다음과 같습니다.
  - 컨테이너 이미지 위치가 있는 환경 파일(**overcloud_images.yaml**). 업그레이드 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하기 위해 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
  - 해당하는 경우 **deprecated_cli_options.yaml** 을 사용하여 더 이상 사용되지 않는 **CLI** 옵션을 **Heat** 매개변수에 매핑하는 환경 파일입니다.
  - 해당하는 경우 **custom _repositories_script.yaml**을 사용하여 사용자 지정 리포지토리 스크립트가 있는 환경 파일입니다.
  - **Ceph Storage** 노드를 사용하는 경우 관련 환경 파일입니다.
  - 사용자 환경과 관련된 추가 환경 파일입니다.
- 사용자 지정 스택 이름을 사용하는 경우 **--stack** 옵션으로 이름을 전달합니다.
- 해당하는 경우 **--roles-file** 을 사용하는 사용자 지정역할(**roles_data**) 파일을 사용합니다.



중요

ffwd-upgrade 명령을 수행할지 묻는 프롬프트가 표시됩니다. **yes** 를 입력합니다.



참고

openstack ffwd-upgrade prepare 명령을 여러 번 실행할 수 있습니다. 명령이 실패하면 템플릿의 문제를 수정한 다음 명령을 다시 실행할 수 있습니다.

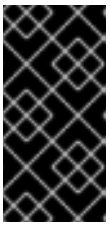
3. 오버클라우드 플랜이 **OpenStack Platform 13** 버전으로 업데이트됩니다. **fast forward** 업그레이드 준비가 완료될 때까지 기다립니다.

4. 업그레이드를 수행하기 전에 **Overcloud**의 스냅샷 또는 백업을 생성합니다.

5. **fast forward upgrade** 명령을 실행합니다.

```
$ openstack overcloud ffwd-upgrade run
```

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



중요

ffwd-upgrade 명령을 수행할지 묻는 프롬프트가 표시됩니다. **yes** 를 입력합니다.

참고

openstack ffwd-upgrade run 명령을 여러 번 실행할 수 있습니다. 명령이 실패하면 템플릿의 문제를 수정한 다음 명령을 다시 실행할 수 있습니다.

6. **fast forward** 업그레이드가 완료될 때까지 기다립니다.

이 단계에서는 다음을 수행합니다.

- 워크로드가 여전히 실행 중입니다
- 오버클라우드 데이터베이스가 **OpenStack Platform 12** 버전으로 업그레이드되었습니다.
- 모든 오버클라우드 서비스가 비활성화됨

- **Ceph Storage** 노드는 여전히 버전 **2**입니다.

즉, 오버클라우드가 **OpenStack Platform 13**에 도달하기 위한 표준 업그레이드 단계를 수행하기 위한 상태입니다.

6.3. 컨트롤러 및 사용자 정의 역할 노드 업그레이드

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

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

[ROLE] 을 역할 이름으로 또는 쉼표로 구분된 역할 목록으로 바꿉니다.

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

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

- **Pacemaker**를 사용하는 모든 역할. 예를 들면 다음과 같습니다. **ControllerOpenStack,**데이터베이스,메시징 및 원격 분석.
- **Networker** 노드
- 기타 사용자 정의 역할

다음 노드를 아직 업그레이드 *하지 마십시오*.

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

- **CephStorage** 노드

이러한 노드는 이후 단계에서 업그레이드합니다.

참고

OpenStack Platform 서비스는 오버클라우드에서 비활성화되어 있으며 검증할 수 없기 때문에 이 절차의 명령은 **--skip-tags** 검증 옵션을 사용합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 모놀리식 컨트롤러 노드를 사용하는 경우 컨트롤러 역할에 대해 **upgrade** 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes Controller --skip-tags validation
```

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

3. 컨트롤러 서비스를 여러 역할로 분할하는 경우 다음을 수행합니다.

a. **Pacemaker** 서비스를 사용하여 역할에 대한 업그레이드 명령을 실행합니다.

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

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

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

```
$ openstack overcloud upgrade run --nodes Networker --skip-tags validation
```

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

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

```
$ openstack overcloud upgrade run --nodes ObjectStorage --skip-tags validation
```

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

이 단계에서는 다음을 수행합니다.

- 워크로드가 여전히 실행 중입니다
- 오버클라우드 데이터베이스가 **OpenStack Platform 13** 버전으로 업그레이드되었습니다.
- 컨트롤러 노드가 **OpenStack Platform 13**으로 업그레이드되었습니다.
- 모든 컨트롤러 서비스가 활성화됨
- 컴퓨팅 노드에는 여전히 업그레이드가 필요합니다.
- **Ceph Storage** 노드는 여전히 버전 **2**이며 업그레이드가 필요합니다

주의

컨트롤러 서비스가 활성화되어 있지만 **Compute** 노드 및 **Ceph Storage** 서비스는 비활성화된 상태에서 워크로드 작업을 수행하지 마십시오. 이로 인해 가상 시스템이 분리될 수 있습니다. 전체 환경을 업그레이드할 때까지 기다립니다.

6.4. 테스트 컴퓨팅 노드 업그레이드

이 프로세스 업그레이드는 테스트용으로 선택되어 **Compute** 노드를 업그레이드합니다. 프로세스에는 **openstack overcloud upgrade run** 명령을 실행하고 테스트 노드로만 작업을 제한하는 **--nodes** 옵션을 포함해야 합니다. 이 절차에서는 명령의 예로 **--nodes compute-0** 을 사용합니다.

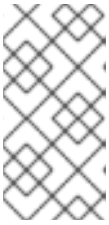
절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes compute-0 --skip-tags validation
```



참고

OpenStack Platform 서비스는 오버클라우드에서 비활성화되어 있고 검증할 수 없기 때문에 **--skip-tags** 검증을 사용합니다.

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

3. 테스트 노드 업그레이드가 완료될 때까지 기다립니다.

6.5. 모든 컴퓨팅 노드 업그레이드

중요

- 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 6.7절. “하이퍼컨버지드 노드 업그레이드” 을 참조하십시오.

- 혼합 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 6.8절. “혼합 하이퍼컨버지드 노드 업그레이드” 을 참조하십시오.

