



Red Hat OpenStack Platform 17.1

DCN(Distributed Compute Node) 아키텍처 배포

Red Hat OpenStack Platform의 엣지 및 스토리지 구성

Red Hat OpenStack Platform 17.1 DCN(Distributed Compute Node) 아키텍처 배포

Red Hat OpenStack Platform의 엣지 및 스토리지 구성

OpenStack Team
rhos-docs@redhat.com

법적 공지

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

초록

heat 스택 분리를 사용하여 엣지 사이트 작동을 위해 DCN(Distributed Compute Node) 아키텍처를 사용하여 RHOSP(Red Hat OpenStack Platform)를 배포할 수 있습니다. 각 사이트에는 Image 서비스(glance) 멀티 저장소에 대한 자체 Ceph 스토리지 백엔드가 있을 수 있습니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체	4
RED HAT 문서에 관한 피드백 제공	5
1장. DCN 이해	6
1.1. 분산 컴퓨팅 노드 아키텍처의 필수 소프트웨어	6
1.2. 멀티 스택 설계	7
1.3. DCN 스토리지	7
1.4. DCN 엣지	7
2장. DCN(DISTRIBUTED COMPUTE NODE) 배포 계획	8
2.1. DCN 아키텍처의 스토리지 고려 사항	8
2.2. DCN 아키텍처의 네트워킹 고려 사항	8
3장. 언더클라우드에서 라우팅된 스파인-리프트 구성	11
3.1. 스파인 리프트 프로비저닝 네트워크 구성	11
3.2. DHCP 릴레이 구성	12
3.3. 리프트 노드에 대한 역할 지정	15
3.4. 베어 메탈 노드 포트를 컨트롤 플레인 네트워크 세그먼트에 매핑	17
3.5. 스파인-리프트 프로비저닝 네트워크에 새 리프트 추가	18
4장. DCN 배포를 위한 오버클라우드 템플릿 준비	20
4.1. 별도의 HEAT 스택을 사용하기 위한 사전 요구 사항	20
4.2. 예제 별도의 HEAT 스택 배포 제한	20
4.3. 별도의 HEAT 스택 배포 설계	20
4.4. 별도의 HEAT 스택 관리	21
4.5. 컨테이너 이미지 검색	21
4.6. 엣지에 대한 빠른 DATAPATH 역할 생성	22
5장. 중앙 위치 설치	24
5.1. 엣지 스토리지 없이 중앙 컨트롤러 배포	24
5.2. 스토리지를 사용하여 중앙 사이트 배포	26
5.3. 외부 CEPH 통합	29
6장. 스토리지 없이 에지 배포	34
6.1. 스토리지가 없는 DCN 에지 사이트의 아키텍처	34
6.2. 스토리지 없이 에지 노드 배포	35
6.3. 에지에서 특정 이미지 유형 제외	37
7장. 엣지에 스토리지 배포	39
7.1. 스토리지를 사용한 엣지 배포의 역할	39
7.2. 스토리지가 있는 DCN 에지 사이트 아키텍처	40
7.3. 하이퍼컨버지드 스토리지를 사용하는 DCN 에지 사이트 아키텍처	41
7.4. 하이퍼컨버지드 스토리지를 사용하여 에지 사이트 배포	42
7.5. 엣지에 사전 설치된 RED HAT CEPH STORAGE 클러스터 사용	44
7.6. 중앙 위치 업데이트	47
7.7. DCN에 RED HAT CEPH STORAGE 대시보드 배포	49
8장. 엣지에서 네트워크 트래픽 로드 밸런싱	51
8.1. 서비스 가용성 영역 로드 밸런싱을 위한 네트워크 리소스 생성	51
8.2. 로드 밸런싱 서비스를 위한 가용성 영역 생성	53
8.3. 가용성 영역에서 로드 밸런서 생성	57
9장. DISTRIBUTEDCOMPUTEHCI 노드 교체	59

9.1. RED HAT CEPH STORAGE 서비스 제거	59
9.2. IMAGE 서비스(GLANCE) 서비스 제거	61
9.3. BLOCK STORAGE(CINDER) 서비스 제거	62
9.4. DISTRIBUTEDCOMPUTEHCI 노드 삭제	62
9.5. 삭제된 DISTRIBUTEDCOMPUTEHCI 노드 교체	63
9.6. 교체된 DISTRIBUTEDCOMPUTEHCI 노드의 기능 확인	64
9.7. DISTRIBUTEDCOMPUTEHCI 상태 문제 해결	66
10장. 키 관리자를 사용하여 배포	68
10.1. 키 관리자를 사용하여 엣지 사이트 배포	68
11장. NOVA에 GLANCE 이미지 사전 연결	69
11.1. TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE 플레이북 실행	70
11.2. 성능 고려 사항	71
11.3. DCN 사이트에 대한 이미지 배포 최적화	71
11.4. NOVA-CACHE 정리 구성	72
12장. DCN용 TLS-E	74
12.1. TLS-E를 사용하여 분산 컴퓨팅 노드 아키텍처 배포	74
13장. 외부 액세스를 위한 CEPH 키 생성	76
13.1. 외부 액세스를 위한 CEPH 키 생성	76
13.2. 외부 CEPH 키 사용	77
부록 A. 배포 마이그레이션 옵션	80
A.1. 엣지 스토리지 검증	80
A.2. 스파인 및 리프 배포로 마이그레이션	84
A.3. 다중 스택 배포로 마이그레이션	85
A.4. 엣지 사이트 간 백업 및 복원	85
A.5. DCN 환경 채택 및 준비	87

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 [CTO Chris Wright의 메시지](#)를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 더 나은지 알려주십시오.

Jira에서 문서 피드백 제공

문제 생성 양식을 사용하여 문서에 대한 피드백을 제공합니다. Jira 문제는 Red Hat OpenStack Platform Jira 프로젝트에서 생성되어 피드백의 진행 상황을 추적할 수 있습니다.

1. Jira에 로그인했는지 확인합니다. Jira 계정이 없는 경우 피드백을 제출할 계정을 생성합니다.
2. 다음 링크를 클릭하여 **문제 생성** 페이지를 엽니다.
<https://issues.redhat.com/secure/CreateInfoDetails!init.jspx?pid=12336920&summary=Documentation%20feedback:%20%3CAdd%20summary%20here%3E&i<Include+the+documentation+URL,+the%20chapter+or+section+number,+and+a+detailed+descrip>
3. **요약** 및 **설명** 필드를 작성합니다. **설명** 필드에 문서 URL, 장 또는 섹션 번호, 문제에 대한 자세한 설명을 포함합니다. 양식의 다른 필드를 수정하지 마십시오.
4. **생성**을 클릭합니다.

1장. DCN 이해

DCN(Distributed Compute Node) 아키텍처는 엣지 사용 사례로 원격 컴퓨팅 및 스토리지 노드를 공통의 중앙 집중식 컨트롤 플레인을 공유하면서 원격으로 배포할 수 있습니다. DCN 아키텍처를 사용하면 성능 향상을 위해 운영 요구에 보다 전략적으로 워크로드를 배치할 수 있습니다.

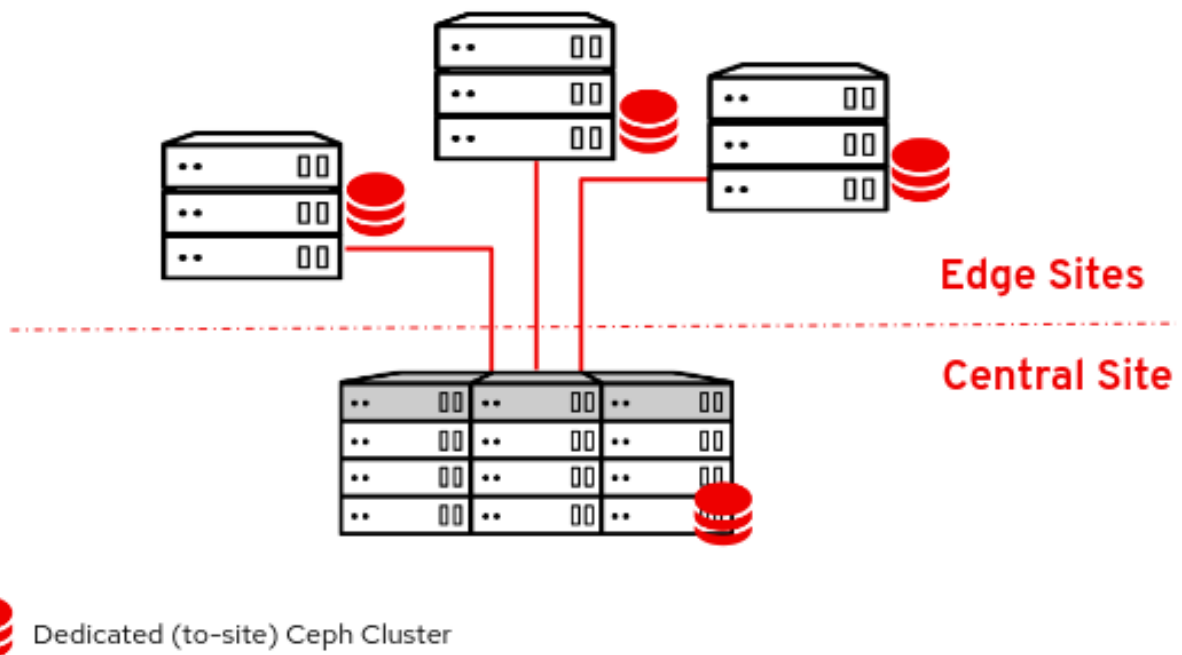
중앙 위치는 모든 역할로 구성될 수 있지만 최소한 세 개의 컨트롤러가 필요합니다. 컴퓨팅 노드는 엣지 및 중앙 위치에 있을 수 있습니다.

DCN 아키텍처는 hub 및 spoke routed network deployment입니다. DCN은 Red Hat OpenStack Platform director를 통해 라우팅된 프로비저닝 및 컨트롤 플레인 네트워킹을 위한 스파인 및 리프 배포와 비교할 수 있습니다.

- 허브는 핵심 라우터 및 데이터 센터 게이트웨이(DC-GW)가 있는 중앙 사이트입니다.
- spoke는 원격 엣지 또는 리프입니다.

엣지 위치에는 컨트롤러가 없으므로 Red Hat OpenStack Platform의 기존 배포와 아키텍처가 다릅니다.

- 컨트롤 플레인 서비스는 중앙 위치에서 원격으로 실행됩니다.
- Pacemaker가 설치되지 않았습니다.
- Block Storage 서비스(cinder)는 활성/활성 모드에서 실행됩니다.
- etcd는 DLM(Distributed lock manager)으로 배포됩니다.



1.1. 분산 컴퓨팅 노드 아키텍처의 필수 소프트웨어

다음 표는 Red Hat OpenStack Platform을 분산 컴퓨팅 노드(DCN) 아키텍처에 배포하는 데 필요한 소프트웨어 및 최소 버전을 보여줍니다.

플랫폼	버전	선택 사항
Red Hat Enterprise Linux	9.2	없음
Red Hat OpenStack Platform	17.1	없음
Red Hat Ceph Storage	5	제공됨

1.2. 멀티 스택 설계

DCN 설계와 함께 RHOSP(Red Hat OpenStack Platform)를 배포할 때 여러 스택 배포 및 관리에 Red Hat director의 기능을 사용하여 각 사이트를 별도의 스택으로 배포합니다.

배포가 Red Hat OpenStack Platform 13에서 업그레이드되지 않는 한 DCN 아키텍처를 단일 스택으로 관리할 수 없습니다. 기존 스택을 분할하는 데 지원되는 방법은 없지만 기존 배포에 스택을 추가할 수 있습니다. 자세한 내용은 [A.3절. "다중 스택 배포로 마이그레이션"](#)의 내용을 참조하십시오.

중앙 위치는 RHOSP의 기존 스택 배포이지만 중앙 스택을 사용하여 컴퓨팅 노드 또는 Red Hat Ceph 스토리지를 배포할 필요는 없습니다.

DCN을 사용하면 각 위치를 별도의 가용성 영역(AZ)으로 배포합니다.

1.3. DCN 스토리지

스토리지 없이 또는 하이퍼컨버지드 노드에서 Ceph를 사용하여 각 에지 사이트를 배포할 수 있습니다. 배포하는 스토리지는 배포하는 사이트 전용입니다.

DCN 아키텍처는 Glance 멀티 저장소를 사용합니다. 스토리지 없이 배포된 에지 사이트의 경우 추가 툴을 사용할 수 있으므로 Compute 서비스(nova) 캐시에 이미지를 캐시하고 저장할 수 있습니다. nova에서 Glance 이미지를 캐싱하면 WAN 링크에서 이미지를 다운로드하는 프로세스를 방지하여 인스턴스의 부팅 시간이 빨라집니다. 자세한 내용은 [11장. nova에 Glance 이미지 사전 연결](#)의 내용을 참조하십시오.

1.4. DCN 엣지

분산 컴퓨팅 노드 아키텍처를 사용하면 중앙 사이트에 제어 노드를 배포하고 이러한 컨트롤러를 사용하여 지리적으로 분산된 에지 사이트를 관리합니다. 엣지 사이트를 배포할 때 컴퓨팅 노드만 배포하여 엣지 사이트를 기존 Red Hat OpenStack Platform 배포와 크게 다릅니다. 에지 사이트에서 인스턴스를 시작하면 필요한 이미지가 로컬 Image 서비스(glance) 저장소에 자동으로 복사됩니다. 인스턴스 시작 중에 시간을 절약하기 위해 glance multistore를 사용하여 중앙 이미지 저장소의 이미지를 에지 사이트로 복사할 수 있습니다. 자세한 내용은 [여러 저장소가 있는 이미지 서비스를](#) 참조하십시오.

