Red Hat OpenStack Platform 13

Red Hat OpenStack Platform 업데이트 유지

Red Hat OpenStack Platform의 마이너 업데이트 수행

Last Updated: 2023-05-25

Red Hat OpenStack Platform 13 Red Hat OpenStack Platform 업데이트 유지

Red Hat OpenStack Platform의 마이너 업데이트 수행

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 문서에서는 Red Hat OpenStack Platform 13(Queens) 환경을 업데이트하는 절차를 설명합니다. 이 문서에서는 Red Hat Enterprise Linux 7에 설치된 컨테이너화된 OpenStack Platform 배포를 업데이트할 것으로 가정합니다.

차례

1장. 소개

이 문서에서는 Red Hat OpenStack Platform 13 환경을 최신 패키지 및 컨테이너로 업데이트하는 데 도움이 되는 워크플로를 제공합니다.

이 가이드에서는 다음 버전을 통해 업그레이드 경로를 제공합니다.

이전 오버클라우드 버전	새 오버클라우드 버전
Red Hat OpenStack Platform 13	Red Hat OpenStack Platform 13.z

1.1. 상위 수준의 워크플로

다음 표에서는 업그레이드 프로세스에 필요한 단계를 간략하게 설명합니다.

Step	설명
새 컨테이너 이미지 가져오기	OpenStack Platform 13 서비스의 최신 컨테이너 이미지가 포함된 새 환경 파일을 생성합니다.
언더클라우드 업데이트	언더클라우드를 최신 OpenStack Platform 13.z 버전으로 업데이트합니다.
오버클라우드 업데이트	오버클라우드를 최신 OpenStack Platform 13.z 버전으로 업데이트합니다.
Ceph Storage 노드 업데이트	모든 Ceph Storage 3 서비스를 업그레이드합니다.
업그레이드 종료	통합 명령을 실행하여 오버클라우드 스택을 새로 고칩니다.

1.2. 업데이트를 차단할 수 있는 알려진 문제

성공적인 마이너 버전 업데이트에 영향을 줄 수 있는 다음과 같은 알려진 문제를 검토하십시오.

Red Hat Ceph Storage 3의 마이너 업데이트로 인해 OSD 손상이 발생할 수 있습니다.

Red Hat Ceph Storage 3은 EL7에서 실행되는 컨테이너화된 배포의 docker를 사용합니다. BZ#1846830 의 ceph-ansible 수정에서는 Ceph 컨테이너를 제어하는 systemd 장치를 업데이트하여 systemd 장치를 실행하여 실행을 위해 docker 서비스가 실행 중이어야 합니다. 이 요구 사항은 안전한 업데이트 경로를 구현하고 docker 패키지의 제어되지 않는 업데이트에서 서비스 중단 또는 데이터 손상을 방지하는 데 필수적입니다.
ceph-ansible 패키지 업데이트로는 ceph-ansible 수정 사항이 적용되기에 충분하지 않습니다. 배포 플레이북을 다시 실행하여 컨테이너의 systemd 장치도 업데이트해야 합니다. director 중심 Ceph Storage 배포의 문제를 해결하는 방법에 대한 자세한 내용은 Red Hat Knowledgebase 솔루션에서 Red Hat Ceph Storage 3의 마이너 업데이트에 영향을 미치는 경우 OSD가 손상될 수 있습니다 .

OSP13 업데이트는 결국 성공하는 동안 실패하는 것처럼 보일 수 있습니다.

openstack overcloud update run 명령에 사용되는 python **tripleo-client** 는 업데이트 프로세스를 완료하기 전에 시간 초과될 수 있습니다. 그러면 **openstack overcloud update run** 명령에서 오류를 반환하지만 업데이트 프로세스가 완료될 때까지 백그라운드에서 계속 실행됩니다.
이 실패를 방지하려면 **tripleo-client/plugin.py** 파일의 **ttl** 매개변수 값을 편집하여 오버클라우드 노드를 업데이트하기 전에 **tripleo-client** 시간 초과 값을 늘립니다. 자세한 내용은 Red Hat Knowledgebase 솔루션 OSP 13 업데이트 프로세스가 백그라운드에서 실행되는 동안 실패하는 것으로 표시되고 성공적으로 완료됩니다.

rabbitmq 연결의 중단으로 인해 전체 동기화 후 데이터 플레인 손실이 발생했습니다.

BZ#1955538 버그에 설명된 데이터 플레인 연결 손실을 방지하기 위해 RHOSP 13 z10 (2004년 12월) 릴리스 이전의 릴리스에서 환경을 업데이트하려면 OSP13의 Red Hat Knowledgebase 솔루션이 업데이트 중에 데이터 플레인 축소를 생성할 수 있습니다.

ceph 업그레이드 중에 모든 OSD(및 기타 ceph 서비스)가 중단되었습니다.

Ceph를 사용하는 경우 다음 절차를 완료하기 전에 OSP13/RHCS3의 최신 패키지 Ceph 서비스가 오프라인으로 업데이트되는 동안 Red Hat 지식베이스 솔루션을 참조하십시오.