이 프로세스는 나머지 모든 컴퓨팅 노드를 **OpenStack Platform 13**으로 업그레이드합니다. 프로세스

에는 **openstack overcloud upgrade run** 명령을 실행하고 작업을 **Compute** 노드로만 제한하는 **--nodes Compute** 옵션을 포함해야 합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes Compute --skip-tags validation
```

참고

OpenStack Platform 서비스는 오버클라우드에서 비활성화되어 있고 검증할 수 없기 때문에 **--skip-tags** 검증을 사용합니다.

- 사용자 지정 스택 이름을 사용하는 경우 **--stack** 옵션으로 이름을 전달합니다.
- 사용자 지정 **Compute** 역할을 사용하는 경우 **--nodes** 옵션을 사용하여 역할 이름을 포함해야 합니다.

3. 컴퓨팅 노드 업그레이드가 완료될 때까지 기다립니다.

이 단계에서는 다음을 수행합니다.

- 워크로드가 여전히 실행 중입니다
- 컨트롤러 노드 및 컴퓨팅 노드가 **OpenStack Platform 13**으로 업그레이드되었습니다.
- **Ceph Storage** 노드는 여전히 버전 **2**이며 업그레이드가 필요합니다

6.6. 모든 CEPH STORAGE 노드 업그레이드

중요

- 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 6.7절. “하이퍼컨버지드 노드 업그레이드” 을 참조하십시오.
- 혼합 하이퍼컨버지드 배포를 사용하는 경우 업그레이드 방법은 6.8절. “혼합 하이퍼컨버지드 노드 업그레이드” 을 참조하십시오.

이 프로세스는 **Ceph Storage** 노드를 업그레이드합니다. 프로세스에는 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** 클러스터 업그레이드를 수행합니다.

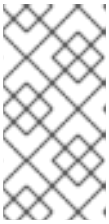
절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업그레이드 명령을 실행합니다.

```
$ openstack overcloud upgrade run --nodes CephStorage --skip-tags validation
```



참고

OpenStack Platform 서비스는 오버클라우드에서 비활성화되어 있고 검증할 수 없기 때문에 **--skip-tags** 검증을 사용합니다.

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

3. 노드 업그레이드가 완료될 때까지 기다립니다.

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

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e <ENVIRONMENT FILE> \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/deprecated_cli_options.yaml \
  -e /home/stack/templates/custom_repositories_script.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  --ceph-ansible-playbook '/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml,/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml'
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일**(-e).** 예를 들면 다음과 같습니다.
  - 컨테이너 이미지 위치가 있는 환경 파일**(overcloud_images.yaml).** 업그레이드 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하기 위해 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
  - 해당하는 경우 **deprecated_cli_options.yaml** 을 사용하여 더 이상 사용되지 않는 **CLI** 옵션을 **Heat** 매개변수에 매핑하는 환경 파일입니다.
  - 해당하는 경우 **custom _repositories_script.yaml**을 사용하여 사용자 지정 리포지토리 스크립트가 있는 환경 파일입니다.
  - **Ceph Storage** 노드의 관련 환경 파일입니다.
  - 사용자 환경과 관련된 추가 환경 파일입니다.
- 사용자 지정 스택 이름을 사용하는 경우 **--stack** 옵션으로 이름을 전달합니다.

- 해당하는 경우 **--roles-file** 을 사용하는 사용자 지정역할(**roles_data)** 파일을 사용합니다.
- 다음 **ansible** 플레이북:
- **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**
- **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

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

6.7. 하이퍼컨버지드 노드 업그레이드

ComputeHCI 역할의 하이퍼컨버지드 노드만 사용하며 전용 계산 노드 또는 전용 **Ceph** 노드를 사용하지 않는 경우 노드를 업그레이드하려면 다음 절차를 완료합니다.

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 **upgrade** 명령에 전달합니다.

3. **Ceph Storage** 업그레이드 명령을 실행합니다. 예를 들면 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 들면 다음과 같습니다.
  - 컨테이너 이미지 위치가 있는 환경 파일(**overcloud_images.yaml**). 업그레이드 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하기 위해 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
  - 해당하는 경우 **deprecated_cli_options.yaml** 을 사용하여 더 이상 사용되지 않는 **CLI** 옵션을 **Heat** 매개변수에 매핑하는 환경 파일입니다.
  - 해당하는 경우 **custom _repositories_script.yaml**을 사용하여 사용자 지정 리포지토리 스크립트가 있는 환경 파일입니다.
  - **Ceph Storage** 노드의 관련 환경 파일입니다.
- 사용자 지정 스택 이름을 사용하는 경우 **--stack** 옵션으로 이름을 전달합니다.
- 해당하는 경우 **--roles-file** 을 사용하는 사용자 지정역할(**roles_data**) 파일을 사용합니다.
- 다음 **ansible** 플레이북:
- **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**
- **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

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

6.8. 혼합 하이퍼컨버지드 노드 업그레이드

ComputeHCI 역할과 같은 하이퍼컨버지드 노드 외에 전용 계산 노드 또는 전용 **ceph** 노드를 사용하는 경우 다음 절차를 완료하여 노드를 업그레이드하십시오.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 컴퓨팅 노드에 대한 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 대한 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 대한 업그레이드 명령을 실행합니다.

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

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

8. **Ceph Storage** 업그레이드 명령을 실행합니다. 예를 들면 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 들면 다음과 같습니다.
  - 컨테이너 이미지 위치가 있는 환경 파일(**overcloud_images.yaml**). 업그레이드 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하기 위해 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
  - 해당하는 경우 **deprecated_cli_options.yaml** 을 사용하여 더 이상 사용되지 않는 **CLI** 옵션을 **Heat** 매개변수에 매핑하는 환경 파일입니다.
  - 해당하는 경우 **custom _repositories_script.yaml**을 사용하여 사용자 지정 리포지토리 스크립트가 있는 환경 파일입니다.
  - **Ceph Storage** 노드의 관련 환경 파일입니다.
  - 사용자 환경과 관련된 추가 환경 파일입니다.
- 사용자 지정 스택 이름을 사용하는 경우 **--stack** 옵션으로 이름을 전달합니다.
- 해당하는 경우 **--roles-file** 을 사용하는 사용자 지정역할(**roles_data**) 파일을 사용합니다.
- 다음 **ansible** 플레이북:
- **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**

- **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

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

이 단계에서는 다음을 수행합니다.

- 모든 노드가 **OpenStack Platform 13**으로 업그레이드되고 워크로드가 계속 실행 중입니다.

이제 환경이 업그레이드되지만 마지막 단계를 수행하여 업그레이드를 완료해야 합니다.

6.9. FAST FORWARD 업그레이드 종료

빠른 업그레이드를 수행하려면 오버클라우드 스택을 업데이트하기 위한 최종 단계가 필요합니다. 이렇게 하면 스택의 리소스 구조가 **OpenStack Platform 13**의 일반 배포에 맞춰 진행되며 나중에 표준 **openstack overcloud deploy** 기능을 수행할 수 있습니다.

절차

1. **stackrc** 파일을 소싱합니다.

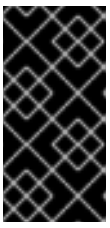
```
$ source ~/stackrc
```

2. **fast forward upgrade finalization** 명령을 실행합니다.