엣지 사이트에서:

- 컨트롤 플레인 서비스는 중앙 위치에서 원격으로 실행됩니다.
- Pacemaker는 DCN 사이트에서 실행되지 않습니다.
- Block Storage 서비스(cinder)는 활성/활성 모드에서 실행됩니다.
- etcd는 DLM(Distributed lock manager)으로 배포됩니다.

2장. DCN(DISTRIBUTED COMPUTE NODE) 배포 계획

DCN 아키텍처를 계획할 때 필요한 기술이 사용 가능하고 지원되는지 확인하십시오.

2.1. DCN 아키텍처의 스토리지 고려 사항

다음 기능은 현재 DCN 아키텍처에서 지원되지 않습니다.

- 에지 사이트 간에 볼륨 스냅샷을 복사합니다. 볼륨에서 이미지를 생성하고 glance를 사용하여 이미지를 복사하여 이 문제를 해결할 수 있습니다. 이미지가 복사되면 이미지에서 볼륨을 생성할 수 있습니다.
- 엣지의 Ceph Rados Gateway(RGW)입니다.
- 에지의 CephFS입니다.
- 에지 사이트의 HA(인스턴스 고가용성)입니다.
- 사이트 간 RBD 미러링.
- 에지 사이트 또는 중앙 위치에서 에지 사이트로 실시간 또는 콜드 인스턴스 마이그레이션. 사이트 경계 내에서 인스턴스를 마이그레이션할 수 있습니다. 사이트 간에 이미지를 이동하려면 이미지를 스냅샷을 작성하고 **glance image-import** 를 사용해야 합니다. 자세한 내용은 [사이트 간에 이미지 스냅샷을 생성하고 복사하는](#) 방법을 참조하십시오.

또한 다음을 고려해야 합니다.

- 이미지를 에지 사이트에 복사하기 전에 중앙 위치에 업로드해야 합니다. 각 이미지의 사본은 중앙 위치의 Image 서비스(glance)에 있어야 합니다.
- 이미지, 컴퓨팅 및 블록 스토리지 서비스에 RBD 스토리지 드라이버를 사용해야 합니다.
- 각 사이트에 대해 고유한 가용성 영역을 할당하고 NovaComputeAvailabilityZone 및 CinderStorageAvailabilityZone 매개변수에 동일한 값을 사용합니다.
- 에지 사이트에서 중앙 위치로 오프라인 볼륨을 마이그레이션하거나 그 반대의 경우도 마찬가지입니다. 에지 사이트 간에 직접 볼륨을 마이그레이션할 수 없습니다.

2.2. DCN 아키텍처의 네트워킹 고려 사항

다음 기능은 현재 DCN 아키텍처에서 지원되지 않습니다.

- DPDK 노드의 DHCP
- TC Flower Hardware Offload의 contrack

TC Flower Hardware Offload용 contrack은 DCN에서 기술 프리뷰로 사용할 수 있으므로 이러한 솔루션을 함께 사용하면 Red Hat에서 완전히 지원되지 않습니다. 이 기능은 테스트를 위해 DCN에서만 사용해야 하며 프로덕션 환경에 배포해서는 안 됩니다. 기술 프리뷰 기능에 대한 자세한 내용은 적용 범위 세부 정보를 참조하십시오.

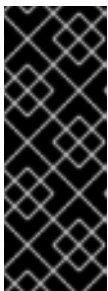
다음 ML2/OVS 기술이 지원됩니다.

- DPDK 노드에서 DHCP가 없는 OVS-DPDK
- SR-IOV

- TC는 conntrack 없이 하드웨어 오프로드를 발생시킵니다.
- 에지에 네트워크터 노드가 있는 Neutron 가용성 영역(AZ)과 사이트당 하나의 AZ
- 라우팅된 공급자 네트워크

다음 ML2/OVN 네트워킹 기술이 완전히 지원됩니다.

- DPDK 노드에서 DHCP가 없는 OVS-DPDK
- SR-IOV (DHCP 제외)
- TC는 conntrack 없이 하드웨어 오프로드를 발생시킵니다.
- 라우팅된 공급자 네트워크
- Neutron AZ가 지원되는 OVN GW(네트워크 노드)



중요

OVNCMOptions: 'enable-chassis-as-gw' 를 설정하고 **OVNAvailabilityZone** 매개변수에 하나 이상의 AZ 값을 제공하여 모든 라우터 게이트웨이 포트가 OpenStack 컨트롤러 노드에 있는지 확인합니다. 이러한 작업을 수행하면 라우터가 라우터 게이트웨이 포트의 잠재적인 호스트로 모든 새시를 예약하지 않습니다. 자세한 내용은 *Red Hat OpenStack Platform 네트워킹 구성에서 ML2/OVN을 사용하여 네트워크 서비스 가용성 영역* 구성을 참조하십시오.

또한 다음을 고려해야 합니다.

- 네트워크 대기 시간: RTT(Round-trip Time)로 측정된 대기 시간을 조정하여 허용 가능한 성능을 유지하기 위해 예상되는 동시 API 작업 수를 조정합니다. 최대 TCP/IP 처리량은 RTT에 비례하지 않습니다. 커널 TCP 매개변수를 조정하여 높은 대역폭으로 대기 시간이 긴 연결로 일부 문제를 완화할 수 있습니다. 사이트 간 통신이 100ms를 초과하면 Red Hat 지원에 문의하십시오.
- 네트워크 삭제: 에지 사이트에서 중앙 사이트에 일시적으로 연결이 끊어지면 중단 기간 동안 영향을 받는 에지 사이트에서 OpenStack 컨트롤 플레인 API 또는 CLI 작업을 실행할 수 없습니다. 예를 들어 에지 사이트의 컴퓨팅 노드는 결과적으로 인스턴스의 스냅샷을 생성하거나 인증 토큰을 발행하거나 이미지를 삭제할 수 없습니다. 일반 OpenStack 컨트롤 플레인 API 및 CLI 작업은 이 중단 중에 계속 작동하고 작동 중인 다른 에지 사이트를 계속 제공할 수 있습니다. 이미지 유형: Ceph 스토리지를 사용하여 DCN 아키텍처를 배포할 때 원시 이미지를 사용해야 합니다.
- 이미지 크기 조정:
 - 오버클라우드 노드 이미지 - Overcloud 노드 이미지가 중앙 언더클라우드 노드에서 다운로드됩니다. 이러한 이미지는 프로비저닝 중에 중앙 사이트에서 에지 사이트로 필요한 모든 네트워크를 통해 전송되는 대용량 파일입니다.
 - 인스턴스 이미지: 에지에 블록 스토리지가 없는 경우 이미지 서비스 이미지는 처음 사용하는 동안 WAN을 통과합니다. 이미지는 나중에 사용할 수 있도록 대상 에지 노드에 로컬로 복사되거나 캐시됩니다. Glance 이미지에는 크기 제한이 없습니다. 전송 시간은 사용 가능한 대역폭 및 네트워크 대기 시간에 따라 다릅니다. 에지에 블록 스토리지가 있는 경우 이미지가 비동기식으로 복사되어 에지에서 더 빠르게 부팅됩니다.
- 공급자 네트워크: DCN 배포에 권장되는 네트워킹 접근 방식입니다. 원격 사이트에서 공급자 네트워크를 사용하는 경우 Networking 서비스(neutron)가 사용 가능한 네트워크를 연결할 수 있는 위

치에 제한이나 검사를 수행하지 않는 것을 고려해야 합니다. 예를 들어 에지 사이트 A에서만 공급자 네트워크를 사용하는 경우 에지 사이트 B의 공급자 네트워크에 연결하지 않도록 해야 합니다. 이는 이를 컴퓨팅 노드에 바인딩할 때 공급자 네트워크에 검증 검사가 없기 때문입니다.

- **사이트별 네트워크:** 특정 사이트에 특정 네트워크를 사용하는 경우 DCN 네트워킹의 제한이 발생합니다. 컴퓨팅 노드와 함께 중앙 집중식 중성자를 배포할 때 특정 컴퓨팅 노드를 원격 노드로 식별하는 트리거가 없습니다. 결과적으로 컴퓨팅 노드는 다른 컴퓨팅 노드 목록을 수신하고 서로 자동으로 터널을 형성합니다. 터널은 중앙 사이트를 통해 에지에서 에지로 구성됩니다. VXLAN 또는 Geneve를 사용하는 경우 모든 사이트의 모든 컴퓨팅 노드는 로컬 또는 원격인지 여부에 관계없이 다른 모든 컴퓨팅 노드 및 컨트롤러 노드와 터널을 형성합니다. 어디에서나 동일한 중성자 네트워크를 사용하는 경우에는 문제가 되지 않습니다. VLAN을 사용하는 경우 neutron은 모든 컴퓨팅 노드에 동일한 브릿지 매핑이 있고 모든 VLAN을 모든 사이트에서 사용할 수 있을 것으로 예상합니다.
- **추가 사이트:** 중앙 사이트에서 추가 원격 사이트로 확장해야 하는 경우 Red Hat OpenStack Platform director의 openstack CLI를 사용하여 새 네트워크 세그먼트와 서브넷을 추가할 수 있습니다.
- **에지 서버가 사전 프로비저닝되지 않은 경우** 라우팅된 세그먼트에서 인트로스펙션 및 프로비저닝을 위해 DHCP 릴레이를 구성해야 합니다.
- **라우팅은 클라우드 또는 각 에지 사이트를 허브에 연결하는 네트워킹 인프라에 구성해야 합니다.** 각 Red Hat OpenStack Platform 클러스터 네트워크(외부, 내부 API 등)에 L3 서브넷을 할당하는 네트워킹 설계를 구현해야 합니다.

3장. 언더클라우드에서 라우팅된 스파인-리프트 구성

이 섹션에서는 구성 가능 네트워크를 사용하여 라우팅된 스파인-리프트를 수용하도록 언더클라우드를 구성하는 방법에 대해 설명합니다.

3.1. 스파인 리프트 프로비저닝 네트워크 구성

스파인 리프트 인프라에 대한 프로비저닝 네트워크를 구성하려면 **undercloud.conf** 파일을 편집하고 다음 절차에 포함된 관련 매개변수를 설정합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. **undercloud.conf** 파일이 없는 경우 샘플 템플릿 파일을 복사합니다.

```
[stack@director ~]$ cp /usr/share/python-tripleoclient/undercloud.conf.sample
~/undercloud.conf
```

3. **undercloud.conf** 파일을 편집합니다.
4. **[DEFAULT]** 섹션에 다음 값을 설정합니다.

- a. **local_ip** 를 **leaf0** 의 언더클라우드 IP로 설정합니다.

```
local_ip = 192.168.10.1/24
```

- b. **undercloud_public_host** 를 언더클라우드의 외부 연결 IP 주소로 설정합니다.

```
undercloud_public_host = 10.1.1.1
```

- c. **undercloud_admin_host** 를 언더클라우드의 관리 IP 주소로 설정합니다. 이 IP 주소는 일반적으로 leaf0에 있습니다.

```
undercloud_admin_host = 192.168.10.2
```

- d. **local_interface** 를 로컬 네트워크의 bridge로 설정합니다.

```
local_interface = eth1
```

- e. **enable_routed_networks** 를 **true** 로 설정합니다.

```
enable_routed_networks = true
```

- f. **subnets** 매개변수를 사용하여 서브넷 목록을 정의합니다. 라우트된 스파인 및 리프트의 각 L2 세그먼트에 대해 하나의 서브넷을 정의합니다.

```
subnets = leaf0,leaf1,leaf2
```

- g. **local_subnet** 매개변수를 사용하여 로컬에 있는 물리적 L2 세그먼트와 연결된 서브넷을 지정합니다.

```
local_subnet = leaf0
```

- h. **undercloud_nameservers** 값을 설정합니다.

```
undercloud_nameservers = 10.11.5.19,10.11.5.20
```

작은 정보

/etc/resolv.conf에서 확인하여 언더클라우드 네임서버에 사용되는 DNS 서버의 현재 IP 주소를 찾을 수 있습니다.

5. **subnets** 매개변수에 정의된 각 서브넷에 대한 새 섹션을 만듭니다.