- 모든 컨트롤러 노드 업데이트
- 모든 HCI 컴퓨팅 노드 업데이트
- 모든 Ceph Storage 노드 업데이트

z11 업그레이드 후 Octavia 및 LB 문제

업데이트 중에 /var/lib/config-data/puppet-generated/octavia/conf.d/common/post-deploy.conf **var/lib/config-data/puppet-generated/octavia/conf.d/common/post-deploy.conf** 로 인해 로드 밸런싱 서비스(Octavia) 컨테이너가 지속적으로 다시 시작됩니다. 이 파일은 Amphora 배포 후 octavia 서비스를 구성하기 위해 Red Hat OpenStack Platform 13 라이프사이클 동안 도입되었습니다. 이 파일은 현재 **openstack overcloud update number** of update 단계에서 생성됩니다. 이 문제를 해결하려면 업데이트를 계속 진행해야 합니다. **openstack overcloud update converge** 명령을 실행하면 octavia 컨테이너가 정상적으로 시작됩니다. Red Hat OpenStack Platform 엔지니어링 팀은 현재 이 문제에 대한 해결 방법을 조사하고 있습니다.

(pymysql.err.InternalError) (1054, u"Unknown column 'pool.tls_certificate_id')에서 래핑된 DBAPIError 예외

로드 밸런싱 서비스(octavia)를 사용하고 RHOSP 13 z13 (8 10월 2020 유지 관리 릴리스) 이전 릴리스에서 업데이트하려는 경우 BZ#1927169 버그를 방지하려면 로드 밸런싱 서비스를 업그레이드하는 데이터베이스 마이그레이션을 올바른 순서로 실행해야 합니다. 나머지 컨트롤 플레인을 업데이트하기 전에 부트스트랩 컨트롤러 노드를 업데이트해야 합니다.

1. 현재 유지 관리 릴리스를 확인하려면 다음 명령을 실행합니다.

```
$ cat /etc/rhosp-release
```

2. 언더클라우드 노드에서 부트스트랩 컨트롤러 노드를 확인하려면 다음 명령을 실행하고 < **any_controller_node_IP_address** >를 배포에 있는 컨트롤러 노드의 IP 주소로 교체합니다.

```
$ ssh heat-admin@<any_controller_node_IP_address> sudo hiera -c
/etc/puppet/hiera.yaml octavia_api_short_bootstrap_node_name
```

3. 언더클라우드 노드에서 **openstack overcloud update run** 명령을 실행하여 부트스트랩 컨트롤러 노드를 업데이트합니다.

```
$ openstack overcloud update run --nodes <bootstrap_node_name>
```

13z16으로의 마이너 업데이트 "일정을 찾을 수 없음"으로 실패했습니다.

Red Hat OpenStack Platform 13z16 오버클라우드 노드 업데이트를 다시 시작하면 **Unable to find 제약 조건이** 발생할 수 있습니다. 이 오류는 업데이트 중에 RabbitMQ 버전의 불일치로 인해 발생합니다. 새 RabbitMQ 버전을 시작할 수 있도록 하려면 오버클라우드에 존재할 수 있는 pacemaker 금지 사항을 지웁니다.
이 문제에 대한 자세한 내용은 Red Hat Knowledgebase 솔루션에서 OSP13z16 컨트롤러 업데이트를 다시 시작할 수 없음을 참조하십시오.

컨트롤러에서 ceph-mon을 중지할 수 없습니다. 이러한 컨테이너 없음: ceph-mon controller-2

Red Hat Ceph Storage 버전 3.3 z5 또는 이전 버전을 사용하고 docker 패키지를 docker-1.13.1-209로 업데이트하면 RHOSP 13 업데이트가 실패합니다. RHOSP 13 업데이트는 docker 패키지를 업데이트하기 전에 ceph-mon 컨테이너를 중지하지 않습니다. 이로 인해 새 ceph-mon 컨테이너가 시작되지 않도록 하는 고립된 ceph-mon 프로세스가 생성됩니다.
이 문제에 대한 자세한 내용은 컨트롤러 업데이트 중에 Red Hat OpenStack Platform 13.z12 및 이전 버전에서 Red Hat Knowledgebase 솔루션 업데이트 Red Hat OpenStack Platform 13.z12 및 이전 버전이 실패할 수 있습니다.

1.3. 문제 해결

- 업데이트 프로세스가 예상보다 오래 걸리는 경우 오류와 함께 시간이 초과될 수 있습니다. **소켓은 이미 닫힙니다**. 이는 설정된 기간 후에 언더클라우드의 인증 토큰이 만료되도록 설정되어 있기 때문에 발생할 수 있습니다. 자세한 내용은 대규모 배포에 대한 권장 사항을 참조하십시오.

2장. 컨테이너 이미지 소스 업데이트

이 장에서는 Red Hat OpenStack Platform의 새 오버클라우드 컨테이너 이미지로 레지스트리 소스를 업데이트하는 방법을 설명합니다.

컨테이너 이미지 소스를 업데이트하기 전에 고려 사항

컨테이너 이미지 소스를 하나의 레지스트리 유형에서 다른 레지스트리로 변경하려면 현재 컨테이너 이미지 환경 파일에서 네임스페이스 및 접두사를 업데이트하고 **openstack overcloud deploy** 명령을 실행하여 다음 작업을 완료하기 전에 네임스페이스를 변경해야 합니다.

- 2.4절. “Red Hat 레지스트리를 원격 레지스트리 소스로 사용”
- 2.5절. “언더클라우드를 로컬 레지스트리로 사용”
- 2.6절. “Satellite 서버를 레지스트리로 사용”

2.1. 레지스트리 방법

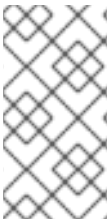
Red Hat OpenStack Platform은 다음과 같은 레지스트리 유형을 지원합니다.