```
$ openstack overcloud ffwd-upgrade converge \
   --templates \
   -e /home/stack/templates/overcloud_images.yaml \
   -e /home/stack/templates/deprecated_cli_options.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
   -e /home/stack/templates/ceph-customization.yaml \
   -e <OTHER ENVIRONMENT FILES>
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일(**-e**). 예를 들면 다음과 같습니다.
  - 컨테이너 이미지 위치가 있는 환경 파일(**overcloud_images.yaml**). 업그레이드 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하기 위해 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
  - 해당하는 경우 **deprecated_cli_options.yaml** 을 사용하여 더 이상 사용되지 않는 **CLI** 옵션을 **Heat** 매개변수에 매핑하는 환경 파일입니다.
  - **Ceph Storage** 노드를 사용하는 경우 관련 환경 파일입니다.
  - 사용자 환경과 관련된 추가 환경 파일입니다.
- 사용자 지정 스택 이름을 사용하는 경우 **--stack** 옵션으로 이름을 전달합니다.
- 해당하는 경우 **--roles-file** 을 사용하는 사용자 지정역할(**roles_data**) 파일을 사용합니다.



중요

ffwd-upgrade 명령을 수행할지 묻는 프롬프트가 표시됩니다. **yes** 를 입력합니다.

3. **fast forward** 업그레이드 완료가 완료될 때까지 기다립니다.

6.10. 다음 단계

오버클라우드 업그레이드가 완료되었습니다. 이제 *8장. 업그레이드 후 단계 실행* 의 단계를 사용하여 적절한 업그레이드 후 오버클라우드 설정을 수행할 수 있습니다. 향후 배포 작업을 위해 업그레이드 중에 생성되거나 변환된 새 환경 파일을 포함하여 **OpenStack Platform 13** 환경과 관련된 모든 환경 파일을 포함해야 합니다.

7장. 업그레이드 후 오버클라우드 재부팅

Red Hat OpenStack 환경을 업그레이드한 후 오버클라우드를 재부팅합니다. 재부팅은 연결된 커널, 시스템 수준 및 컨테이너 구성 요소 업데이트로 노드를 업데이트합니다. 이러한 업데이트는 성능 및 보안상의 이점을 제공할 수 있습니다.

다음 재부팅 절차를 수행하기 위해 다운타임을 계획합니다.

7.1. 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.

7.2. 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 **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** 노드를 재부팅할 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
```

7.3. 컴퓨팅 노드 재부팅

컴퓨팅 노드를 재부팅하려면 다음 워크플로가 포함됩니다.

- 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. 재부팅하려는 컴퓨팅 노드를 확인하려면 모든 컴퓨팅 노드를 나열합니다.

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

3. 오버클라우드에서 컴퓨팅 노드를 선택하여 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 컴퓨팅 노드 간 가상 머신 마이그레이션을 참조하십시오.

6. **Compute** 노드에 로그인하여 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 활성화되어 있는지 확인합니다.

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

7.4. 컴퓨팅 HCI 노드 재부팅

다음 절차에서는 **Compute HCI(**하이퍼컨버지드 인프라**)** 노드를 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. **stack** 사용자로 언더클라우드에 로그인합니다.

3. 모든 컴퓨팅 노드 및 해당 **UUID**를 나열합니다.

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

재부팅하려는 컴퓨팅 노드의 **UUID**를 확인합니다.

4. 언더클라우드에서 컴퓨팅 노드를 선택하여 비활성화합니다.

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

5. 컴퓨팅 노드에 모든 인스턴스를 나열합니다.

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

6. 다음 명령 중 하나를 사용하여 인스턴스를 마이그레이션합니다.

a. 인스턴스를 선택한 특정 호스트로 마이그레이션합니다.

```
(overcloud) $ openstack server migrate [instance-id] --live [target-host]--wait
```

b. **nova-scheduler**에서 대상 호스트를 자동으로 선택하도록 합니다.

```
(overcloud) $ nova live-migration [instance-id]
```

c. 한 번에 모든 인스턴스를 실시간 마이그레이션합니다.

```
$ nova host-evacuate-live [hostname]
```

참고

nova 명령으로 인해 몇 가지 사용 중단 경고가 표시될 수 있으며, 이러한 경고는 무시해도 됩니다.

7. 마이그레이션이 완료될 때까지 기다립니다.

8. 마이그레이션을 성공적으로 완료했음을 확인합니다.

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

9. 선택한 컴퓨팅 노드에 남은 항목이 없을 때까지 인스턴스를 계속 마이그레이션합니다.

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

```
$ sudo ceph -s
```

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

11. 컴퓨팅 **HCI** 노드를 재부팅합니다.

```
$ sudo reboot
```

12. 노드가 부팅될 때까지 기다립니다.

13. 컴퓨팅 노드를 다시 활성화합니다.

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

14. 컴퓨팅 노드가 활성화되어 있는지 확인합니다.

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

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

16. 완료되면 **Ceph MON** 또는 컨트롤러 노드에 로그인하고 클러스터 재조정을 다시 활성화합니

다.

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

17. 최종 상태 검사를 수행하여 클러스터가 **HEALTH_OK**를 보고하는지 확인합니다.