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
gateway = 192.168.11.1
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False
```

6. **undercloud.conf** 파일을 저장합니다.

7. 언더클라우드 설치 명령을 실행합니다.

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

이 구성은 provisioning 네트워크 또는 컨트롤 플레인에 세 개의 서브넷을 생성합니다. 오버클라우드는 각 네트워크를 사용하여 각 리프 내에 시스템을 프로비저닝합니다.

언더클라우드에 대한 DHCP 요청을 올바르게 릴레이하려면 DHCP 릴레이를 구성해야 할 수 있습니다.

3.2. DHCP 릴레이 구성

요청을 전달하려는 원격 네트워크 세그먼트에 연결된 스위치, 라우터 또는 서버에서 DHCP 릴레이 서비스를 실행합니다.



참고

언더클라우드에서 DHCP 릴레이 서비스를 실행하지 마십시오.

언더클라우드는 provisioning 네트워크에서 두 개의 DHCP 서버를 사용합니다.

- 인트로스펙션 DHCP 서버입니다.
- 프로비저닝 DHCP 서버.

DHCP 요청을 언더클라우드의 두 DHCP 서버로 전달하도록 DHCP 릴레이를 구성해야 합니다.

이를 지원하는 장치와 함께 UDP 브로드캐스트를 사용하여 언더클라우드 프로비저닝 네트워크가 연결된 L2 네트워크 세그먼트에 DHCP 요청을 릴레이할 수 있습니다. 또는 DHCP 요청을 특정 IP 주소로 릴레이하는 UDP 유니캐스트를 사용할 수 있습니다.



참고

특정 장치 유형의 DHCP 릴레이 구성은 이 문서의 범위를 벗어납니다. 이 문서에서는 ISC DHCP 소프트웨어의 구현을 사용하여 DHCP 릴레이 구성 예제를 제공합니다. 자세한 내용은 수동 페이지 [dhcrelay\(8\)](#)를 참조하십시오.



중요

DHCP 옵션 79는 일부 릴레이, 특히 DHCPv6 주소를 제공하는 릴레이 및 원래 MAC 주소에서 전달하지 않는 릴레이에 필요합니다. 자세한 내용은 [RFC6939](#)를 참조하십시오.

브로드캐스트 DHCP 릴레이

이 방법은 DHCP 서버 또는 서버가 있는 L2 네트워크 세그먼트에 UDP 브로드캐스트 트래픽을 사용하여 DHCP 요청을 릴레이합니다. 네트워크 세그먼트의 모든 장치는 브로드캐스트 트래픽을 수신합니다. UDP 브로드캐스트를 사용하면 언더클라우드의 두 DHCP 서버가 릴레이된 DHCP 요청을 수신합니다. 구현에 따라 인터페이스 또는 IP 네트워크 주소를 지정하여 이를 구성할 수 있습니다.

인터페이스

DHCP 요청이 릴레이되는 L2 네트워크 세그먼트에 연결된 인터페이스를 지정합니다.

IP 네트워크 주소

DHCP 요청이 릴레이되는 IP 네트워크의 네트워크 주소를 지정합니다.

유니캐스트 DHCP 릴레이

이 방법은 UDP 유니캐스트 트래픽을 사용하여 특정 DHCP 서버로 DHCP 요청을 릴레이합니다. UDP 유니캐스트를 사용하는 경우 언더클라우드의 인트로스펙션에 사용되는 인터페이스에 할당된 IP 주소와 OpenStack Networking(neutron) 서비스에서 생성한 네트워크 네임스페이스의 IP 주소와 **ctlplane** 네트워크의 DHCP 서비스를 호스팅하기 위해 DHCP 릴레이를 제공하는 장치를 구성해야 합니다.

인트로스펙션에 사용되는 인터페이스는 **undercloud.conf** 파일에서 **inspection_interface**로 정의된 인터페이스입니다. 이 매개변수를 설정하지 않은 경우 언더클라우드의 기본 인터페이스는 **br-ctlplane**입니다.



참고

일반적으로 인트로스펙션에 **br-ctlplane** 인터페이스를 사용하는 것이 일반적입니다. **undercloud.conf** 파일에서 **local_ip** 로 정의된 IP 주소는 **br-ctlplane** 인터페이스에 있습니다.

Neutron DHCP 네임스페이스에 할당된 IP 주소는 **undercloud.conf** 파일의 **local_subnet** 에 대해 구성하는 IP 범위에서 사용 가능한 첫 번째 주소입니다. IP 범위의 첫 번째 주소는 구성에서 **dhcp_start** 로 정의된 주소입니다. 예를 들어 다음 구성을 사용하는 경우 **192.168.10.10** 은 IP 주소입니다.

```
[DEFAULT]
local_subnet = leaf0
subnets = leaf0,leaf1,leaf2

[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False
```



주의

DHCP 네임스페이스의 IP 주소가 자동으로 할당됩니다. 대부분의 경우 이 주소는 IP 범위의 첫 번째 주소입니다. 이 경우 언더클라우드에서 다음 명령을 실행합니다.

```
$ openstack port list --device-owner network:dhcp -c "Fixed IP Addresses"
+-----+
| Fixed IP Addresses |
+-----+
| ip_address='192.168.10.10', subnet_id='7526fbe3-f52a-4b39-a828-ec59f4ed12b2' |
+-----+
$ openstack subnet show 7526fbe3-f52a-4b39-a828-ec59f4ed12b2 -c name
+-----+
| Field | Value |
+-----+
| name | leaf0 |
+-----+
```

dhcrelay 구성의 예

다음 예제에서 **dhcp** 패키지의 **dhcrelay** 명령은 다음 구성을 사용합니다.

- 들어오는 DHCP 요청을 릴레이하는 인터페이스: **eth1**, **eth2** 및 **eth3**.
- 네트워크 세그먼트의 언더클라우드 DHCP 서버가 **eth0** 에 연결되어 있습니다.
- 인트로스펙션에 사용되는 DHCP 서버는 IP 주소 **192.168.10.1** 에서 수신 대기 중입니다.

- 프로비저닝에 사용되는 DHCP 서버는 IP 주소 **192.168.10.10** 에서 수신 대기 중입니다.

그러면 다음과 같은 **dhcrelay** 명령이 생성됩니다.

- **dhcrelay** 버전 4.2.x:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
-i eth0 -i eth1 -i eth2 -i eth3
```

- **dhcrelay** 버전 4.3.x 이상:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
-iu eth0 -id eth1 -id eth2 -id eth3
```

Cisco IOS 라우팅 스위치 구성의 예

이 예에서는 다음 Cisco IOS 구성을 사용하여 다음 작업을 수행합니다.

- provisioning 네트워크에 사용할 VLAN을 구성합니다.
- 리프의 IP 주소를 추가합니다.
- UDP 및 BOOTP 요청을 IP 주소 **192.168.10.1** 에서 수신 대기하는 인트로스펙션 DHCP 서버로 전달합니다.
- UDP 및 BOOTP 요청을 IP 주소 **192.168.10.10** 에서 수신 대기하는 provisioning DHCP 서버로 전달합니다.

```
interface vlan 2
ip address 192.168.24.254 255.255.255.0
ip helper-address 192.168.10.1
ip helper-address 192.168.10.10
!
```

provisioning 네트워크를 구성했으므로 나머지 오버클라우드 리프 네트워크를 구성할 수 있습니다.

3.3. 리프 노드에 대한 역할 지정

각 리프 네트워크의 각 역할에는 해당 리프에 노드를 태그할 수 있도록 플레이어 및 역할 할당이 필요합니다. 각 플레이어를 만들고 역할에 할당하려면 다음 단계를 완료합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
[stack@director ~]$ source ~/stackrc
```

2. 노드 목록을 검색하여 UUID를 확인합니다.

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

3. 리프 네트워크 및 역할을 식별하는 사용자 정의 리소스 클래스를 사용하여 역할에 지정할 각 베어 메탈 노드를 할당합니다.

```
openstack baremetal node set \
--resource-class baremetal.<ROLE> <node>
```

- <ROLE>을 역할을 식별하는 이름으로 바꿉니다.
- <node>를 베어 메탈 노드의 ID로 바꿉니다.
예를 들어 다음 명령을 입력하여 UUID 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13을 Leaf2의 Compute 역할에 태그를 지정합니다.

```
(undercloud)$ openstack baremetal node set \
--resource-class baremetal.COMPUTE-LEAF2 58c3d07e-24f2-48a7-bbb6-
6843f0e8ee13
```

4. 아직 정의되지 않은 경우 **overcloud-baremetal-deploy.yaml**에 각 역할을 추가합니다.
5. 역할의 노드에 할당할 리소스 클래스를 정의합니다.

```
- name: <role>
  count: 1
  defaults:
    resource_class: baremetal.<ROLE>
```

- <role>을 역할 이름으로 바꿉니다.
 - <ROLE>을 역할을 식별하는 이름으로 바꿉니다.
6. baremetal-deploy.yaml 파일에서 역할의 노드에 할당할 리소스 클래스를 정의합니다. 배포하려는 역할, 프로필, 수량 및 관련 네트워크를 지정합니다.

```
- name: <role>
  count: 1
  hostname_format: <role>-%index%
  ansible_playbooks:
    - playbook: bm-deploy-playbook.yaml
  defaults:
    resource_class: baremetal.<ROLE>
    profile: control
    networks:
      - network: external
        subnet: external_subnet
      - network: internal_api
        subnet: internal_api_subnet01
      - network: storage
        subnet: storage_subnet01
      - network: storage_mgmt
        subnet: storage_mgmt_subnet01
      - network: tenant
        subnet: tenant_subnet01
    network_config:
      template: templates/multiple_nics/multiple_nics_dvr.j2
      default_route_network:
        - external
```

- <role>을 역할 이름으로 바꿉니다.

- <ROLE>을 역할을 식별하는 이름으로 바꿉니다.



참고

배포하는 모든 스택에 대해 **baremetal-deploy.yaml** 환경 파일을 **/home/stack/<stack>** 에서 생성해야 합니다.

3.4. 베어 메탈 노드 포트를 컨트롤 플레인 네트워크 세그먼트에 매핑

L3 라우팅 네트워크에서 배포를 활성화하려면 베어 메탈 포트에서 **physical_network** 필드를 구성해야 합니다. 각 베어 메탈 포트는 OpenStack Bare Metal(ironic) 서비스의 베어 메탈 노드와 연결됩니다. 물리적 네트워크 이름은 언더클라우드 구성의 **subnets** 옵션에 포함된 이름입니다.



참고

undercloud.conf 파일에서 **local_subnet** 으로 지정된 서브넷의 물리적 네트워크 이름은 항상 **ctlplane** 입니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 베어 메탈 노드를 확인합니다.

```
$ openstack baremetal node list
```

3. 베어 메탈 노드가 등록 또는 관리 가능 상태인지 확인합니다. 베어 메탈 노드가 이러한 상태 중 하나에 없는 경우 baremetal 포트에서 **physical_network** 속성을 설정하는 명령이 실패합니다. 모든 노드를 **manageable** 상태로 설정하려면 다음 명령을 실행합니다.

```
$ for node in $(openstack baremetal node list -f value -c Name); do openstack baremetal node manage $node --wait; done
```

4. 어떤 baremetal 포트가 어떤 baremetal 노드와 연결되어 있는지 확인합니다.

```
$ openstack baremetal port list --node <node-uuid>
```

5. 포트의 **physical-network** 매개변수를 설정합니다. 아래 예제에서 세 개의 서브넷은 **leaf0,leaf1,leaf2** 에 정의되어 있습니다. **local_subnet**은 **leaf0** 입니다. **local_subnet**의 물리적 네트워크는 항상 **ctlplane** 이므로 **leaf0** 에 연결된 baremetal 포트는 **ctlplane**을 사용합니다. 나머지 포트는 다른 리프트 이름을 사용합니다.

```
$ openstack baremetal port set --physical-network ctlplane <port-uuid>
$ openstack baremetal port set --physical-network leaf1 <port-uuid>
$ openstack baremetal port set --physical-network leaf2 <port-uuid>
```

6. 오버클라우드를 배포하기 전에 노드를 검사합니다. 배포를 위해 노드를 사용 가능한 대로 설정하려면 **--all-manageable** 및 **--provide** 옵션을 포함합니다.

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

3.5. 스파인-리프 프로비저닝 네트워크에 새 리프 추가

새 물리적 사이트 추가 포함 가능한 네트워크 용량을 늘릴 때 Red Hat OpenStack Platform 스파인-리프 프로비저닝 네트워크에 새 리프와 해당 서브넷을 추가해야 할 수 있습니다. 오버클라우드에서 리프를 프로비저닝하면 해당 언더클라우드 리프가 사용됩니다.

사전 요구 사항

- RHOSP 배포에서는 스파인-리프트 네트워크 토폴로지를 사용합니다.

절차

1. 언더클라우드 호스트에 `stack` 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **/home/stack/undercloud.conf** 파일에서 다음을 수행합니다.

- a. **subnets** 매개변수를 찾고 추가할 리프의 새 서브넷을 추가합니다.
서브넷은 라우팅된 스파인 및 리프의 L2 세그먼트를 나타냅니다.

예제

이 예에서는 새 리프(**leaf3**)에 대해 새 서브넷(**leaf3**)이 추가됩니다.

```
subnets = leaf0,leaf1,leaf2,leaf3
```

- b. 추가한 서브넷에 대한 섹션을 생성합니다.

예제

이 예제에서는 새 서브넷(**leaf3**)에 대해 **[leaf3]** 섹션이 추가되었습니다.

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
gateway = 192.168.11.1
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
```

```
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False

[leaf3]
cidr = 192.168.13.0/24
dhcp_start = 192.168.13.10
dhcp_end = 192.168.13.90
inspection_iprange = 192.168.13.100,192.168.13.190
gateway = 192.168.13.1
masquerade = False
```

4. **undercloud.conf** 파일을 저장합니다.
5. 언더클라우드를 다시 설치합니다.