원격 레지스트리

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

로컬 레지스트리

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



참고

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

Satellite Server

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

목록에서 방법을 선택하고 레지스트리 세부 정보를 계속 구성합니다.

참고

다중 아키텍처 클라우드를 빌드하는 경우 로컬 레지스트리 옵션이 지원되지 않습니다.

2.2. 컨테이너 이미지 준비 명령 사용

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

오버클라우드용 컨테이너 이미지 환경 파일 생성

openstack overcloud container image prepare 명령의 주요 사용 중 하나는 오버클라우드에서 사용하는 이미지 목록이 포함된 환경 파일을 생성하는 것입니다. **openstack overcloud deploy** 와 같은 오버클라우드 배포 명령에 이 파일을 추가합니다. **openstack overcloud container image prepare** 명령은 이 기능에 대해 다음 옵션을 사용합니다.

--output-env-file

결과 환경 파일 이름을 정의합니다.

다음 스니펫은 이 파일의 내용의 예입니다.

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

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

가져오기 메서드의 컨테이너 이미지 목록 생성

OpenStack Platform 컨테이너 이미지를 다른 레지스트리 소스로 가져오려면 이미지 목록을 생성할 수 있습니다. 목록의 구문은 주로 컨테이너 이미지를 언더클라우드의 컨테이너 레지스트리로 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 대해 다음 옵션을 사용합니다.

--output-images-file

가져오기 목록의 결과 파일 이름을 정의합니다.

다음은 이 파일의 내용의 예입니다.

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

컨테이너 이미지의 네임스페이스 설정

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

--namespace

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

--prefix

이미지 이름 앞에 추가할 접두사를 정의합니다.

결과적으로 director는 다음 형식을 사용하여 이미지 이름을 생성합니다.

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

컨테이너 이미지 태그 설정

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

--tag

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

--tag-from-label

지정된 컨테이너 이미지 레이블의 값을 사용하여 모든 이미지에 대해 버전 지정된 태그를 검색하고 가져옵니다. director는 **--tag** 에 설정한 값으로 태그된 각 컨테이너 이미지를 검사한 다음 컨테이너 이미지 레이블을 사용하여 director가 레지스트리에서 가져오는 새 태그를 구성합니다. 예를 들어 **--tag-from-label {version}-{release}** 를 설정하면 director는 **version** 및 **release** 레이블을 사용하여 새 태그를 구성합니다. 한 컨테이너의 경우 **version** 을 **13.0** 으로 설정하고 **release** 를 **34** 으로 설정할 수 있으며, 이로 인해 태그 **13.0-34** 가 생성됩니다.

중요

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

2.3. 추가 서비스의 컨테이너 이미지

director는 핵심 OpenStack Platform Services를 위한 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 파일 및 해당 컨테이너 이미지를 포함합니다.

-e

추가 컨테이너 이미지를 활성화하려면 환경 파일을 포함합니다.

다음 표에서는 컨테이너 이미지를 사용하는 추가 서비스 샘플 목록과 **/usr/share/openstack-tripleo-heat-templates** 디렉터리 내의 해당 환경 파일 위치를 제공합니다.

Service	환경 파일
Ceph Storage	**environments/ceph-ansible/ceph-ansible.yaml**
collectd	**environments/services-docker/collectd.yaml**
구문 분석	**environments/services-docker/congress.yaml**
fluentd	**environments/services-docker/fluentd.yaml**

Service	환경 파일
OpenStack Bare Metal(ironic)	**environments/services-docker/ironic.yaml**
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 부분에서는 추가 서비스를 포함한 예제를 제공합니다.

Ceph Storage

오버클라우드를 사용하여 Red Hat Ceph Storage 클러스터를 배포하는 경우 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible.yaml** 환경 파일을 포함해야 합니다. 이 파일을 사용하면 오버클라우드에서 구성 가능 컨테이너 서비스를 사용할 수 있으며 director는 이미지를 준비하기 위해 이러한 서비스가 활성화되어 있는지 확인해야 합니다.

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

--set ceph_namespace

Ceph Storage 컨테이너 이미지의 네임스페이스를 정의합니다. 이 함수는 **--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 함수는 **--tag** 옵션과 유사합니다. **--tag-from-label** 이 지정되면 이 태그에서 시작되는 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 director가 이미지를 준비할 수 있도록 **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml** 환경 파일을 포함해야 합니다. 다음 스니펫은 이 환경 파일을 포함하는 방법에 대한 예입니다.

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

OpenStack Data Processing(sahara)

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

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

OpenStack Neutron SR-IOV

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

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

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

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

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

OpenStack Shared File System(manila)

manila-{backend-name}-config.yaml 형식을 사용하여 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 파일을 사용자 정의 및 배포하는 방법에 대한 자세한 내용은 다음 리소스를 참조하십시오.

- *공유 파일 시스템 서비스에 대한 NFS 백엔드 가이드를 통해 CephFS* 에서 업데이트된 환경 배포
- 공유 파일 시스템 서비스 *용 NetApp 백엔드 가이드에 NetApp Back Ends를 사용하여 공유 파일 시스템 서비스*배포
- 공유 파일 시스템 서비스의 CephFS 백엔드 *가이드에서 CephFS 백엔드를 사용하여 공유 파일 시스템 서비스*배포

2.4. RED HAT 레지스트리를 원격 레지스트리 소스로 사용

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

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

절차

1. 오버클라우드 배포의 **registry.redhat.io** 에서 이미지를 직접 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.
   - 사용자 지정 역할 파일을 포함하려면 **-r** 옵션을 사용합니다.
   - Ceph Storage를 사용하는 경우 추가 매개변수를 포함하여 Ceph Storage 컨테이너 이미지 위치를 정의합니다. **--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. 이 파일을 배포와 함께 포함합니다.