```
$ sudo ceph status
```

8장. 업그레이드 후 단계 실행

이 프로세스는 기본 업그레이드 프로세스를 완료한 후 최종 단계를 구현합니다. 여기에는 이미지 변경 및 추가 구성 단계 또는 빠른 전달 업그레이드 프로세스가 완료된 후 고려 사항이 포함됩니다.

8.1. 언더클라우드 검증

다음은 언더클라우드의 기능을 확인하는 일련의 단계입니다.

절차

1. 언더클라우드 액세스 세부 정보를 가져옵니다.

```
$ source ~/stackrc
```

2. 실패한 **Systemd** 서비스를 확인합니다.

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

3. 언더클라우드의 사용 가능한 공간을 확인합니다.

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

"Undercloud Requirements(Undercloud 요구 사항)" 를 기준으로 사용하여 사용 가능한 공간이 적절한지 확인합니다.

4. 언더클라우드에 **NTP**가 설치되어 있는 경우 시계가 동기화되었는지 확인합니다.

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

5. 언더클라우드 네트워크 서비스를 확인합니다.

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

모든 에이전트는 **Alive** 이고 해당 상태는 **UP** 이어야 합니다.

6. 언더클라우드 계산 서비스를 확인합니다.

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

모든 에이전트의 상태가 활성화되고 상태가 **up**이어야 합니다

관련 정보

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

8.2. 컨테이너화된 오버클라우드 검증

다음은 컨테이너화된 오버클라우드의 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)**가있어야 하며 유지 관리 모드는 **false** 여야 합니다.

3. 실패한 **Systemd** 서비스를 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. 실패한 컨테이너화된 서비스를 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 이러한 세부 정보를 사용합니다.

```
(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
```

<PASSWORD> 및 **<IP ADDRESS>** 세부 정보를 **haproxy.stats** 서비스의 실제 세부 정보로 바꿉니다. 결과 목록은 각 노드의 **OpenStack Platform** 서비스와 해당 연결 상태를 표시합니다.

참고

노드가 **Redis** 서비스를 실행하는 경우 하나의 노드만 해당 서비스의 **ON** 상태를 표시합니다. 이는 **Redis**가 한 번에 하나의 노드에서만 실행되는 액티브-패시브 서비스이기 때문입니다.

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"
```

다음 항목 검색:

- 온라인 의 모든 클러스터 노드.
- 모든 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. **Overcloud 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 OSD**에서 사용 가능한 공간이 있는지 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 클록이 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. 오버클라우드 액세스 세부 정보를 소싱합니다.

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

14. 오버클라우드 네트워크 서비스를 확인합니다.

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

모든 에이전트는 **Alive** 이고 해당 상태는 **UP** 이어야 합니다.

15. 오버클라우드 계산 서비스를 확인합니다.

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

모든 에이전트의 상태가 활성화되고 상태가 **up**이어야 합니다

16. 오버클라우드 볼륨 서비스를 확인합니다.

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

모든 에이전트의 상태가 활성화되고 상태가 **up** 이어야 합니다.

관련 정보

- "Red Hat 권장 구성으로 OpenStack 환경이 배포되었는지 어떻게 확인할 수 있습니까?"라는 문서를 검토하십시오. 이 문서에서는 **Red Hat OpenStack Platform** 환경을 확인하고 **Red Hat**의 권장 사항에 맞게 구성을 조정하는 방법에 대해 설명합니다.

8.3. 오버클라우드 이미지 업그레이드

현재 오버클라우드 이미지를 새 버전으로 교체해야 합니다. 새 이미지를 사용하면 최신 버전의 **OpenStack Platform** 소프트웨어를 사용하여 **director**가 노드를 세부 검사하고 프로비저닝할 수 있습니다.

사전 요구 사항

- 언더클라우드를 최신 버전으로 업그레이드했습니다.

절차

1. 언더클라우드 액세스 세부 정보를 가져옵니다.

```
$ source ~/stackrc
```

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

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

3. 아카이브를 추출합니다.

```
$ 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 **director**로 가져옵니다.

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

5. 새 이미지를 사용하도록 노드를 구성합니다.

```
$ 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** 템플릿과 함께 **OpenStack Platform 13** 이미지만 사용합니다.

중요

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

8.4. 배포 테스트

Overcloud가 업그레이드되었지만 테스트 배포를 실행하여 향후 배포 작업이 성공적으로 수행되도록 하는 것이 좋습니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 배포 명령을 실행하고 오버클라우드와 관련된 모든 환경 파일을 포함합니다.

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

환경과 관련된 다음 옵션을 포함합니다.

- **-e** 를 사용하여 사용자 지정 구성 환경 파일.
- 해당하는 경우 **--roles-file** 을 사용하는 사용자 지정역할(**roles_data)** 파일을 사용합니다.

3. 배포가 완료될 때까지 기다립니다.

8.5. CONCLUSION

이렇게 하면 **fast forward** 업그레이드 프로세스가 완료됩니다.

부록 A. 언더클라우드 복원

다음 복원 절차에서는 언더클라우드 노드가 실패했으며 복구할 수 없는 상태라고 가정합니다. 다음 절차에서는 새로운 설치에서 데이터베이스와 중요한 파일 시스템을 복원하는 작업이 포함됩니다. 다음과 같이 가정합니다.

- 최신 버전의 **Red Hat Enterprise Linux 7**을 다시 설치했습니다.
- 하드웨어 레이아웃도 마찬가지입니다.
- 시스템의 호스트 이름과 **Undercloud** 설정은 동일합니다.
- 백업 아카이브가 루트 디렉터리에 복사되었습니다.