```
$ openstack undercloud install
```

추가 리소스

- [스파인-리프트 배포에 새 리프트 추가](#)

4장. DCN 배포를 위한 오버클라우드 템플릿 준비

4.1. 별도의 HEAT 스택을 사용하기 위한 사전 요구 사항

별도의 heat 스택을 사용하여 배포를 생성하기 전에 환경이 다음 사전 요구 사항을 충족해야 합니다.

- Red Hat OpenStack Platform director 17.1의 설치된 인스턴스입니다.
- Ceph Storage 사용자의 경우: Red Hat Ceph Storage 5에 액세스합니다.
- 중앙 위치의 경우: 중앙 컨트롤러 노드로 사용할 수 있는 3개의 노드 세 개의 컨트롤러 노드 모두 동일한 heat 스택에 있어야 합니다. 별도의 heat 스택에서 컨트롤러 노드 또는 컨트롤 플레인 서비스를 분할할 수 없습니다.
- 에지에서 Ceph 스토리지를 배포하려는 경우 Ceph 스토리지는 중앙 위치에서 필요합니다.
- 추가 DCN 사이트마다 3개의 HCI 컴퓨팅 노드입니다.
- 모든 노드는 사전 프로비저닝되거나 중앙 배포 네트워크에서 PXE 부팅이 가능해야 합니다. DHCP 릴레이를 사용하여 DCN에 대한 이 연결을 활성화할 수 있습니다.
- 모든 노드가 ironic에 의해 검사되었습니다.
- Red Hat은 <role>HostnameFormat 매개변수를 기본값인 %stackname%-<role>-%index%로 두는 것이 좋습니다. %stackname% 접두사를 포함하지 않으면 오버클라우드에서 다른 스택의 분산 컴퓨팅 노드에 동일한 호스트 이름을 사용합니다. 분산 컴퓨팅 노드에서 %stackname% 접두사를 사용하여 노드를 다른 에지 사이트와 구별하는지 확인합니다. 예를 들어 **dcn0** 및 **dcn1**이라는 두 개의 에지 사이트를 배포하는 경우 스택 이름 접두사는 언더클라우드에서 **openstack server list** 명령을 실행할 때 dcn0-distributedcompute-0과 dcn1-distributedcompute-0을 구분하는 데 도움이 됩니다.
- **centralrc** 인증 파일을 가져와 에지 사이트뿐만 아니라 중앙 위치에서 워크로드를 예약합니다. 에지 사이트에 대해 자동으로 생성되는 인증 파일이 필요하지 않습니다.

4.2. 예제 별도의 HEAT 스택 배포 제한

이 문서에서는 Red Hat OpenStack Platform에서 별도의 heat 스택을 사용하는 배포 예제를 제공합니다. 이 예제 환경에는 다음과 같은 제한 사항이 있습니다.

- 스핀인/Leaf 네트워킹 - 이 가이드의 예제에서는 DCN(Distributed compute node) 배포에 필요한 라우팅 요구 사항을 설명하지 않습니다.
- Ironic DHCP Relay - 이 안내서에는 DHCP 릴레이를 사용하여 Ironic을 구성하는 방법이 포함되어 있지 않습니다.

4.3. 별도의 HEAT 스택 배포 설계

별도의 heat 스택 내에서 배포를 분할하려면 먼저 컨트롤 플레인을 사용하여 단일 오버클라우드를 배포해야 합니다. 그런 다음 분산 컴퓨팅 노드(DCN) 사이트에 대해 별도의 스택을 생성할 수 있습니다. 다음 예제에서는 다양한 노드 유형에 대한 별도의 스택을 보여줍니다.

- 컨트롤러 노드: **central**이라는 별도의 heat 스택(예: 컨트롤러 배포)입니다. DCN 사이트의 새 heat 스택을 생성할 때 **중앙** 스택의 데이터로 생성해야 합니다. 컨트롤러 노드는 모든 인스턴스 관리 작업에서 사용할 수 있어야 합니다.

- DCN 사이트: **dcn0, dcn1** 등과 같은 고유하게 이름이 지정된 heat 스택을 별도로 보유할 수 있습니다. DHCP 릴레이를 사용하여 프로비저닝 네트워크를 원격 사이트로 확장합니다.



참고

각 스택에 대해 별도의 가용성 영역(AZ)을 생성해야 합니다.

4.4. 별도의 HEAT 스택 관리

이 가이드의 절차에서는 세 개의 heat 스택에 대해 환경 파일을 구성하는 방법을 보여줍니다(중앙, **dcn0** 및 **dcn1**). 각 heat 스택의 템플릿을 별도의 디렉터리에 저장하여 각 배포에 대한 정보를 격리하는 것이 좋습니다.

절차

1. 중앙 heat 스택을 정의합니다.

```
$ mkdir central
$ touch central/overrides.yaml
```

2. 중앙 heat 스택에서 모든 DCN 사이트의 공통 디렉터리로 데이터를 추출합니다.

```
$ mkdir dcn-common
$ touch dcn-common/overrides.yaml
```

3. **dcn0** 사이트를 정의합니다.

```
$ mkdir dcn0
$ touch dcn0/overrides.yaml
```

더 많은 DCN 사이트를 배포하려면 번호별로 추가 **dcn** 디렉토리를 생성하십시오.



참고

touch는 파일 조직의 예제를 제공하는 데 사용됩니다. 각 파일에는 성공적인 배포를 위한 적절한 콘텐츠가 포함되어야 합니다.

4.5. 컨테이너 이미지 검색

별도의 heat 스택이 있는 배포에 필요한 컨테이너 이미지를 검색하려면 다음 절차와 해당 예제 파일 콘텐츠를 사용합니다. 에지 사이트의 환경 파일을 사용하여 **openstack 컨테이너 image prepare** 명령을 실행하여 선택적 또는 에지별 서비스의 컨테이너 이미지가 포함되어 있는지 확인해야 합니다.

자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 컨테이너 이미지 준비*를 참조하십시오.

절차

1. **containers.yaml**에 레지스트리 서비스 계정 인증 정보를 추가합니다.

```
parameter_defaults:
  ContainerImagePrepare:
```

```
- push_destination: true
  set:
    ceph_namespace: registry.redhat.io/rhceph
    ceph_image: rhceph-6-rhel9
    ceph_tag: latest
    name_prefix: openstack-
    namespace: registry.redhat.io/rhosp17-rhel9
    tag: latest
ContainerImageRegistryCredentials:
  # https://access.redhat.com/RegistryAuthentication
  registry.redhat.io:
    registry-service-account-username: registry-service-account-password
```

2. **images-env.yaml**:으로 환경 파일을 생성합니다.

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file images-env.yaml
```

결과 **images-env.yaml** 파일은 생성된 스택에 대한 오버클라우드 배포 절차의 일부로 포함됩니다.

4.6. 엣지에 대한 빠른 DATAPATH 역할 생성

여기에서 빠른 datapath 서비스를 사용하려면 빠른 datapath 및 edge 서비스를 모두 정의하는 사용자 지정 역할을 생성해야 합니다. 배포에 사용할 역할 파일을 생성할 때 분산 컴퓨팅 노드 아키텍처 및 DPDK 또는 SR-IOV와 같은 빠른 데이터 경로 서비스 둘 다에 필요한 서비스를 정의하는 새로 생성된 역할을 포함할 수 있습니다.

예를 들어 DPDK를 사용하여 distributedCompute에 대한 사용자 정의 역할을 생성합니다.

사전 요구 사항

성공적인 언더클라우드 설치 자세한 내용은 [언더클라우드 설치](#)를 참조하십시오.

절차

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. 기본 **역할** 디렉토리를 복사합니다.

```
cp -r /usr/share/openstack-tripleo-heat-templates/roles ~/.
```

3. **DistributedCompute.yaml** 파일에서 **DistributedComputeDpdk.yaml** 이라는 새 파일을 생성합니다.

```
cp roles/DistributedCompute.yaml roles/DistributedComputeDpdk.yaml
```

4. 새 **DistributedComputeDpdk.yaml** 파일에 DPDK 서비스를 추가합니다.
DistributedComputeDpdk.yaml 파일에 없는 **ComputeOvsDpdk.yaml** 파일에서 매개변수를 지정하여 추가해야 하는 매개변수를 확인할 수 있습니다.

```
diff -u roles/DistributedComputeDpdk.yaml roles/ComputeOvsDpdk.yaml
```

출력에서 + 앞에 있는 매개변수는 ComputeOvsDpdk.yaml 파일에 있지만 DistributedComputeDpdk.yaml 파일에는 표시되지 않습니다. 이러한 매개변수를 새 **DistributedComputeDpdk.yaml** 파일에 포함합니다.

5. **DistributedComputeDpdk.yaml** 을 사용하여 **DistributedComputeDpdk** 역할 파일을 생성합니다.

```
openstack overcloud roles generate --roles-path ~/roles/ -o ~/roles/roles-custom.yaml
DistributedComputeDpdk
```

이 동일한 방법을 사용하여 SR-IOV에 대한 빠른 데이터 경로 역할을 생성하거나 요구 사항을 충족하기 위해 에지에 SR-IOV 및 DPDK 조합을 생성할 수 있습니다.

블록 스토리지 없이 에지 사이트를 배포하려는 경우 다음을 참조하십시오.

- [5장. 중앙 위치 설치](#)
- [6.2절. "스토리지 없이 에지 노드 배포"](#)

Red Hat Ceph Storage를 사용하여 에지 사이트를 배포하려면 다음을 참조하십시오.

- [5장. 중앙 위치 설치](#)
- [7.4절. "하이퍼컨버지드 스토리지를 사용하여 에지 사이트 배포"](#)

5장. 중앙 위치 설치

DCN(Distributed Compute Node) 아키텍처를 사용하여 Red Hat OpenStack 플랫폼을 배포하는 경우 스토리지 전략을 미리 결정해야 합니다. Red Hat Ceph Storage 없이 Red Hat OpenStack Platform을 중앙 위치에 배포하는 경우 Red Hat Ceph 스토리지를 사용하여 에지 사이트를 배포할 수 없습니다. 또한 나중에 재배포하여 중앙 위치에 Red Hat Ceph Storage를 추가할 수 없습니다.

DCN(Distributed Compute Node) 아키텍처의 중앙 위치를 배포할 때 클러스터를 배포할 수 있습니다.

- 컴퓨팅 노드 사용 중 또는 없음
- Red Hat Ceph Storage 사용 중

5.1. 엣지 스토리지 없이 중앙 컨트롤러 배포

중앙 위치에서 Image 서비스(glance)를 위한 백엔드로 Object Storage 서비스(swift)를 사용하는 경우 에지 사이트에서 블록 스토리지 없이 분산 컴퓨팅 노드 클러스터를 배포할 수 있습니다. 블록 스토리지 없이 배포된 사이트는 나중에 각 아키텍처의 다양한 역할 및 네트워킹 프로필로 인해 블록 스토리지를 갖도록 업데이트할 수 없습니다.

중요: 다음 절차에서는 lvm을 프로덕션에서 지원되지 않는 Cinder의 백엔드로 사용합니다. 인증된 블록 스토리지 솔루션을 Cinder용 백엔드로 배포해야 합니다.

일반적인 오버클라우드 배포와 유사한 방식으로 중앙 컨트롤러 클러스터를 배포합니다. 이 클러스터에는 컴퓨팅 노드가 필요하지 않으므로 컴퓨팅 수를 **0**으로 설정하여 기본값 **1**을 덮어쓸 수 있습니다. 중앙 컨트롤러에는 특정 스토리지 및 Oslo 구성 요구 사항이 있습니다. 이러한 요구 사항을 해결하려면 다음 절차를 사용하십시오.

사전 요구 사항

- 환경과 관련된 **network_data.yaml** 및 **vip_data.yaml** 파일을 생성해야 합니다. **/usr/share/openstack-tripleo-heat-templates/network-data-samples**에서 샘플 파일을 찾을 수 있습니다.
- 환경과 관련된 **overcloud-baremetal-deploy.yaml** 파일을 생성해야 합니다. 자세한 내용은 [오버클라우드의 베어 메탈 노드 프로비저닝](#)에서 참조하십시오.

프로세스

다음 절차에서는 중앙 위치의 초기 배포 단계를 간략하게 설명합니다.



참고

다음 단계에서는 glance 다중 저장소 없이 예제 DCN 배포와 관련된 배포 명령 및 환경 파일을 자세히 설명합니다. 이러한 단계에는 네트워킹과 같은 구성 측면이 관련이 없지만 필수 구성 요소는 포함되지 않습니다.

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

```
[stack@director ~]$ source /home/stack/stackrc
```

3. 환경 파일을 생성합니다.

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file /home/stack/central/central-images-env.yaml
```

4. 홈 디렉터리에서 배포하려는 각 스택에 대한 디렉터리를 생성합니다. 중앙 위치의 **network_data.yaml**, **vip_data.yaml** 및 **overcloud-baremetal-deploy.yaml** 템플릿을 **/home/stack/central/**로 이동합니다.

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

5. 오버클라우드의 네트워크를 프로비저닝합니다. 이 명령은 오버클라우드 네트워크의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. 오버클라우드의 가상 IP를 프로비저닝합니다. 이 명령은 가상 IP의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
(undercloud)$ openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

7. 베어 메탈 인스턴스를 프로비저닝합니다. 이 명령은 베어 메탈 노드의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
(undercloud)$ openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

8. 다음과 유사한 설정을 사용하여 **central/overrides.yaml** 이라는 파일을 생성합니다.

```
parameter_defaults:
  NtpServer:
    - 0.pool.ntp.org
    - 1.pool.ntp.org
  GlanceBackend: swift
```

- **ControllerCount: 3** 은 세 개의 노드가 배포되도록 지정합니다. Glance에는 swift를 사용하고 cinder의 경우 lvm을 사용하고 에지 컴퓨팅 노드에 대한 control-plane 서비스를 호스팅합니다.
- **ComputeCount: 0** 은 중앙 컨트롤러 노드와 함께 컴퓨팅 노드가 배포되지 않도록 하는 선택적 매개변수입니다.

- **GlanceBackend: swift** 는 Object Storage(swift)를 Image 서비스(glance) 백엔드로 사용합니다.
결과 구성은 다음과 같은 방식으로 DCN(Distributed Compute nodes)과 상호 작용합니다.
- DCN의 이미지 서비스는 중앙 Object Storage 백엔드에서 수신하는 이미지의 캐시된 사본을 생성합니다. 이미지 서비스는 HTTP를 사용하여 오브젝트 스토리지의 이미지를 로컬 디스크 캐시로 복사합니다.



참고

중앙 컨트롤러 노드는 분산 계산 노드(DCN) 사이트에 연결할 수 있어야 합니다. 중앙 컨트롤러 노드는 라우팅된 계층 3 연결을 사용할 수 있습니다.

9. **site-name.yaml** 환경 파일에서 사이트에 대한 이름 지정 규칙을 구성합니다. Nova 가용성 영역인 Cinder 스토리지 가용 영역은 다음과 일치해야 합니다.

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
  NovaComputeAvailabilityZone: central
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
  NovaCrossAZAttach: false
EOF
```

10. 중앙 컨트롤러 노드를 배포합니다. 예를 들어 다음 내용과 함께 **deploy.sh** 파일을 사용할 수 있습니다.