참고

openstack overcloud deploy 명령을 실행하기 전에 원격 레지스트리에 로그인해야 합니다.

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

레지스트리 구성이 준비되었습니다.

2.5. 언더클라우드를 로컬 레지스트리로 사용

오버클라우드 컨테이너 이미지를 저장하도록 언더클라우드에 로컬 레지스트리를 구성할 수 있습니다.

director를 사용하여 **registry.redhat.io** 에서 각 이미지를 가져와서 언더클라우드에서 실행되는 **docker-distribution** 레지스트리로 각 이미지를 푸시할 수 있습니다. 오버클라우드 생성 프로세스 중에 director를 사용하여 오버클라우드를 생성하면 노드가 언더클라우드 **docker-distribution** 레지스트리에서 관련 이미지를 가져옵니다.

이렇게 하면 컨테이너 이미지의 네트워크 트래픽이 내부 네트워크에 연결되어 있어 외부 네트워크 연결을 제한하지 않고 배포 프로세스의 속도를 높일 수 있습니다.

절차

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

```
<REGISTRY_IP_ADDRESS>:8787
```

   이전에 **undercloud.conf** 파일의 **local_ip** 매개변수로 설정한 언더클라우드의 IP 주소를 사용합니다. 아래 명령의 경우 주소는 **192.168.24.1:8787** 인 것으로 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.
- 사용자 지정 역할 파일을 포함하려면 **-r** 옵션을 사용합니다.
- Ceph Storage를 사용하는 경우 추가 매개변수를 포함하여 Ceph Storage 컨테이너 이미지 위치를 정의합니다. **--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag**.

4. 다음 두 파일이 생성되었는지 확인합니다.
   - remote 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. 이 파일을 배포와 함께 포함합니다.
5. 원격 레지스트리에서 컨테이너 이미지를 가져와서 언더클라우드 레지스트리로 푸시합니다.

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

필요한 이미지를 가져오는 데 네트워크 및 언더클라우드 디스크의 속도에 따라 다소 시간이 걸릴 수 있습니다.

참고

컨테이너 이미지는 약 10GB의 디스크 공간을 사용합니다.

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

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

참고

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

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

특정 이미지에 대한 태그 목록을 보려면 **skopeo** 명령을 사용합니다.

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

태그된 이미지를 확인하려면 **skopeo** 명령을 사용합니다.

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

레지스트리 구성이 준비되었습니다.

2.6. 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 가져오는 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.
   - 사용자 지정 역할 파일을 포함하려면 **-r** 옵션을 사용합니다.
   - Ceph Storage를 사용하는 경우 추가 매개변수를 포함하여 Ceph Storage 컨테이너 이미지 위치를 정의합니다. **--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag**.

참고

이 버전의 **openstack overcloud container image prepare** 명령은 **registry.redhat.io** 에서 레지스트리를 대상으로 이미지 목록을 생성합니다. 이후 단계에서 사용된 **openstack overcloud container image prepare** 명령과 다른 값을 사용합니다.

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

   이를 통해 Satellite 서버로 가져오는 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 컨테이너**)을 생성합니다.

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

이 사용자 지정 제품에 이미지를 저장합니다.

6. 제품에 기본 컨테이너 이미지를 추가합니다.

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

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

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

8. 컨테이너 이미지를 동기화합니다.

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

Satellite 서버가 동기화를 완료할 때까지 기다립니다.

참고

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

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

10. **기본** 이미지에 사용 가능한 태그를 확인합니다.

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

그러면 OpenStack Platform 컨테이너 이미지에 대한 태그가 표시됩니다.

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

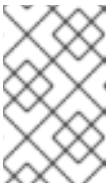
```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=satellite6.example.com:5000 \
  --prefix=acme-osp13_containers- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

참고

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

이 명령을 실행할 때 다음 데이터를 포함합니다.

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



참고

Red Hat Satellite 버전 6.10을 사용하는 경우 포트를 지정할 필요가 없습니다. 기본 포트 **443** 이 사용됩니다. 자세한 내용은 "Red Hat Satellite 6.10에 RHOSP13 배포를 조정하는 방법" 을 참조하십시오.

- **--prefix=** - 접두사는 레이블에 대한 Satellite 6 규칙을 기반으로 하며, 이 규칙은 소문자를 사용하고 밑줄을 위한 공백을 대체합니다. 접두사는 콘텐츠 뷰 사용 여부에 따라 다릅니다.
  - 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. 예: **acme-osp13_containers-**.
- **--tag-from-label {version}-{release}** - 각 이미지의 최신 태그를 식별합니다.
- **-e** - 선택적 서비스에 대한 모든 환경 파일을 포함합니다.
- **-r** - 사용자 지정 역할 파일을 포함합니다.
- **--set ceph_namespace**,**--set ceph_image**,**--set ceph_tag** - Ceph Storage를 사용하는 경우 추가 매개변수를 포함하여 Ceph Storage 컨테이너 이미지 위치를 정의합니다. 이제 **ceph_image**에는 Satellite별 접두사가 포함됩니다. 이 접두사는 **--prefix** 옵션과 동일한 값입니다. 예를 들면 다음과 같습니다.

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

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

12. **overcloud_images.yaml** 파일에는 Satellite 서버의 이미지 위치가 포함되어 있습니다. 이 파일을 배포와 함께 포함합니다.

레지스트리 구성이 준비되었습니다.

2.7. 다음 단계

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

이제 업데이트를 위해 오버클라우드를 준비할 수 있습니다.

3장. 언더클라우드 업그레이드

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

3.1. 언더클라우드의 마이너 업데이트 수행

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

절차

1. **stack** 사용자로 director에 로그인합니다.
2. **python-tripleoclient** 패키지 및 해당 종속 항목을 업데이트하여 마이너 버전 업데이트에 대한 최신 스크립트가 있는지 확인합니다.

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

3. director는 **openstack undercloud upgrade** 명령을 사용하여 Undercloud 환경을 업데이트합니다. 명령을 실행합니다.