절차

1. **root** 사용자로 언더클라우드에 로그인합니다.
2. 시스템을 **Content Delivery Network**에 등록하고 메시지가 표시되면 고객 포털 사용자 이름과 암호를 입력합니다.

```
[root@director ~]# subscription-manager register
```

3. **Red Hat OpenStack Platform** 인타이틀먼트를 연결합니다.

```
[root@director ~]# subscription-manager attach --pool=Valid-Pool-Number-123456
```

4. 기본 리포지토리를 모두 비활성화하고 필수 **Red Hat Enterprise Linux** 리포지토리를 활성화합니다.

```
[root@director ~]# subscription-manager repos --disable=*
[root@director ~]# subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-extras-rpms --enable=rhel-7-server-rh-common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-openstack-10-rpms
```

5. 시스템에서 업데이트를 실행하여 기본 시스템 패키지가 최신 상태인지 확인합니다.

```
[root@director ~]# yum update -y
[root@director ~]# reboot
```

6. **Undercloud**의 시간이 동기화되었는지 확인합니다. 예를 들면 다음과 같습니다.

```
[root@director ~]# yum install -y ntp
[root@director ~]# systemctl start ntpd
[root@director ~]# systemctl enable ntpd
[root@director ~]# ntpdate pool.ntp.org
[root@director ~]# systemctl restart ntpd
```

7. **Undercloud** 백업 아카이브를 **Undercloud**의 루트 디렉터리에 복사합니다. 다음 단계에서는 **undercloud-backup-$TIMESTAMP.tar** 을 파일 이름으로 사용합니다. 여기서 **$TIMESTAMP**는 아카이브의 타임스탬프에 대한 **Bash** 변수입니다.

8. 데이터베이스 서버 및 클라이언트 툴을 설치합니다.

```
[root@director ~]# yum install -y mariadb mariadb-server
```

9. 데이터베이스를 시작합니다.

```
[root@director ~]# systemctl start mariadb
[root@director ~]# systemctl enable mariadb
```

10. 데이터베이스 백업 크기를 수용하도록 허용된 패킷을 늘립니다.

```
[root@director ~]# mysql -uroot -e"set global max_allowed_packet = 1073741824;"
```

11. 아카이브에서 데이터베이스 및 데이터베이스 구성을 추출합니다.

```
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar
etc/my.cnf.d/*server*.cnf
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar root/undercloud-all-
databases.sql
```

12. 데이터베이스 백업을 복원합니다.

```
[root@director ~]# mysql -u root < /root/undercloud-all-databases.sql
```

13. 루트 구성 파일의 임시 버전을 추출합니다.

```
[root@director ~]# tar -xvf undercloud-backup-$TIMESTAMP.tar root/.my.cnf
```

14. 이전 **root** 데이터베이스 암호를 가져옵니다.

```
[root@director ~]# OLDPASSWORD=$(sudo cat root/.my.cnf | grep -m1 password | cut -d'='
-f2 | tr -d """)
```

15. **root** 데이터베이스 암호를 재설정합니다.

```
[root@director ~]# mysqladmin -u root password "$OLDPASSWORD"
```

16. 임시 디렉터리에서 루트 디렉토리로 루트 구성 파일을 이동합니다.

```
[root@director ~]# mv ~/root/.my.cnf ~/.
[root@director ~]# rmdir ~/root
```

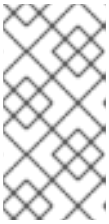
17. 이전 사용자 권한 목록을 가져옵니다.

```
[root@director ~]# mysql -e 'select host, user, password from mysql.user;'
```

18. 나열된 각 호스트에 대한 이전 사용자 권한을 제거합니다. 예를 들면 다음과 같습니다.

```
[root@director ~]# HOST="192.0.2.1"
[root@director ~]# USERS=$(mysql -Nse "select user from mysql.user WHERE user !=
\"root\" and host = \"$HOST\";" | uniq | xargs)
[root@director ~]# for USER in $USERS ; do mysql -e "drop user \"$USER\"@\"$HOST\"" ||
true ;done
[root@director ~]# for USER in $USERS ; do mysql -e "drop user $USER" || true ;done
[root@director ~]# mysql -e 'flush privileges'
```

호스트 **IP** 및 모든 호스트(**"%"**)를 통해 액세스하는 모든 사용자에게 이 작업을 수행합니다.



참고

HOST 매개변수의 IP 주소는 컨트롤 플레인에 있는 언더클라우드의 IP 주소입니다.

19. **데이터베이스를 다시 시작하십시오.**

```
[root@director ~]# systemctl restart mariadb
```

20. **stack 사용자를 생성합니다.**

```
[root@director ~]# useradd stack
```

21. **사용자 암호를 설정합니다.**

```
[root@director ~]# passwd stack
```

22. **sudo 사용 시 암호를 요구하지 않도록 비활성화합니다.**

```
[root@director ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/stack
[root@director ~]# chmod 0440 /etc/sudoers.d/stack
```

23. **stack 사용자 홈 디렉터리를 복원합니다.**

```
# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar home/stack
```

24. **policycoreutils-python 패키지를 설치합니다.**

```
[root@director ~]# yum -y install policycoreutils-python
```

25. **openstack-glance 패키지를 설치하고 데이터 및 파일 권한을 복원합니다.**

```
[root@director ~]# yum install -y openstack-glance
[root@director ~]# tar --xattrs --xattrs-include='*.*' -xvC / -f undercloud-backup-
$TIMESTAMP.tar var/lib/glance/images
[root@director ~]# chown -R glance: /var/lib/glance/images
[root@director ~]# restorecon -R /var/lib/glance/images
```

26. **openstack-swift 패키지를 설치하고 데이터 및 파일 권한을 복원합니다.**

```
[root@director ~]# yum install -y openstack-swift
[root@director ~]# tar --xattrs --xattrs-include='*.*' -xvC / -f undercloud-backup-$TIMESTAMP.tar srv/node
[root@director ~]# chown -R swift: /srv/node
[root@director ~]# restorecon -R /srv/node
```

27. **openstack-keystone 패키지를 설치하고 구성 데이터를 복원합니다.**

```
[root@director ~]# yum -y install openstack-keystone
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/keystone
[root@director ~]# restorecon -R /etc/keystone
```

28. **openstack-heat 및 복원 구성을 설치합니다.**

```
[root@director ~]# yum install -y openstack-heat*
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/heat
[root@director ~]# restorecon -R /etc/heat
```

29. **puppet을 설치하고 설정 데이터를 복원합니다.**

```
[root@director ~]# yum install -y puppet hiera
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/puppet/hieradata/
```

30. **언더클라우드에서 SSL을 사용하는 경우 CA 인증서를 새로 고칩니다. 언더클라우드 구성에 따라 사용자 제공 인증서 또는 자동 생성된 인증서의 단계를 사용합니다.**