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-n /home/stack/central/network_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml
```



참고

openstack overcloud deploy 명령에 네트워킹 구성에 대한 heat 템플릿을 포함해야 합니다. 엣지 아키텍처를 위해 설계하려면 회전 및 리프트 네트워킹이 필요합니다. 자세한 내용은 [스파인-리프트 네트워킹](#) 구성을 참조하십시오.

5.2. 스토리지를 사용하여 중앙 사이트 배포

여러 저장소 및 Ceph Storage를 백엔드로 사용하여 이미지 서비스를 배포하려면 다음 단계를 완료합니다.

사전 요구 사항

- 환경과 관련된 **network_data.yaml** 및 **vip_data.yaml** 파일을 생성해야 합니다. **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 에서 샘플 파일을 찾을 수 있습니다.
- 환경과 관련된 **overcloud-baremetal-deploy.yaml** 파일을 생성해야 합니다. 자세한 내용은 [오버클라우드의 베어 메탈 노드 프로비저닝](#)에서 참조하십시오.
- 중앙 위치 및 각 가용성 영역 또는 스토리지 서비스가 필요한 각 지역에 Ceph 클러스터에 대한 하드웨어가 있습니다.
- 중앙 위치 및 각 가용성 영역 또는 스토리지 서비스가 필요한 각 지역에 있는 세 개의 Image 서비스(glance) 서버에 대한 하드웨어가 있습니다. 예지 위치에서 이미지 서비스는 DistributedComputeHCI 노드에 배포됩니다.

프로세스

이미지 서비스(glance)를 여러 저장소와 함께 사용할 수 있도록 Red Hat OpenStack Platform 중앙 위치를 배포합니다.

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

```
[stack@director ~]$ source /home/stack/stackrc
```

3. 환경 파일 /home/stack/central/central-images-env.yaml을 생성합니다.

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file /home/stack/central/central-images-env.yaml
```

4. 환경에 적합한 역할을 사용하여 중앙 위치에 대한 역할을 생성합니다.

```
openstack overcloud roles generate Compute Controller CephStorage \
-o /home/stack/central/central_roles.yaml
```

5. 홈 디렉터리에서 배포하려는 각 스택에 대한 디렉터리를 생성합니다. 중앙 위치의 **network_data.yaml**, **vip_data.yaml** 및 **overcloud-baremetal-deploy.yaml** 템플릿을 **/home/stack/central/**로 이동합니다.

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

6. 오버클라우드의 네트워크를 프로비저닝합니다. 이 명령은 오버클라우드 네트워크의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

7. 오버클라우드의 가상 IP를 프로비저닝합니다. 이 명령은 가상 IP의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

8. 베어 메탈 인스턴스를 프로비저닝합니다. 이 명령은 베어 메탈 노드의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

9. 하이퍼컨버지드 스토리지를 사용하여 중앙 위치를 배포하는 경우 다음 매개 변수를 사용하여 **initial-ceph.conf** 구성 파일을 생성해야 합니다. 자세한 내용은 [HCI용 Red Hat Ceph Storage 클러스터 구성](#)을 참조하십시오.

```
[osd]
osd_memory_target_autotune = true
osd_numa_auto_affinity = true
[mgr]
mgr/cephadm/autotune_memory_target_ratio = 0.2
```

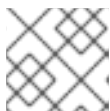
10. **deployed_metal.yaml** 파일을 **openstack overcloud ceph deploy** 명령에 대한 입력으로 사용합니다. **openstack overcloud ceph deploy** 명령은 배포된 Ceph 클러스터를 설명하는 yaml 파일을 출력합니다.

```
openstack overcloud ceph deploy \
--stack central \
/home/stack/central/deployed_metal.yaml \
--config /home/stack/central/initial-ceph.conf \ 1
--output /home/stack/central/deployed_ceph.yaml \
--container-image-prepare /home/stack/containers.yaml \
--network-data /home/stack/network-data.yaml \
--cluster central \
--roles-data /home/stack/central/central_roles.yaml
```

1 하이퍼컨버지드 인프라를 배포할 때만 initial-ceph.conf를 포함합니다.

11. 계속하기 전에 기능적인 Ceph 배포를 확인합니다. **ssh**를 사용하여 **ceph-mon** 서비스를 실행하는 서버에 연결합니다. HCI 배포에서 이는 컨트롤러 노드입니다. 다음 명령을 실행합니다.

```
cephadm shell --config /etc/ceph/central.conf \
--keyring /etc/ceph/central.client.admin.keyring
```



참고

--config 및 **--keyring** 매개 변수를 사용해야 합니다.

12. **site-name.yaml** 환경 파일에서 사이트에 대한 이름 지정 규칙을 구성합니다. Nova 가용성 영역과 Cinder 스토리지 가용성 영역이 일치해야 합니다.

```
parameter_defaults:
  NovaComputeAvailabilityZone: central
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: central
  GlanceBackendID: central
```

13. 다음과 유사한 내용을 사용하여 glance.yaml 템플릿을 구성합니다.

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  GlanceBackendID: central
  CephClusterName: central
```

14. 중앙 위치의 스택을 배포합니다.

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r /home/stack/central/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e ~/central/glance.yaml
```

15. 중앙 위치에 오버클라우드를 배포한 후 에지 사이트의 추가 스택 배포를 위해 입력으로 필요한 데이터를 내보내서 **/home/stack/overcloud-deploy** 디렉터리에 배치됩니다. **central-export.yaml** 파일이 있는지 확인합니다.

```
stat /home/stack/overcloud-deploy/central/central-export.yaml
```

16. Ceph 관련 데이터를 내보냅니다.

```
openstack overcloud export ceph \
--stack central \
--output-file /home/stack/dcn-common/central_ceph_external.yaml
```

5.3. 외부 CEPH 통합

DCN(Distributed Compute Node) 아키텍처의 중앙 위치를 배포하고 사전 배포된 Red Hat Ceph Storage 솔루션을 통합할 수 있습니다. director 없이 Red Hat Ceph Storage를 배포할 때 해당 환경의 Red Hat

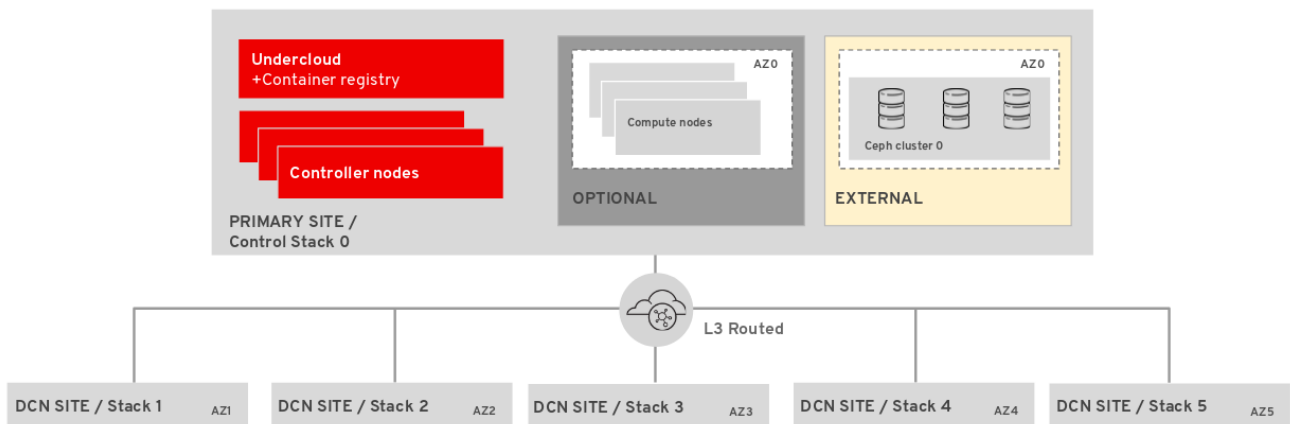
Ceph 스토리지에 대한 정보가 director에 없습니다. **openstack overcloud export ceph** 명령은 실행할 수 없으며 **central_ceph_external.yaml** 을 수동으로 생성해야 합니다.

사전 요구 사항

- 환경과 관련된 **network_data.yaml** 및 **vip_data.yaml** 파일을 생성해야 합니다. **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 에서 샘플 파일을 찾을 수 있습니다.
- 환경과 관련된 **overcloud-baremetal-deploy.yaml** 파일을 생성해야 합니다. 자세한 내용은 [오버클라우드의 베어 메탈 노드 프로비저닝에서 참조하십시오](#).
- 중앙 위치 및 각 가용성 영역의 Ceph 클러스터 또는 스토리지 서비스가 필요한 각 지리적 위치에 대한 하드웨어입니다.

다음은 두 개 이상의 스택을 배포하는 예입니다.

- **central** 이라는 중앙 위치에 있는 하나의 스택입니다.
- **dcn0** 이라는 에지 사이트에 있는 하나의 스택입니다.
- **dcn1,dcn2** 등과 같이 **dcn0** 과 유사하게 배포된 추가 스택입니다.



프로세스

기존 Red Hat Ceph Storage 클러스터 통합에 설명된 프로세스에 따라 기존 Red Hat Ceph Storage 솔루션과 통합 되도록 중앙 위치를 설치할 수 있습니다. Red Hat Ceph Storage를 DCN 배포의 중앙 사이트와 통합할 특별한 요구 사항은 없지만 오버클라우드를 배포하기 전에 DCN 특정 단계를 완료해야 합니다.

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

```
[stack@director ~]$ source ~/stackrc
```

3. ~/central/central-images-env.yaml 환경 파일을 생성

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/central/central-images-env.yaml
```

4. 홈 디렉터리에서 배포하려는 각 스택에 대한 디렉터리를 생성합니다. 이를 사용하여 해당 사이트에 맞게 설계된 템플릿을 분리합니다. 중앙 위치의 **network_data.yaml**, **vip_data.yaml** 및 **overcloud-baremetal-deploy.yaml** 템플릿을 **/home/stack/central/**로 이동합니다.

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

5. 오버클라우드의 네트워크를 프로비저닝합니다. 이 명령은 오버클라우드 네트워크의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. 오버클라우드의 가상 IP를 프로비저닝합니다. 이 명령은 가상 IP의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

7. 베어 메탈 인스턴스를 프로비저닝합니다. 이 명령은 베어 메탈 노드의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

8. **site-name.yaml** 환경 파일에서 사이트에 대한 이름 지정 규칙을 구성합니다. Compute(nova) 가용성 영역과 Block Storage(cinder) 가용성 영역이 일치해야 합니다.

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
  NovaComputeAvailabilityZone: central
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: central
  GlanceBackendID: central
EOF
```

9. 다음과 유사한 콘텐츠를 사용하여 **external-ceph.yaml** 템플릿을 구성합니다.

```
parameter_defaults:
  CinderEnableScsiBackend: false
  CinderEnableRbdBackend: true
```

```

CinderEnableNfsBackend: false
NovaEnableRbdBackend: true
GlanceBackend: rbd
GlanceBackendID: central
GlanceEnabledImportMethods: web-download,copy-image
GlanceStoreDescription: 'central rbd glance store'
CinderRbdPoolName: "openstack-cinder"
NovaRbdPoolName: "openstack-nova"
GlanceRbdPoolName: "openstack-images"
CinderBackupRbdPoolName: "automation-backups"
GnocchiRbdPoolName: "automation-metrics"
CephClusterFSID: 38dd387e-837a-437c-891c-7fc69e17a3c
CephClusterName: central
CephExternalMonHost: 10.9.0.1,10.9.0.2,10.9.0.3
CephClientKey: "AQAKtECeLemfiBBdQp7cjNYQRGW9y8GnhhFZg=="
CephClientUserName: "openstack"

```

10. 중앙 위치를 배포합니다.

```

openstack overcloud deploy \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-n /home/stack/central/network-data.yaml \
...
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/external-ceph.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/external-ceph.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/central_roles.yaml

```

11. 중앙 위치에 오버클라우드를 배포한 후 에지 사이트의 추가 스택 배포를 위해 입력으로 필요한 데이터를 내보내서 **/home/stack/overcloud-deploy** 디렉터리에 배치됩니다. 이 control-plane-export.yaml 파일이 있는지 확인합니다.

```
stat ~/overcloud-deploy/control-plane/control-plane-export.yaml
```

12. Red Hat Ceph Storage 배포에 대한 세부 정보를 사용하여 **central_ceph_external.yaml** 이라는 환경 파일을 생성합니다. 이 파일은 에지 사이트의 추가 스택 배포로 전달할 수 있습니다.

```

parameter_defaults:
  CephExternalMultiConfig:
    - cluster: "central"
      fsid: "3161a3b4-e5ff-42a0-9f53-860403b29a33"
      external_cluster_mon_ips: "172.16.11.84, 172.16.11.87, 172.16.11.92"
      keys:
        - name: "client.openstack"
      caps:
        mgr: "allow *"
        mon: "profile rbd"
        osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
      key: "AQD29WteAAAAABAaphgOjFD7nyjdYe8Lz0mQ5Q=="

```

```

mode: "0600"
dashboard_enabled: false
ceph_conf_overrides:
  client:
    keyring: /etc/ceph/central.client.openstack.keyring

```

- **fsid** 매개 변수는 Ceph Storage 클러스터의 파일 시스템 ID입니다. 이 값은 **[global]** 섹션의 클러스터 구성 파일에 지정됩니다.

```

[global]
fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
...

```

- **key** 매개 변수는 openstack 계정의 ceph 클라이언트 키입니다.

```

[root@ceph ~]# ceph auth list
...
[client.openstack]
key = AQC+vYNXgDAgAhAAc8UoYt+OTz5uhV7ltLdwUw==
caps mgr = "allow *"
caps mon = "profile rbd"
caps osd = "profile rbd pool=volumes, profile rbd pool=vms, profile rbd pool=images,
profile rbd pool=backups, profile rbd pool=metrics"
...

```

샘플 **central_ceph_external.yaml** 파일에 표시된 매개 변수에 대한 자세한 내용은 [사용자 지정 환경 파일 생성](#)을 참조하십시오.

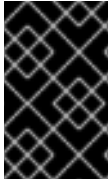
추가 리소스

- [외부 Ceph Storage 클러스터 통합 확인](#)

6장. 스토리지 없이 에지 배포

중앙 위치에서 Image 서비스(glance)의 백엔드로 Object Storage 서비스(swift)를 사용하는 경우 에지 사이트에서 블록 스토리지 없이 DCN(Distributed Compute node) 클러스터를 배포할 수 있습니다. 블록 스토리지 없이 사이트를 배포하는 경우 나중에 블록 스토리지를 갖도록 업데이트할 수 없습니다.