```
$ openstack undercloud upgrade
```

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

```
$ sudo reboot
```

6. 노드가 부팅될 때까지 기다립니다.

3.2. 오버클라우드 이미지 업데이트

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

사전 요구 사항

- 언더클라우드를 최신 버전으로 업데이트했습니다.

절차

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

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

2. 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 ~
```

3. 언더클라우드 노드에서 언더클라우드 인증 정보를 가져옵니다.

```
$ source ~/stackrc
```

4. 최신 이미지를 director로 가져옵니다.

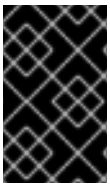
```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

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 템플릿이 있는 경우에만 OpenStack Platform 13 이미지를 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.

3.3. 언더클라우드 업그레이드 후 참고 사항

- **스택** 사용자 홈 디렉터리에서 로컬 코어 템플릿 세트를 사용하는 경우 "사용자 지정 코어 Heat 템플릿 사용"에서 권장 워크플로를 사용하여 템플릿을 업데이트해야 합니다. 오버클라우드를 업그레이드하기 전에 로컬 복사본을 업데이트해야 합니다.

3.4. 다음 단계

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

4장. 오버클라우드 업데이트

이 프로세스는 오버클라우드를 업데이트합니다.

사전 요구 사항

- 언더클라우드를 최신 버전으로 업데이트했습니다.

4.1. 오버클라우드 업데이트 속도 향상

오버클라우드 업데이트 프로세스의 속도를 높이기 위해 **DockerPuppetProcessCount** heat 매개변수를 구성하고, 삭제된 데이터베이스 항목을 아카이브하고, 업데이트를 수행하기 전에 오버클라우드 노드에서 필요한 패키지를 다운로드할 수 있습니다.

대규모 OpenStack 배포의 업데이트 프로세스 속도를 높이는 방법에 대한 자세한 내용은 Red Hat Knowledgebase 문서 Openstack Director Node Performance Tuning for large deployment를 참조하십시오.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **container-puppet** 에서 구성 파일을 생성하는 데 사용하는 동시 프로세스 수를 늘리려면 **DockerPuppetProcessCount** 매개변수를 구성해야 합니다.
   a. **templates** 디렉터리에 **updates-environment.yaml** 이라는 환경 파일을 생성합니다.

```
$ touch ~/templates/updates-environment.yaml
```

   b. 파일을 편집하고 다음 콘텐츠를 추가합니다.

```
parameter_defaults:
  DockerPuppetProcessCount: 8
```

   c. **openstack overcloud update prepare**,**openstack overcloud ceph-upgrade run**, **openstack overcloud update converge** 명령을 실행할 때 **-e** 옵션을 사용하여 이 환경 파일을 포함합니다.
4. 컨트롤러 노드에서 삭제된 데이터베이스 항목을 보관합니다.
   a. 오버클라우드에서 컨트롤러 노드의 모든 인스턴스를 나열합니다.

```
$ source ~/overcloudrc
$ openstack server list
```

   b. **nova_api_cron** 컨테이너를 실행 중인 컨트롤러 노드에 로그인합니다.

```
ssh heat-admin@<controller_ip>
```

- & **lt;controller name 또는 IP&** gt;를 컨트롤러 노드의 IP 주소로 바꿉니다.

c. 삭제된 아카이브 데이터베이스 항목:

```
$ sudo docker exec -u 42436 -ti nova_api_cron bash
$ nova-manage db archive_deleted_rows --max_rows 1000
$ exit
```

5. 모든 오버클라우드 노드에서 업데이트에 필요한 모든 패키지를 다운로드하려면 다음 단계를 완료하십시오.

a. 오버클라우드의 정적 인벤토리 파일을 생성합니다.

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

b. 다음 Ansible 플레이북을 생성합니다.

```
$ cat > ~/yum-download-only.yaml <<'EOF'
- hosts: all
  gather_facts: false
  tasks:
    - name: Pre-download all packages on all overcloud nodes
      shell:
        yum upgrade -y --downloadonly
      become: true
EOF
```

c. **yum-download-only.yaml** Ansible 플레이북을 실행합니다.