- **언더클라우드가 사용자 제공 인증서로 구성된 경우 다음 단계를 완료합니다.**

  a. **인증서를 추출합니다.**

  ```
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/instack-certs/undercloud.pem
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/ca-trust/source/anchors/*
  ```

  b. **SELinux 컨텍스트를 복원하고 파일 시스템 레이블을 관리합니다.**

-

```
[root@director ~]# restorecon -R /etc/pki
[root@director ~]# semanage fcontext -a -t etc_t "/etc/pki/instack-certs(/.*)?"
[root@director ~]# restorecon -R /etc/pki/instack-certs
```

c. **인증서를 업데이트합니다.**

```
[root@director ~]# update-ca-trust extract
```

- **certmonger 를 사용하여 언더클라우드의 인증서를 자동으로 생성하는 경우 다음 단계를 완료합니다.**

a. **인증서, CA 인증서 및 certmonger 파일을 추출합니다.**

```
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar
var/lib/certmonger/*
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/tls/*
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/ca-
trust/source/anchors/*
```

b. **SELinux 컨텍스트를 복원하십시오.**

```
[root@director ~]# restorecon -R /etc/pki
[root@director ~]# restorecon -R /var/lib/certmonger
```

c. **/var/lib/certmonger/lock 파일을 제거합니다.**

```
[root@director ~]# rm -f /var/lib/certmonger/lock
```

31. **stack 사용자로 전환합니다.**

```
[root@director ~]# su - stack
[stack@director ~]$
```

32. **python-tripleoclient 패키지를 설치합니다.**

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

33. **Undercloud** 설치 명령을 실행합니다. **stack** 사용자의 홈 디렉터리에서 실행해야 합니다.

```
[stack@director ~]$ openstack undercloud install
```

설치가 완료되면 언더클라우드에서 오버클라우드에 대한 연결을 자동으로 복원합니다. 노드는 보류 중인 작업에 대해 계속 **OpenStack Orchestration(heat)**을 폴링합니다.

부록 B. 오버클라우드 복원

B.1. 오버클라우드 컨트롤 플레인 서비스 복원

다음 절차에서는 오버클라우드 데이터베이스 및 구성의 백업을 복원합니다. 이 경우 세 개의 컨트롤러 노드 모두에서 특정 작업을 동시에 수행할 수 있도록 3개의 터미널 창을 여는 것이 좋습니다. 또한 고가용성 작업을 수행하기 위해 컨트롤러 노드를 선택하는 것이 좋습니다. 이 절차에서는 이 컨트롤러 노드를 *부트스트랩 컨트롤러 노드*로 참조합니다.

중요

이 절차에서는 컨트롤 플레인 서비스만 복원합니다. Ceph Storage 노드의 복원 계산 노드 워크로드 또는 데이터는 포함되지 않습니다.

참고

Red Hat은 OVS(Open vSwitch) 및 기본 OVN(Open Virtual Network)과 같은 기본 SDN을 사용하여 Red Hat OpenStack Platform 백업을 지원합니다. 타사 SDN에 대한 자세한 내용은 타사 SDN 문서를 참조하십시오.

절차

1. Pacemaker를 중지하고 컨테이너화된 모든 서비스를 제거합니다.

   a. 부트스트랩 컨트롤러 노드에 로그인하고 pacemaker 클러스터를 중지합니다.

   ```
   # sudo pcs cluster stop --all
   ```

   b. 클러스터가 완전히 종료될 때까지 기다립니다.

   ```
   # sudo pcs status
   ```

   c. 모든 컨트롤러 노드에서 OpenStack 서비스의 모든 컨테이너를 제거합니다.

   ```
   # docker stop $(docker ps -a -q)
   # docker rm $(docker ps -a -q)
   ```

2.

실패한 주요 버전 업그레이드에서 복원하는 경우 모든 노드에서 발생한 **yum** 트랜잭션을 취소해야 할 수 있습니다. 이 작업은 각 노드에서 다음 작업을 수행해야 합니다.

a. 이전 버전의 리포지토리를 활성화합니다. 예를 들면 다음과 같습니다.

```
# sudo subscription-manager repos --enable=rhel-7-server-openstack-10-rpms
# sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
# sudo subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
```

b. 다음 **Ceph** 리포지토리를 활성화합니다.

```
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-2-tools-rpms
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-2-mon-rpms
```

c. **yum** 기록을 확인합니다.

```
# sudo yum history list all
```

업그레이드 프로세스 중에 발생한 트랜잭션을 식별합니다. 이러한 작업은 대부분 컨트롤러 노드 중 하나에서 수행됩니다(업그레이드 중에 부트스트랩 노드로 선택한 컨트롤러 노드). 특정 트랜잭션을 확인해야 하는 경우 **history info** 하위 명령을 사용하여 확인합니다.

```
# sudo yum history info 25
```

참고

yum history를 모두 나열 하여 각 트랜잭션에서 실행된 명령을 표시하려면 **yum.conf** 파일에 **history_list_view=commands** 를 설정합니다.

d. 업그레이드 이후 발생한 모든 **yum** 트랜잭션을 되돌립니다. 예를 들면 다음과 같습니다.

```
# sudo yum history undo 25
# sudo yum history undo 24
# sudo yum history undo 23
...
```

마지막 트랜잭션에서 시작하여 내림차순으로 계속합니다. 롤백 옵션을 사용하여 한 번 실행에서 여러 트랜잭션을 되돌릴 수도 있습니다. 예를 들어 다음 명령은 마지막 트랜잭션에

서 **23**으로 트랜잭션을 롤백합니다.

```
# sudo yum history rollback 23
```

중요

각 트랜잭션의 역추적을 확인할 수 있도록 롤백 대신 각 트랜잭션에 대해 실행 취소 를 사용하는 것이 좋습니다.

e. 관련 **yum** 트랜잭션이 되돌리면 모든 노드에서 원래 **OpenStack Platform** 리포지토리만 활성화합니다. 예를 들면 다음과 같습니다.

```
# sudo subscription-manager repos --disable=rhel-7-server-openstack-*-rpms
# sudo subscription-manager repos --enable=rhel-7-server-openstack-10-rpms
```

f. 다음 **Ceph** 리포지토리를 비활성화합니다.