스토리지 없이 에지 사이트를 배포할 때 **컴퓨팅** 역할을 사용합니다.

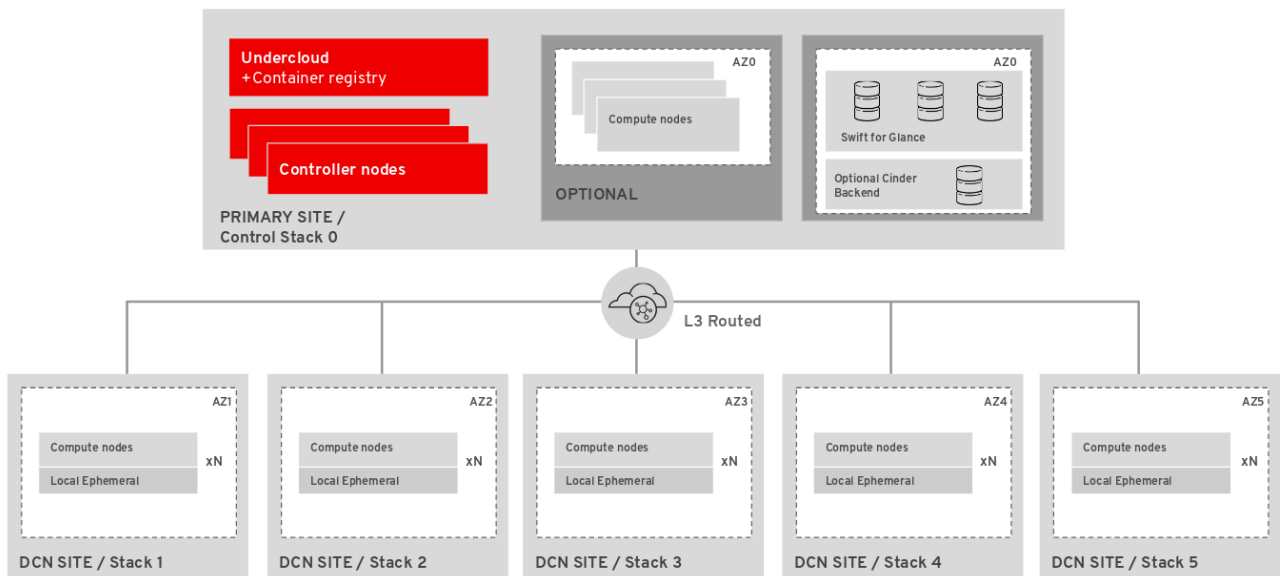


중요

다음 절차에서는 lvm을 Block Storage 서비스(cinder)의 백엔드로 사용하며 프로덕션에서는 지원되지 않습니다. 인증된 블록 스토리지 솔루션을 블록 스토리지 서비스의 백엔드로 배포해야 합니다.

6.1. 스토리지가 없는 DCN 에지 사이트의 아키텍처

이 아키텍처를 배포하려면 **Compute** 역할을 사용합니다.



엣지에 블록 스토리지 없음

- 컨트롤 플레인의 Object Storage(swift) 서비스는 Image(glance) 서비스 백엔드로 사용됩니다.
- Multi-backend 이미지 서비스를 사용할 수 없습니다.
 - 이미지는 Nova의 에지 사이트에 로컬로 캐시됩니다. 자세한 내용은 [11장. nova에 Glance 이미지 사전 연결](#)에서 참조하십시오.
- 인스턴스는 컴퓨팅 노드에 로컬로 저장됩니다.
- Block Storage(cinder)와 같은 볼륨 서비스는 에지 사이트에서 사용할 수 없습니다.



중요

Red Hat Ceph 스토리지로 중앙 위치를 배포하지 않으면 나중에 스토리지가 있는 에지 사이트를 배포할 수 없습니다.

에지에서 블록 스토리지 없이 배포하는 방법에 대한 자세한 내용은 [6.2절. "스토리지 없이 에지 노드 배포"](#) 을 참조하십시오.

6.2. 스토리지 없이 에지 노드 배포

에지 사이트에 컴퓨팅 노드를 배포할 때 중앙 위치를 컨트롤 플레인으로 사용합니다. 배포에 새 DCN 스택을 추가하고 중앙 위치의 구성 파일을 재사용하여 새 환경 파일을 생성할 수 있습니다.

사전 요구 사항

- 환경과 관련된 **network_data.yaml** 파일을 생성해야 합니다. **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 에서 샘플 파일을 찾을 수 있습니다.
- 환경과 관련된 **overcloud-baremetal-deploy.yaml** 파일을 생성해야 합니다. 자세한 내용은 [오버클라우드의 베어 메탈 노드 프로비저닝](#)에서 참조하십시오.
- 이미지를 에지 사이트에 복사하기 전에 중앙 위치에 업로드해야 합니다. 각 이미지의 사본은 중앙 위치의 Image 서비스(glance)에 있어야 합니다.
- Image, Compute, Block Storage 서비스에 RBD 스토리지 드라이버를 사용해야 합니다.

프로세스

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

```
[stack@director ~]$ source ~/stackrc
```

3. 환경 파일 ~/dcn0/dcn0-images-env.yaml[d]을 생성합니다.

```
sudo[e] openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/dcn0/dcn0-images-env.yaml
```

4. 에지 위치에 대한 역할 파일을 생성합니다. 환경에 적합한 역할을 사용하여 옛지 위치에 대한 역할을 생성합니다.

```
(undercloud)$ openstack overcloud roles \
generate Compute \
-o /home/stack/dcn0/dcn0_roles.yaml
```

5. 네트워킹 오버레이에 ML2/OVS를 사용하는 경우 Compute 역할에 **NeutronDhcpAgent** 및 **NeutronMetadataAgent** 서비스가 포함되어 있어야 합니다.

- a. Compute 역할에 대한 역할 파일을 생성합니다.

```
openstack overcloud roles \
generate Compute \
-o /home/stack/dcn0/dcn0_roles.yaml
```

- b. **NeutronDhcpAgent** 및 **NeutronMetadataAgent** 서비스를 포함하도록 /home/stack/dcn0/dcn0_roles.yaml 파일을 편집합니다.

```
...
- OS::TripleO::Services::MySQLClient
- OS::TripleO::Services::NeutronBgpVpnBagpipe
+ - OS::TripleO::Services::NeutronDhcpAgent
+ - OS::TripleO::Services::NeutronMetadataAgent
- OS::TripleO::Services::NeutronLinuxbridgeAgent
- OS::TripleO::Services::NeutronVppAgent
- OS::TripleO::Services::NovaAZConfig
- OS::TripleO::Services::NovaCompute
...
```

자세한 내용은 [라우팅된 공급자 네트워크 준비](#)를 참조하십시오.

6. 오버클라우드의 네트워크를 프로비저닝합니다. 이 명령은 오버클라우드 네트워크의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/dcn0/overcloud-networks-deployed.yaml \
/home/stack/dcn0/network_data.yaml
```

중요

network_data.yaml 템플릿에 중앙 위치의 네트워크를 프로비저닝할 때 포함되지 않은 추가 네트워크가 포함된 경우 중앙 위치에서 네트워크 프로비저닝 명령을 다시 실행해야 합니다.

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

7. 베어 메탈 인스턴스를 프로비저닝합니다. 이 명령은 베어 메탈 노드의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
(undercloud)$ openstack overcloud node provision \
--stack dcn0 \
--network-config \
-o /home/stack/dcn0/deployed_metal.yaml \
~/overcloud-baremetal-deploy.yaml
```

8. site-name.yaml 환경 파일에서 사이트에 대한 이름 지정 규칙을 구성합니다.

```
parameter_defaults:
  NovaComputeAvailabilityZone: dcn0
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: dcn0
  NovaCrossAZAttach: false
```

9. dcn0 에지 사이트의 스택을 배포합니다.

```
openstack overcloud deploy \
--deployed-server \
--stack dcn0 \
--templates /usr/share/openstack-tripleo-heat-templates/ \
```

```
-r /home/stack/dcn0/dcn0_roles.yaml \
-n /home/stack/network_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/overcloud-deploy/central/central-export.yaml \
-e /home/stack/dcn0/overcloud-networks-deployed.yaml \
-e /home/stack/dcn0/overcloud-vip-deployed.yaml \
-e /home/stack/dcn0/deployed_metal.yaml
```

6.3. 에지에서 특정 이미지 유형 제외

기본적으로 컴퓨팅 노드는 지원하는 모든 이미지 형식을 알립니다. 컴퓨팅 노드에서 Ceph 스토리지를 사용하지 않는 경우 이미지 형식 광고에서 RAW 이미지를 제외할 수 있습니다. RAW 이미지 형식은 QCOW2 이미지보다 더 많은 네트워크 대역폭과 로컬 스토리지를 사용하며 Ceph 스토리지가 없는 에지 사이트에서 사용할 때 비효율적입니다. 특정 이미지 형식을 제외하려면 **NovalImageTypeExcludeList** 매개변수를 사용합니다.



중요

Ceph에는 RAW 이미지가 필요하므로 Ceph의 에지 사이트에서 이 매개변수를 사용하지 마십시오.



참고

RAW 이미지를 알리지 않는 컴퓨팅 노드는 RAW 이미지에서 생성된 인스턴스를 호스팅할 수 없습니다. 이는 snapshot-redeploy 및 보류에 영향을 미칠 수 있습니다.

사전 요구 사항

- Red Hat OpenStack Platform director가 설치됨
- 중앙 위치 설치
- DCN 배포에 컴퓨팅 노드 사용 가능

절차

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. 사용자 지정 환경 파일 중 하나에 **NovalImageTypeExcludeList** 매개변수를 포함합니다.

```
parameter_defaults:
  NovalImageTypeExcludeList:
    - raw
```

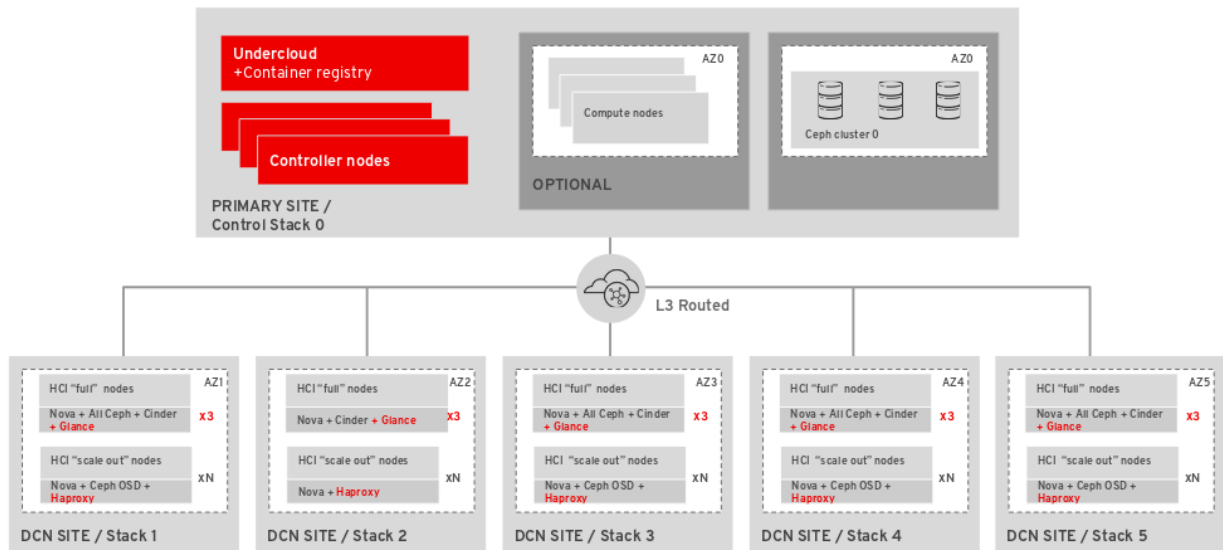
4. 배포와 관련된 기타 환경 파일과 함께 오버클라우드 배포 명령에 **NovalImageTypeExcludeList** 매개변수가 포함된 환경 파일을 포함합니다.

```
openstack overcloud deploy --templates \
```

```
-n network_data.yaml \  
-r roles_data.yaml \  
-e <environment_files> \  
-e <new_environment_file>
```

7장. 엣지에 스토리지 배포

Red Hat OpenStack Platform director를 활용하여 Red Hat OpenStack Platform 및 Ceph Storage를 사용하는 이점과 함께 엣지에서 분산 이미지 관리 및 영구 스토리지를 포함하도록 분산 컴퓨팅 노드 배포를 확장할 수 있습니다.



7.1. 스토리지를 사용한 엣지 배포의 역할

스토리지가 있는 엣지 배포에 다음 역할을 사용할 수 있습니다. 선택한 구성에 따라 환경에 적합한 역할을 선택합니다.

7.1.1. 하이퍼컨버지드 노드가 없는 스토리지

스토리지를 사용하여 엣지를 배포하고 하이퍼컨버지드 노드를 배포하지 않는 경우 다음 4개의 역할 중 하나를 사용합니다.

DistributedCompute

DistributedCompute 역할은 스토리지 배포의 처음 세 compute 노드에 사용됩니다.

DistributedCompute 역할에는 **GlanceApiEdge** 서비스가 포함되어 있어 중앙 허브 위치가 아닌 로컬 엣지 사이트에서 이미지 서비스를 사용할 수 있습니다. 추가 노드의 경우

DistributedComputeScaleOut 역할을 사용합니다.

DistributedComputeScaleOut

DistributedComputeScaleOut 역할에는 DistributedComputeScaleOut 역할에서 생성된 인스턴스가 이미지 서비스에 대한 요청을 엣지 사이트에서 해당 서비스를 제공하는 노드에 프록시할 수 있는

HAproxyEdge 서비스가 포함되어 있습니다. DistributedCompute의 역할이 있는 세 개의 노드를 배포한 후 **DistributedCompute** ScaleOut 역할을 사용하여 컴퓨팅 리소스를 확장할 수 있습니다.