```
$ ansible-playbook \
-i ~/inventory.yaml \
-f 20 ~/yum-download-only.yaml \
--limit Controller,Compute,CephStorage
```

4.2. 사용자 정의 역할 고려 사항

배포에 사용자 정의 역할이 포함된 경우 역할 파일의 다음 값을 확인합니다.

- 사용자 지정 역할 파일을 **/usr/share/openstack-tripleo-heat-templates/roles** 디렉터리의 최신 파일과 비교합니다. 환경에 대한 관련 역할에 대한 **RoleParametersDefault** 섹션의 새 매개변수를 사용자 지정 역할 파일의 동등한 역할로 추가합니다.

- DPDK(Data Plane Development Kit)를 사용하고 13.4 이하에서 업그레이드하는 경우 OVS-DPDK 서비스가 포함된 역할에 다음과 같은 필수 매개변수도 포함되어 있는지 확인합니다.

```
RoleParametersDefault:
  VhostuserSocketGroup: "hugetlbfs"
  TunedProfileName: "cpu-paritioning"
  NovaLibvirtRxQueueSize: 1024
  NovaLibvirtTxQueueSize: 1024
```

4.3. 오버클라우드 업데이트 준비 실행

업데이트를 수행하려면 다음 작업을 수행하는 **openstack overcloud update prepare** 명령을 실행해야 합니다.

- 오버클라우드 계획을 OpenStack Platform 13으로 업데이트
- 업데이트를 위해 노드를 준비합니다

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. update 준비 명령을 실행합니다.

```
$ openstack overcloud update prepare \
  --templates \
  -r <ROLES DATA FILE> \
  -n <NETWORK DATA FILE> \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/updates-environment.yaml \
  -e <ENVIRONMENT FILE> \
  -e <ENVIRONMENT FILE> \
  --stack <STACK_NAME>
  ...
```

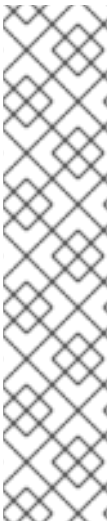
환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일(**-e**).
- 새 컨테이너 이미지 위치(**-e**)가 포함된 환경 파일입니다. update 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하는 대신 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
- 고유한 사용자 지정 역할을 사용하는 경우 사용자 지정 역할(**roles_data**) 파일(**-r**)을 포함합니다.
- 사용자 지정 네트워크를 사용하는 경우 구성 가능 네트워크(**network_data**) 파일(**-n**)을 포함합니다.
- 고가용성 클러스터를 배포하는 경우 업데이트 준비 명령에 **--ntp-server** 옵션을 포함하거나 환경 파일에 **NtpServer** 매개변수 및 값을 포함합니다.
- 오버클라우드 스택 이름이 기본 오버클라우드와 다른 경우 업데이트 준비 명령에 **--stack** 옵션을 **포함하고 <STACK_NAME>** 을 스택 이름으로 교체합니다.

3. 업데이트가 완료될 때까지 기다립니다.

4.4. CEPH STORAGE 클러스터 업데이트

이 프로세스는 Ceph Storage 클러스터를 업데이트합니다. 이 프로세스에서는 Red Hat Ceph Storage 3 클러스터에 대한 업데이트를 수행하기 위해 **openstack overcloud ceph-upgrade run** 명령을 실행해야 합니다.



참고

다음 Ansible 조합과 **ceph-ansible** 이 지원됩니다.

- **Ansible-2.6** (**ceph-ansible-3.2**포함)
- **ansible-2.4** with **ceph-ansible-3.1**

사용자 환경에 **ceph-ansible-3.1** 이 있는 **ansible-2.6** 이 있는 경우 **ceph-ansible** 을 최신 버전으로 업데이트합니다.