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

3. 데이터베이스를 복원하십시오.

a. 데이터베이스 백업을 부트스트랩 컨트롤러 노드에 복사합니다.

b. 모든 컨트롤러 노드에서 데이터베이스 포트에 대한 외부 연결을 중지합니다.

```
# MYSQLIP=$(hiera -c /etc/puppet/hiera.yaml mysql_bind_host)
# sudo /sbin/iptables -I INPUT -d $MYSQLIP -p tcp --dport 3306 -j DROP
```

이렇게 하면 모든 데이터베이스 트래픽을 노드에 격리합니다.

c. 데이터베이스 복제를 일시적으로 비활성화합니다. 모든 컨트롤러 노드에서 **/etc/my.cnf.d/galera.cnf** 파일을 편집합니다.

```
# vi /etc/my.cnf.d/galera.cnf
```

다음과 같이 변경합니다.

- **wsrep_cluster_address** 매개 변수를 주석 처리합니다.
- **wsrep_provider** 를 **none**으로 설정합니다

d. **/etc/my.cnf.d/galera.cnf** 파일을 저장합니다.

e. 모든 컨트롤러 노드에서 **MariaDB** 데이터베이스가 비활성화되어 있는지 확인합니다. **OpenStack Platform 13**으로 업그레이드하는 동안 **MariaDB** 서비스는 이전에 비활성화한 컨테이너화된 서비스로 이동합니다. 서비스가 호스트에서 프로세스로 실행되지 않는지 확인합니다.

```
# mysqladmin -u root shutdown
```

참고

HAProxy에서 데이터베이스가 비활성화되었다는 경고가 표시될 수 있습니다.

f. 기존 **MariaDB** 데이터 디렉터리를 이동하고 모든 컨트롤러 노드에서 새 데이터 디렉토리를 준비합니다.

```
# mv /var/lib/mysql/ /var/lib/mysql.old
# mkdir /var/lib/mysql
# chown mysql:mysql /var/lib/mysql
# chmod 0755 /var/lib/mysql
# mysql_install_db --datadir=/var/lib/mysql --user=mysql
# chown -R mysql:mysql /var/lib/mysql/
# restorecon -R /var/lib/mysql
```

g. 모든 컨트롤러 노드에서 데이터베이스를 수동으로 시작합니다.

```
# mysqld_safe --skip-grant-tables --skip-networking --wsrep-on=OFF &
```

h. 모든 컨트롤러 노드에서 이전 암호를 **Reset the database password**를 가져옵니다.

-

```
# OLDPASSWORD=$(sudo cat .my.cnf | grep -m1 password | cut -d'=' -f2 | tr -d '"')
# mysql -uroot -e"use mysql;update user set
password=PASSWORD($OLDPASSWORD)"
```

i. 모든 컨트롤러 노드에서 데이터베이스를 중지합니다.

```
# /usr/bin/mysqladmin -u root shutdown
```

j. **--skip-grant-tables** 옵션 없이 부트스트랩 컨트롤러 노드에서 데이터베이스를 수동으로 시작합니다.

```
# mysqld_safe --skip-networking --wsrep-on=OFF &
```

k. 부트스트랩 컨트롤러 노드에서 **OpenStack** 데이터베이스를 복원합니다. 이 작업은 나중에 다른 컨트롤러 노드에 복제됩니다.

```
# mysql -u root < openstack_database.sql
```

l. 부트스트랩 컨트롤러 노드에서 사용자 및 권한을 복원합니다.

```
# mysql -u root < grants.sql
```

m. 다음 명령을 사용하여 부트스트랩 컨트롤러 노드를 종료합니다.

```
# mysqladmin shutdown
```

n. 데이터베이스 복제를 활성화합니다. 모든 컨트롤러 노드에서 **/etc/my.cnf.d/galera.cnf** 파일을 편집합니다.

```
# vi /etc/my.cnf.d/galera.cnf
```

다음과 같이 변경합니다.

- **wsrep_cluster_address** 매개 변수의 주석을 제거합니다.

- **wsrep_provider** 를 **/usr/lib64/galera/libgalera_smm.so**로 설정합니다.

o. **/etc/my.cnf.d/galera.cnf** 파일을 저장합니다.

p. 부트스트랩 노드에서 데이터베이스를 실행합니다.

```
# /usr/bin/mysqld_safe --pid-file=/var/run/mysql/mysqld.pid --
socket=/var/lib/mysql/mysql.sock --datadir=/var/lib/mysql --log-
error=/var/log/mysql_cluster.log --user=mysql --open-files-limit=16384 --wsrep-cluster-
address=gcomm:// &
```

ws rep-cluster-address 옵션에 노드 부족으로 인해 **Galera**가 새 클러스터를 생성하고 부트스트랩 노드를 마스터 노드로 설정합니다.

q. 노드의 상태를 확인합니다.

```
# clustercheck
```

이 명령에서 **Galera** 클러스터 노드가 동기화됨을 보고해야 합니다. **/var/log/mysql_cluster.log** 파일에 오류가 있는지 확인합니다.

r. 나머지 컨트롤러 노드에서 데이터베이스를 시작합니다.

```
$ /usr/bin/mysqld_safe --pid-file=/var/run/mysql/mysqld.pid --
socket=/var/lib/mysql/mysql.sock --datadir=/var/lib/mysql --log-
error=/var/log/mysql_cluster.log  --user=mysql --open-files-limit=16384 --wsrep-cluster-
address=gcomm://overcloud-controller-0,overcloud-controller-1,overcloud-controller-2 &
```

ws rep-cluster-address 옵션에 노드가 포함되면 새 클러스터에 노드를 추가하고 마스터의 콘텐츠를 동기화합니다.

s. 각 노드의 상태를 주기적으로 확인합니다.

```
# clustercheck
```

모든 노드가 동기화 작업을 완료하면 이 명령에서 각 노드의 **Galera** 클러스터 노드가 동기화됨을 보고해야 합니다.

t. 모든 노드에서 데이터베이스를 중지합니다.

```
$ mysqladmin shutdown
```

u. 서비스에 대한 각 노드에서 방화벽 규칙을 제거하여 데이터베이스에 대한 액세스를 복원합니다.

```
# sudo /sbin/iptables -D INPUT -d $MYSQLIP -p tcp --dport 3306 -j DROP
```

4. **Pacemaker** 구성을 복원합니다.

a. **Pacemaker** 아카이브를 부트스트랩 노드에 복사합니다.

b. 부트 스트랩 노드에 로그인합니다.