DistributedComputeScaleOut 역할을 사용하여 배포하는 데 필요한 최소 호스트 수가 없습니다.

CephAll

CephAll 역할에는 Ceph OSD, Ceph mon, Ceph Mgr 서비스가 포함됩니다. CephAll 역할을 사용하여 최대 3개의 노드를 배포할 수 있습니다. 추가 스토리지 용량의 경우 CephStorage 역할을 사용합니다.

CephStorage

CephStorage 역할에는 Ceph OSD 서비스가 포함됩니다. 3개의 CephAll 노드가 충분한 스토리지 용량을 제공하지 않으면 필요한 만큼 CephStorage 노드를 추가합니다.

7.1.2. 하이퍼컨버지드 노드가 있는 스토리지

스토리지를 사용하여 에지를 배포하고 컴퓨팅 및 스토리지를 결합하는 하이퍼컨버지드 노드를 사용하면 다음 두 역할 중 하나를 사용합니다.

DistributedComputeHCI

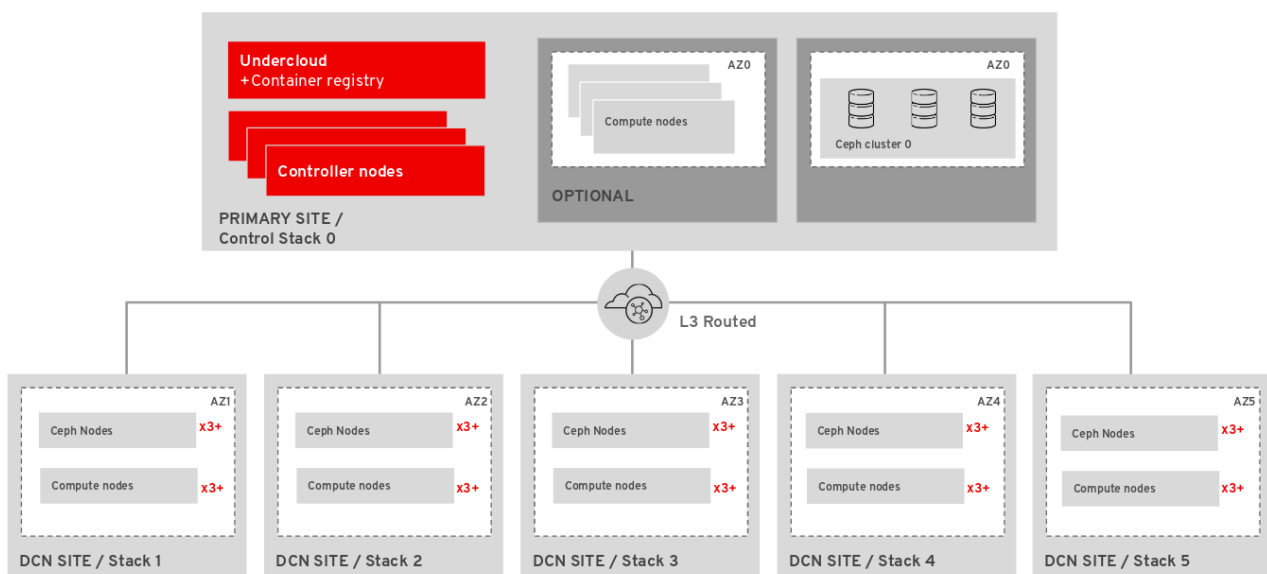
DistributedComputeHCI 역할은 Ceph 관리 및 OSD 서비스를 포함하여 에지에서 하이퍼컨버지드 배포를 활성화합니다. DistributedComputeHCI 역할을 사용할 때 정확히 세 개의 노드를 사용해야 합니다.

DistributedComputeHCIScaleOut

DistributedComputeHCIScaleOut 역할에는 **Ceph OSD** 서비스가 포함되어 있어 에지에 더 많은 노드를 추가할 때 컴퓨팅으로 스토리지 용량을 확장할 수 있습니다. 이 역할에는 이미지 다운로드 요청을 에지 사이트의 **GlanceAPIEdge** 노드로 리디렉션하는 **HAProxyEdge** 서비스도 포함됩니다. 이 역할은 에지에서 하이퍼컨버지드 배포를 활성화합니다. **DistributedComputeHCI** 역할을 사용할 때 정확히 세 개의 노드를 사용해야 합니다.

7.2. 스토리지가 있는 DCN 에지 사이트 아키텍처

스토리지를 사용하여 DCN을 배포하려면 중앙 위치에 Red Hat Ceph Storage도 배포해야 합니다. **dcn-storage.yaml** 및 **cephadm.yaml** 환경 파일을 사용해야 합니다. 하이퍼컨버지드 Red Hat Ceph Storage 노드가 포함된 에지 사이트의 경우 DistributedCompute, **DistributedCompute ScaleOut**, **CephAll**, **CephStorage** 역할을 사용합니다.



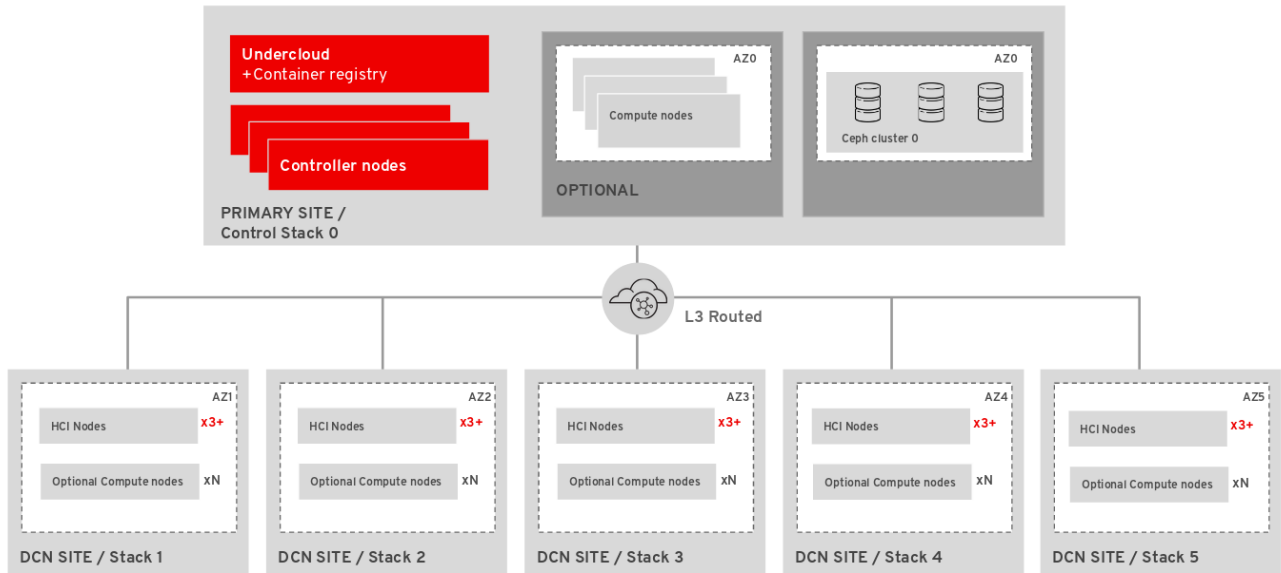
엣지의 블록 스토리지 사용

- Red Hat Ceph Block Devices(RBD)는 Image(glance) 서비스 백엔드로 사용됩니다.
- 중앙과 DCN 사이트 간에 이미지를 복사할 수 있도록 Image 서비스(glance)를 사용할 수 있습니다.
- Block Storage(cinder) 서비스는 모든 사이트에서 사용할 수 있으며 Red Hat Ceph Block Devices(RBD) 드라이버를 사용하여 액세스할 수 있습니다.
- Block Storage(cinder) 서비스는 컴퓨팅 노드에서 실행되며 Red Hat Ceph Storage는 전용 스토리지 노드에서 별도로 실행됩니다.
- Nova 임시 스토리지는 Ceph(RBD)에서 지원합니다.

자세한 내용은 5.2절. “스토리지를 사용하여 중앙 사이트 배포”의 내용을 참조하십시오.

7.3. 하이퍼컨버지드 스토리지를 사용하는 DCN 에지 사이트 아키텍처

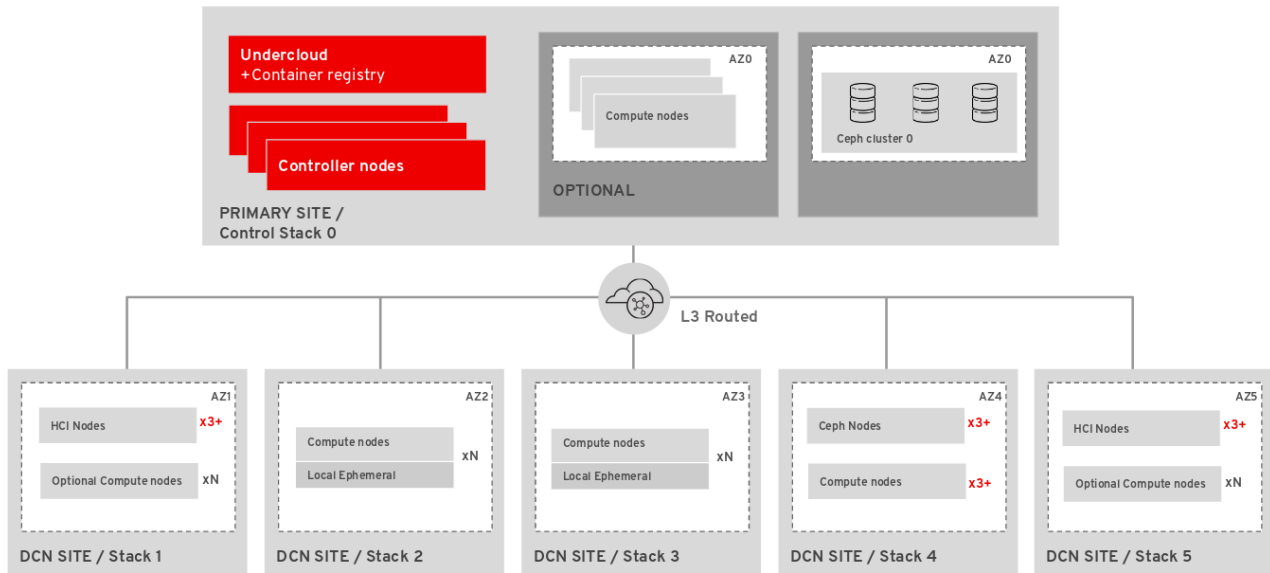
이 구성을 배포하려면 중앙 위치에 Red Hat Ceph Storage도 배포해야 합니다. **dcn-storage.yaml** 및 **cephadm.yaml** 환경 파일을 구성해야 합니다. **DistributedComputeHCI** 및 **DistributedComputeHCIScaleOut** 역할을 사용합니다. **DistributedComputeScaleOut** 역할을 사용하여 Red Hat Ceph Storage 서비스 제공에 참여하지 않는 컴퓨팅 노드를 추가할 수도 있습니다.



에지에서 하이퍼컨버지드 스토리지 사용

- Red Hat Ceph Block Devices(RBD)는 Image(glance) 서비스 백엔드로 사용됩니다.
- 중앙과 DCN 사이트 간에 이미지를 복사할 수 있도록 Image 서비스(glance)를 사용할 수 있습니다.
- Block Storage(cinder) 서비스는 모든 사이트에서 사용할 수 있으며 Red Hat Ceph Block Devices(RBD) 드라이버를 사용하여 액세스할 수 있습니다.
- 블록 스토리지 서비스와 Red Hat Ceph Storage는 모두 컴퓨팅 노드에서 실행됩니다.
자세한 내용은 7.4절. “하이퍼컨버지드 스토리지를 사용하여 에지 사이트 배포”의 내용을 참조하십시오.

분산 컴퓨팅 아키텍처에 Red Hat OpenStack Platform을 배포할 때 각 사이트에 고유한 구성으로 여러 스토리지 토폴로지를 배포할 수 있습니다. 스토리지와 함께 에지 사이트를 배포하려면 Red Hat Ceph 스토리지와 함께 중앙 위치를 배포해야 합니다.



7.4. 하이퍼컨버지드 스토리지를 사용하여 에지 사이트 배포

중앙 사이트를 배포한 후 에지 사이트를 빌드하고 각 엣지 위치가 주로 중앙 위치의 스토리지 백엔드뿐만 아니라 자체 스토리지 백엔드에 연결되도록 합니다. ceph에 필요한 스토리지 및 storage_mgmt 네트워크를 추가하여 이 구성에 스파인 및 리프 네트워킹 구성을 포함해야 합니다. 자세한 내용은 Spine Leaf Networking을 참조하십시오. 사이트 간에 Image 서비스(glance) 이미지를 이동할 수 있도록 중앙 위치에서 스토리지 네트워크와 각 에지 사이트의 스토리지 네트워크 간에 연결해야 합니다.

중앙 위치가 각 에지 사이트에서 mons 및 OSD와 통신할 수 있는지 확인합니다. 그러나 OSD 재조정에 스토리지 관리 네트워크가 사용되므로 사이트 위치 경계에서 스토리지 관리 네트워크를 종료해야 합니다.

사전 요구 사항

- 환경과 관련된 **network_data.yaml** 파일을 생성해야 합니다. **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 에서 샘플 파일을 찾을 수 있습니다.
- 환경과 관련된 **overcloud-baremetal-deploy.yaml** 파일을 생성해야 합니다. 자세한 내용은 [오버클라우드의 베어 메탈 노드 프로비저닝](#)에서 참조하십시오.
- 중앙 위치 및 각 가용성 영역 또는 스토리지 서비스가 필요한 각 지역에 있는 세 개의 Image 서비스(glance) 서버에 대한 하드웨어가 있습니다. 에지 위치에서 이미지 서비스는 DistributedComputeHCI 노드에 배포됩니다.