```
# subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
# subscription-manager repos --enable=rhel-7-server-ansible-2.6-rpms
# yum update ceph-ansible
```

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. Ceph Storage update 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/updates-environment.yaml
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일 (**-e**)
- 컨테이너 이미지 위치(**-e**)가 포함된 환경 파일입니다. update 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하는 대신 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
- 해당하는 경우 사용자 지정 역할(**roles_data**) 파일(**--roles-file**)
- 해당되는 경우 구성 가능 네트워크(**network_data**) 파일(**--networks-file**)

3. Ceph Storage 노드 업데이트가 완료될 때까지 기다립니다.

참고

프로세스 중에 Heat 스택이 시간 초과되면 Ceph 노드의 순차적 업데이트 중에 **openstack overcloud ceph-upgrade run** 이 시간 초과되는 것으로 표시되는 Red Hat 지식베이스 문서를 참조하십시오.

4.5. 모든 컨트롤러 노드 업데이트

컨트롤러 노드를 RHOSP(Red Hat OpenStack Platform) 13 버전으로 업데이트하려면 **openstack overcloud update run** 명령에 **--nodes Controller** 옵션을 포함합니다. **--nodes Controller** 옵션은 업데이트 작업을 컨트롤러 노드로만 제한합니다.

경고

Ceph를 사용하는 경우 OSP13/RHCS3에서 최신 패키지로의 마이너 업데이트 중에 Ceph 서비스가 오프라인으로 전환되는 동안 Red Hat 지식베이스 솔루션을 참조하십시오. Ceph 서비스가 오프라인으로 전환되고 버그 BZ#1910842 의 버그를 방지하기 위해 컨트롤러 노드를 업데이트하기 전에 수동으로 다시 시작해야 합니다.

사전 요구 사항

- 로드 밸런싱 서비스(octavia)를 사용하고 RHOSP 13 z13 (8 10월 2020 유지 관리 릴리스) 이전 릴리스에서 업데이트하려는 경우 BZ#1927169 버그를 방지하려면 로드 밸런싱 서비스를 업그레이드하는 데이터베이스 마이그레이션을 올바른 순서로 실행해야 합니다. 나머지 컨트롤 플레인을 업데이트하기 전에 부트스트랩 컨트롤러 노드를 업데이트해야 합니다.

  1. 현재 유지 관리 릴리스를 확인하려면 다음 명령을 실행합니다.

     ```
     $ cat /etc/rhosp-release
     ```

  2. 언더클라우드 노드에서 부트스트랩 컨트롤러 노드를 확인하려면 다음 명령을 실행하고 < **any_controller_node_IP_address** >를 배포에 있는 컨트롤러 노드의 IP 주소로 교체합니다.

     ```
     $ ssh heat-admin@<any_controller_node_IP_address> sudo hiera -c
     /etc/puppet/hiera.yaml octavia_api_short_bootstrap_node_name
     ```

  3. 언더클라우드 노드에서 **openstack overcloud update run** 명령을 실행하여 부트스트랩 컨트롤러 노드를 업데이트합니다.

     ```
     $ openstack overcloud update run --nodes <bootstrap_node_name>
     ```

절차

1. **stackrc** 파일을 소싱합니다.

   ```
   $ source ~/stackrc
   ```

2. update 명령을 실행합니다.

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

3. 컨트롤러 노드 업데이트가 완료될 때까지 기다립니다.

4.6. 모든 컴퓨팅 노드 업데이트

이 프로세스에서는 모든 컴퓨팅 노드가 최신 OpenStack Platform 13 버전으로 업데이트됩니다. 프로세스에는 **openstack overcloud update run 명령을 실행하고** 작업을 컴퓨팅 노드로만 제한하는 **--nodes Compute** 옵션을 포함해야 합니다.

병렬화 고려 사항

많은 수의 컴퓨팅 노드를 업데이트하여 성능을 향상시키는 경우 **--nodes Compute** 옵션을 사용하여 20개의 노드 배치에서 동시에 **openstack overcloud update run** 명령을 실행할 수 있습니다. 예를 들어 배포에 80개의 컴퓨팅 노드가 있는 경우 다음 명령을 실행하여 컴퓨팅 노드를 병렬로 업데이트할 수 있습니다.

```
$ openstack overcloud update run --nodes 'Compute[0:19]' > update-compute-0-19.log 2>&1 &
$ openstack overcloud update run --nodes 'Compute[20:39]' > update-compute-20-39.log 2>&1 &
$ openstack overcloud update run --nodes 'Compute[40:59]' > update-compute-40-59.log 2>&1 &
$ openstack overcloud update run --nodes 'Compute[60:79]' > update-compute-60-79.log 2>&1 &
```

'Compute[0:19]', **'Compute[20:39]'**, **'Compute[40:59]'**, 'Compute[40:59]' 및 **'Compute[60:79]'** 방식으로 노드 공간을 무작위로 분할하는 방법은 임의적이고 노드가 각 배치에서 업데이트되는 순서를 제어할 수 없습니다.

특정 컴퓨팅 노드를 업데이트하려면 쉼표로 구분된 일괄 처리로 업데이트할 노드를 나열합니다.

```
$ openstack overcloud update run --nodes <Compute0>,<Compute1>,<Compute2>,<Compute3>
```

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. update 명령을 실행합니다.

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

3. 컴퓨팅 노드 업데이트가 완료될 때까지 기다립니다.

4.7. 모든 HCI 컴퓨팅 노드 업데이트

이 프로세스에서는 HCI(Hyperconverged Infrastructure) 컴퓨팅 노드를 업데이트합니다. 프로세스에는 **openstack overcloud update run** 명령을 실행하고 **--nodes ComputeHCI** 옵션을 포함하여 HCI 노드로만 작업을 제한합니다.

경고

Ceph를 사용하는 경우 다음 버그가 발생하지 않도록 OSP13/RHCS3에서 최신 패키지로의 마이너 업데이트 중에 Ceph 서비스가 오프라인으로 전환되는 동안 Red Hat 지식베이스 솔루션을 참조하십시오.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. update 명령을 실행합니다.

```
$ openstack overcloud update run --nodes ComputeHCI
```

3. 노드 업데이트가 완료될 때까지 기다립니다.

4.8. 모든 CEPH STORAGE 노드 업데이트

이 프로세스는 Ceph Storage 노드를 업데이트합니다. 이 프로세스에는 **openstack overcloud update run** 명령을 실행하고 **--nodes CephStorage** 옵션을 포함하여 작업을 Ceph Storage 노드로만 제한합니다.

경고

Ceph를 사용하는 경우 다음 버그가 발생하지 않도록 OSP13/RHCS3에서 최신 패키지로의 마이너 업데이트 중에 Ceph 서비스가 오프라인으로 전환되는 동안 Red Hat 지식베이스 솔루션을 참조하십시오.

절차

1. **stackrc** 파일을 소싱합니다.

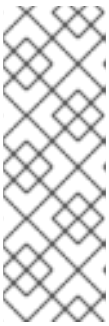
```
$ source ~/stackrc
```

2. 그룹 노드를 업데이트합니다.
그룹의 모든 노드를 업데이트하려면 다음을 수행합니다.

```
$ openstack overcloud update run --nodes <GROUP_NAME>
```

그룹의 단일 노드를 업데이트하려면 다음을 수행합니다.