c. 구성 복원 명령을 실행합니다.

```
# pcs config restore pacemaker_controller_backup.tar.bz2
```

5. 파일 시스템을 복원하십시오.

a. 각 컨트롤러 노드의 백업 **tar** 파일을 임시 디렉터리에 복사하고 모든 데이터 압축을 해제합니다.

```
# mkdir /var/tmp/filesystem_backup/
# cd /var/tmp/filesystem_backup/
# mv <backup_file>.tar.gz .
# tar --xattrs --xattrs-include='*.*' -xvzf <backup_file>.tar.gz
```

참고

/ 디렉토리로 직접 추출하지 마십시오. 이렇게 하면 현재 파일 시스템이 재정의됩니다. 임시 디렉터리에 파일을 추출하는 것이 좋습니다.

b. **os-*-config** 파일을 복원하고 **os-collect-config** 를 다시 시작합니다.

```
# cp -rf /var/tmp/filesystem_backup/var/lib/os-collect-config/* /var/lib/os-collect-config/.
# cp -rf /var/tmp/filesystem_backup/usr/libexec/os-apply-config/* /usr/libexec/os-apply-config/.
# cp -rf /var/tmp/filesystem_backup/usr/libexec/os-refresh-config/* /usr/libexec/os-refresh-config/.
# systemctl restart os-collect-config
```

c. **Puppet hieradata** 파일을 복원합니다.

```
# cp -r /var/tmp/filesystem_backup/etc/puppet/hieradata /etc/puppet/hieradata
# cp -r /var/tmp/filesystem_backup/etc/puppet/hiera.yaml /etc/puppet/hiera.yaml
```

d. 구성 파일이 필요한 경우 이 디렉터리를 유지합니다.

6. **redis** 리소스를 복원합니다.

a. **Redis** 덤프를 각 컨트롤러 노드에 복사합니다.

b. **Redis** 덤프를 각 컨트롤러의 원래 위치로 이동합니다.

```
# mv dump.rdb /var/lib/redis/dump.rdb
```

c. **Redis** 디렉터리에 권한을 복원하십시오.

```
# chown -R redis: /var/lib/redis
```

7. 다음 디렉터리의 내용을 제거합니다.

```
# rm -rf /var/lib/config-data/puppet-generated/*
# rm /root/.ffu_workaround
```

8. **OpenStack Object Storage(swift)** 서비스에 대한 권한을 복원합니다.

```
# chown -R swift: /srv/node
# chown -R swift: /var/lib/swift
# chown -R swift: /var/cache/swift
```

9. 언더클라우드에 로그인하고 **OpenStack Platform 10** 배포에서 원래 **openstack overcloud deploy** 명령을 실행합니다. 원래 배포와 관련된 모든 환경 파일을 포함해야 합니다.

10. 배포가 완료될 때까지 기다립니다.

11. 오버클라우드 컨트롤 플레인 데이터를 복원한 후 각 관련 서비스가 활성화되어 올바르게 실행되고 있는지 확인합니다.

    a. 컨트롤러 노드의 고가용성 서비스:

    ```
    # pcs resource enable [SERVICE]
    # pcs resource cleanup [SERVICE]
    ```

    b. 컨트롤러 및 계산 노드의 시스템 서비스의 경우 다음을 수행합니다.

    ```
    # systemctl start [SERVICE]
    # systemctl enable [SERVICE]
    ```

다음 몇 섹션에서는 활성화해야 하는 서비스 참조를 제공합니다.

B.2. 복원된 고가용성 서비스

다음은 복원 후 **OpenStack Platform 10** 컨트롤러 노드에서 활성화해야 하는 고가용성 서비스 목록입니다. 이러한 서비스가 비활성화된 경우 다음 명령을 사용하여 활성화합니다.

```
# pcs resource enable [SERVICE]
# pcs resource cleanup [SERVICE]
```

컨트롤러 서비스
Galera
haproxy

컨트롤러 서비스
openstack-cinder-volume
rabbitmq
Redis

B.3. 복원된 컨트롤러 서비스

다음은 복원 후 **OpenStack Platform 10** 컨트롤러 노드에서 활성화해야 하는 핵심 **Systemd** 서비스 목록입니다. 이러한 서비스가 비활성화된 경우 다음 명령을 사용하여 활성화합니다.

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

컨트롤러 서비스
httpd
Memcached
neutron-dhcp-agent
neutron-l3-agent
neutron-metadata-agent
neutron-openvswitch-agent
neutron-ovs-cleanup
neutron-server
ntpd
openstack-aodh-evaluator
openstack-aodh-listener
openstack-aodh-notifier
openstack-ceilometer-central

컨트롤러 서비스
openstack-ceilometer-collector
openstack-ceilometer-notification
openstack-cinder-api
openstack-cinder-scheduler
openstack-glance-api
openstack-glance-registry
openstack-gnocchi-metricd
openstack-gnocchi-statsd
openstack-heat-api-cfn
openstack-heat-api-cloudwatch
openstack-heat-api
openstack-heat-engine
openstack-nova-api
openstack-nova-conductor
openstack-nova-consoleauth
openstack-nova-novncproxy
openstack-nova-scheduler
openstack-swift-account-auditor
openstack-swift-account-reaper
openstack-swift-account-replicator
openstack-swift-account
openstack-swift-container-auditor
openstack-swift-container-replicator

컨트롤러 서비스
openstack-swift-container-updater
openstack-swift-container
openstack-swift-object-auditor
openstack-swift-object-expirer
openstack-swift-object-replicator
openstack-swift-object-updater
openstack-swift-object
openstack-swift-proxy
openvswitch
os-collect-config
ovs-delete-transient-ports
ovs-vswitchd
ovsdb-server
pacemaker

B.4. 복원된 오버클라우드 계산 서비스

다음은 복원 후 **OpenStack Platform 10** 컴퓨팅 노드에서 활성화해야 하는 핵심 **Systemd** 서비스 목록입니다. 이러한 서비스가 비활성화된 경우 다음 명령을 사용하여 활성화합니다.

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

Compute 서비스
neutron-openvswitch-agent
neutron-ovs-cleanup

Compute 서비스
ntpd
openstack-ceilometer-compute
openstack-nova-compute
openvswitch
os-collect-config