```
$ openstack overcloud update run --nodes <GROUP_NAME> [NODE_INDEX]
```



참고

노드를 개별적으로 업데이트하도록 선택하는 경우 모든 노드를 업데이트해야 합니다.

그룹에서 첫 번째 노드의 인덱스는 0(0)입니다. 예를 들어 **CephStorage** 라는 그룹에서 첫 번째 노드를 업데이트하려면 명령은 다음과 같습니다.

OpenStack overcloud update run --nodes CephStorage[0]

3. 노드 업데이트가 완료될 때까지 기다립니다.

4.9. 업데이트 종료

업데이트하려면 오버클라우드 스택을 업데이트하기 위한 최종 단계가 필요합니다. 이렇게 하면 스택 리소스 구조가 Red Hat OpenStack Platform 13의 일반 배포와 일치시킬 수 있으며 향후 표준 **openstack overcloud deploy** 기능을 수행할 수 있습니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 업데이트 완료 명령을 실행합니다.

```
$ openstack overcloud update converge \
  --templates \
```

```
  --stack <STACK_NAME> \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/updates-environment.yaml \
  -e <ENVIRONMENT FILE>
  ...
```

환경과 관련된 다음 옵션을 포함합니다.

- 사용자 지정 구성 환경 파일 (**-e**)
- 새 컨테이너 이미지 위치(**-e**)가 포함된 환경 파일입니다. update 명령은 **--container-registry-file** 사용에 대한 경고를 표시할 수 있습니다. 이 옵션은 컨테이너 이미지 환경 파일에 **-e** 를 사용하는 대신 더 이상 사용되지 않으므로 이 경고를 무시할 수 있습니다.
- 해당하는 경우 사용자 지정 역할(**roles_data**) 파일(**--roles-file**)
- 해당되는 경우 구성 가능 네트워크(**network_data**) 파일(**--networks-file**)
- 오버클라우드 스택 이름이 기본 이름 **overcloud** 와 다른 경우 업데이트 준비 명령에 **--stack** 옵션을 포함하고 < **STACK_NAME** >을 오버클라우드 스택 이름으로 교체해야 합니다.

3. 업데이트 완료가 완료될 때까지 기다립니다.

5장. 오버클라우드 재부팅

마이너 Red Hat OpenStack 버전 업데이트 후 오버클라우드를 재부팅합니다. 재부팅은 연결된 커널, 시스템 수준 및 컨테이너 구성 요소 업데이트를 통해 노드를 새로 고칩니다. 이러한 업데이트는 성능 및 보안상의 이점을 제공할 수 있습니다.

다음 재부팅 절차를 수행하기 위해 다운타임을 계획합니다.

5.1. 컨트롤러 노드 및 구성 가능 노드 재부팅

다음 절차에서는 구성 가능 역할을 기반으로 컨트롤러 노드 및 독립 실행형 노드를 재부팅합니다. 이는 컴퓨팅 노드 및 Ceph Storage 노드를 제외합니다.

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 가능 노드에 대해 이 단계를 반복합니다.

5.2. 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
```

5.3. 컴퓨팅 노드 재부팅

컴퓨팅 노드를 재부팅하려면 다음 워크플로우가 포함됩니다.

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

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.

2. 재부팅할 컴퓨팅 노드를 확인하려면 모든 컴퓨팅 노드를 나열합니다.

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

3. 오버클라우드에서 컴퓨팅 노드를 선택하고 비활성화합니다.

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

4. 컴퓨팅 노드에 모든 인스턴스를 나열합니다.

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

5. 인스턴스를 마이그레이션합니다. 마이그레이션 전략에 대한 자세한 내용은 컴퓨팅 노드 간 가상 머신 마이그레이션을 참조하십시오.

6. 컴퓨팅 노드에 로그인하고 재부팅합니다.

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

7. 노드가 부팅될 때까지 기다립니다.

8. 컴퓨팅 노드를 활성화합니다.

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

9. 컴퓨팅 노드가 활성화되어 있는지 확인합니다.

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

5.4. 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 UUID를 나열합니다.

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

재부팅하려는 컴퓨팅 노드의 UUID를 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.

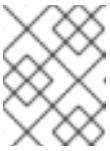
```
(overcloud) $ openstack server migrate [instance-id] --live [target-host]--wait
```

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 노드를 재부팅할 때까지 이 프로세스를 반복합니다.
16. 완료되면 Ceph MON 또는 컨트롤러 노드에 로그인하고 클러스터 재조정을 다시 활성화합니다.

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

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

```
$ sudo ceph status
```

6장. 업데이트 후 작업 수행

마이너 Red Hat OpenStack 버전 업데이트 후 업데이트 후 관련 작업을 수행하여 환경이 완전히 지원되고 향후 작업을 수행할 수 있는지 확인해야 합니다.

6.1. OCTAVIA 배포에 대한 고려 사항

배포에서 Octavia 서비스를 사용하는 경우 실행 중인 로드 밸런싱 amphora 인스턴스를 새 이미지로 업데이트해야 합니다.

amphora 이미지를 업데이트하려면 로드 밸런서를 통해 장애 조치한 다음 로드 밸런서가 활성 상태가 될 때까지 기다려야 합니다. 로드 밸런서가 다시 활성화되면 새 이미지를 실행합니다.

자세한 내용은 *Load Balancing-as-a-Service 가이드에서 Octavia를 사용하여 로드 밸런싱 서비스 인스턴스 실행* 을 참조하십시오.