프로세스

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

```
[stack@director ~]$ source ~/stackrc
```

3. ~/dcn0/dcn0-images-env.yaml 환경 파일을 생성합니다.

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file /home/stack/dcn0/dcn0-images-env.yaml
```

4. dcn0 에지 위치에 대한 적절한 역할을 생성합니다.

```
openstack overcloud roles generate DistributedComputeHCI
DistributedComputeHCIScaleOut \
-o ~/dcn0/dcn0_roles.yaml
```

5. 오버클라우드의 네트워크를 프로비저닝합니다. 이 명령은 오버클라우드 네트워크의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/dcn0/overcloud-networks-deployed.yaml \
/home/stack/network_data.yaml
```

중요

network_data.yaml 템플릿에 중앙 위치의 네트워크를 프로비저닝할 때 포함되지 않은 추가 네트워크가 포함된 경우 중앙 위치에서 네트워크 프로비저닝 명령을 다시 실행해야 합니다.

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. 베어 메탈 인스턴스를 프로비저닝합니다. 이 명령은 베어 메탈 노드의 정의 파일을 입력으로 사용합니다. 명령의 출력 파일을 사용하여 오버클라우드를 배포해야 합니다.

```
(undercloud)$ openstack overcloud node provision \
--stack dcn0 \
--network-config \
-o /home/stack/dcn0/deployed_metal.yaml \
/home/stack/overcloud-baremetal-deploy.yaml
```

7. 하이퍼컨버지드 스토리지를 사용하여 에지 사이트를 배포하는 경우 다음 매개변수를 사용하여 **initial-ceph.conf** 구성 파일을 생성해야 합니다. 자세한 내용은 [HCI를 위한 Red Hat Ceph Storage 클러스터 구성](#)을 참조하십시오.

```
[osd]
osd_memory_target_autotune = true
osd_numa_auto_affinity = true
[mgr]
mgr/cephadm/autotune_memory_target_ratio = 0.2
```

8. **deployed_metal.yaml** 파일을 **openstack overcloud ceph deploy** 명령에 대한 입력으로 사용합니다. **openstack overcloud ceph deploy** 명령은 배포된 Ceph 클러스터를 설명하는 yaml 파일을 출력합니다.

```
openstack overcloud ceph deploy \
/home/stack/dcn0/deployed_metal.yaml \
--stack dcn0 \
--config ~/dcn0/initial-ceph.conf \ 1
--output ~/dcn0/deployed_ceph.yaml \
--container-image-prepare ~/containers.yaml \
```

```
--network-data ~/network-data.yaml \
--cluster dcn0 \
--roles-data dcn_roles.yaml
```

- 1 하이퍼컨버지드 인프라를 배포할 때만 initial-ceph.conf를 포함합니다.

9. site-name.yaml 환경 파일에서 사이트에 대한 이름 지정 규칙을 구성합니다. Nova 가용성 영역과 Cinder 스토리지 가용성 영역이 일치해야 합니다.

```
parameter_defaults:
  NovaComputeAvailabilityZone: dcn0
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: dcn0
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: dcn0
  CinderVolumeCluster: dcn0
  GlanceBackendID: dcn0
```

10. 다음과 유사한 내용을 사용하여 glance.yaml 템플릿을 구성합니다.

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'dcn0 rbd glance store'
  GlanceBackendID: dcn0
  GlanceMultistoreConfig:
    central:
      GlanceBackend: rbd
      GlanceStoreDescription: 'central rbd glance store'
      CephClusterName: central
```

11. dcn0 위치의 스택을 배포합니다.[d]

```
openstack overcloud deploy \
--deployed-server \
--stack dcn0 \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/dcn0/dcn0_roles.yaml \
-n ~/dcn0/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-only.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/overcloud-deploy/central/central-export.yaml \
-e /home/stack/dcn0/deployed_ceph.yaml \
-e /home/stack/dcn-common/central_ceph_external.yaml \
-e /home/stack/dcn0/overcloud-vip-deployed.yaml \
-e /home/stack/dcn0/deployed_metal.yaml \
-e /home/stack/dcn0/overcloud-networks-deployed.yaml \
-e ~/control-plane/glance.yaml
```

7.5. 엣지에 사전 설치된 RED HAT CEPH STORAGE 클러스터 사용

기존 Ceph 클러스터를 사용하도록 Red Hat OpenStack Platform을 구성할 수 있습니다. 이를 외부 Ceph 배포라고 합니다.

사전 요구 사항

- 대기 시간 요구 사항을 초과하지 않도록 DCN 사이트의 로컬인 사전 설치된 Ceph 클러스터가 있어야 합니다.

절차

1. Ceph 클러스터에 다음 풀을 생성합니다. 중앙 위치에 배포하는 경우 **백업** 및 **메트릭** 풀을 포함합니다.

```
[root@ceph ~]# ceph osd pool create volumes <_PGnum_>
[root@ceph ~]# ceph osd pool create images <_PGnum_>
[root@ceph ~]# ceph osd pool create vms <_PGnum_>
[root@ceph ~]# ceph osd pool create backups <_PGnum_>
[root@ceph ~]# ceph osd pool create metrics <_PGnum_>
```

<_PGnum_>을 배치 그룹 수로 바꿉니다. **풀 계산기**당 Ceph PG(배치 그룹)를 사용하여 적절한 값을 결정할 수 있습니다.

2. Ceph에 OpenStack 클라이언트 사용자를 생성하여 적절한 풀에 Red Hat OpenStack Platform 환경 액세스 권한을 제공합니다.

```
ceph auth add client.openstack mon 'allow r' osd 'allow class-read object_prefix rbd_children, allow rwx pool=volumes, allow rwx pool=vms, allow rwx pool=images'
```

반환된 Ceph 클라이언트 키를 저장합니다. 언더클라우드를 구성할 때 이 키를 **CephClientKey** 매개변수 값으로 사용합니다.



참고

중앙 위치에서 이 명령을 실행하고 Cinder 백업 또는 원격 분석 서비스를 사용하려는 경우 allow rwx pool=backups를 추가하여 명령에 pool=metrics를 허용합니다.

3. Ceph Storage 클러스터의 파일 시스템 ID를 저장합니다. Ceph 구성 파일의 **[global]** 섹션에 있는 **fsid** 매개변수 값은 파일 시스템 ID입니다.

```
[global]
fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
...
```

언더클라우드를 구성할 때 이 값을 **CephClusterFSID** 매개변수 값으로 사용합니다.

4. 언더클라우드에서 환경 파일을 생성하여 관리되지 않는 Ceph 클러스터에 연결하도록 노드를 구성합니다. ceph-external-<SITE>.yaml과 같은 인식 가능한 이름 지정 규칙을 사용합니다. 여기서 SITE는 ceph-external-central.yaml, ceph-external-dcn1.yaml 등과 같은 배포의 위치입니다.

```
parameter_defaults:
  # The cluster FSID
  CephClusterFSID: '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  # The CephX user auth key
  CephClientKey: 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ=='
```

```
# The list of IPs or hostnames of the Ceph monitors
CephExternalMonHost: '172.16.1.7, 172.16.1.8, 172.16.1.9'
# The desired name of the generated key and conf files
CephClusterName: dcn1
```

- a. CephClusterFSID 및 CephClientKey 매개변수에 대해 이전에 저장된 값을 사용합니다.
 - b. Ceph 모니터의 쉽표로 구분된 ip 주소 목록을 CephExternalMonHost 매개변수 값으로 사용합니다.
 - c. 에지 사이트 간에 **CephClusterName** 매개변수의 고유 값을 선택해야 합니다. 이름을 다시 사용하면 구성 파일을 덮어씁니다.
5. 중앙 위치에서 Red Hat OpenStack Platform director를 사용하여 Red Hat Ceph 스토리지를 배포한 경우 ceph 설정을 **central_ceph_external.yaml** 환경 파일로 내보낼 수 있습니다. 이 환경 파일은 DCN 사이트를 중앙 허브 Ceph 클러스터에 연결하므로 정보는 이전 단계에서 배포한 Ceph 클러스터에 고유합니다.

```
sudo -E openstack overcloud export ceph \
--stack central \
--output-file /home/stack/dcn-common/central_ceph_external.yaml
```

중앙 위치에 Red Hat Ceph Storage가 외부에 배포된 경우 **openstack overcloud export ceph** 명령을 사용하여 **central_ceph_external.yaml** 파일을 생성할 수 없습니다. 대신 central_ceph_external.yaml 파일을 수동으로 생성해야 합니다.

```
parameter_defaults:
  CephExternalMultiConfig:
    - cluster: "central"
      fsid: "3161a3b4-e5ff-42a0-9f53-860403b29a33"
      external_cluster_mon_ips: "172.16.11.84, 172.16.11.87, 172.16.11.92"
      keys:
        - name: "client.openstack"
          caps:
            mgr: "allow *"
            mon: "profile rbd"
            osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
            key: "AQD29WteAAAAABAphgOjFD7nyjdYe8Lz0mQ5Q=="
            mode: "0600"
      dashboard_enabled: false
      ceph_conf_overrides:
        client:
          keyring: /etc/ceph/central.client.openstack.keyring
```

6. 중앙 위치에 대해 관리되지 않는 Red Hat Ceph Storage 클러스터를 사용하여 각 사이트에 대해 비슷한 세부 정보를 사용하여 환경 파일을 만듭니다. 관리되지 않는 Red Hat Ceph Storage 클러스터가 있는 사이트에서 **openstack overcloud export ceph** 명령이 작동하지 않습니다. 중앙 위치를 업데이트할 때 이 파일은 에지 사이트의 스토리지 클러스터를 보조 위치로 중앙 위치하도록 허용합니다.

```
parameter_defaults:
  CephExternalMultiConfig:
    cluster: dcn1
```

```
...
cluster: dcn2
...
```

7. 오버클라우드를 배포할 때 external-ceph.yaml, ceph-external-<SITE>.yaml 및 central_ceph_external.yaml 환경 파일을 사용합니다.

```
openstack overcloud deploy \
  --stack dcn1 \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/dcn1/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/external-ceph.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
  -e /home/stack/dcn1/ceph-external-dcn1.yaml \
  ....
  -e /home/stack/overcloud-deploy/central/central-export.yaml \
  -e /home/stack/dcn-common/central_ceph_external.yaml \
  -e /home/stack/dcn1/dcn_ceph_keys.yaml \
  -e /home/stack/dcn1/role-counts.yaml \
  -e /home/stack/dcn1/ceph.yaml \
  -e /home/stack/dcn1/site-name.yaml \
  -e /home/stack/dcn1/tuning.yaml \
  -e /home/stack/dcn1/glance.yaml
```

8. 모든 엣지 위치가 배포된 후 중앙 위치를 다시 배포합니다.

7.6. 중앙 위치 업데이트

샘플 절차를 사용하여 모든 엣지 사이트를 구성하고 배포한 후 중앙 이미지 서비스에서 이미지를 엣지 사이트로 푸시할 수 있도록 중앙 위치에서 구성을 업데이트합니다.



주의

이 절차에서는 Image 서비스(glance)를 재시작하고 장기 실행 중인 이미지 서비스 프로세스를 중단합니다. 예를 들어 중앙 이미지 서비스 서버에서 DCN 이미지 서비스 서버로 이미지를 복사하는 경우 해당 이미지 복사가 중단되고 이를 다시 시작해야 합니다. 자세한 내용은 [중단된 이미지 서비스 프로세스 후 Residual 데이터 삭제](#)를 참조하십시오.

절차

1. 다음과 유사한 **~/central/glance_update.yaml** 파일을 만듭니다. 이 예제에는 dcn0 및 dcn1의 두 엣지 사이트에 대한 구성이 포함됩니다.

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  CephClusterName: central
```

```

GlanceBackendID: central
GlanceMultistoreConfig:
  dcn0:
    GlanceBackend: rbd
    GlanceStoreDescription: 'dcn0 rbd glance store'
    CephClientUserName: 'openstack'
    CephClusterName: dcn0
    GlanceBackendID: dcn0
  dcn1:
    GlanceBackend: rbd
    GlanceStoreDescription: 'dcn1 rbd glance store'
    CephClientUserName: 'openstack'
    CephClusterName: dcn1
    GlanceBackendID: dcn1

```

2. **dcn_ceph.yaml** 파일을 생성합니다. 다음 예에서 이 파일은 중앙 사이트에서 glance 서비스를 에지 사이트, **dcn0** 및 **dcn1** 의 Ceph 클러스터 클라이언트로 구성합니다.

```

openstack overcloud export ceph \
--stack dcn0,dcn1 \
--output-file ~/central/dcn_ceph.yaml

```

3. 원래 템플릿을 사용하여 중앙 사이트를 다시 배포하고 새로 생성된 **dcn_ceph.yaml** 및 **glance_update.yaml** 파일을 포함합니다.



참고

중앙 스택을 생성하는 동안 리프 네트워크가 처음에 제공되지 않은 경우 **오버클라우드 배포** 명령에 다른 에지 사이트의 **deployed_metal.yaml** 을 포함합니다.

```

openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/central/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e /home/stack/central/dcn_ceph.yaml \
-e /home/stack/central/glance_update.yaml

```

4. 중앙 위치의 컨트롤러에서 **cinder-volume** 서비스를 다시 시작합니다. **cinder-backup** 서비스를 사용하여 중앙 위치를 배포한 경우 **cinder-backup** 서비스도 다시 시작합니다.

```

ssh tripleo-admin@controller-0 sudo pcs resource restart openstack-cinder-volume
ssh tripleo-admin@controller-0 sudo pcs resource restart openstack-cinder-backup

```

7.6.1. 이미지 서비스 프로세스가 중단된 후 남은 데이터 삭제

중앙 위치를 다시 시작하면 장기 실행 이미지 서비스(glance) 프로세스가 중단됩니다. 이러한 프로세스를 재시작하려면 먼저 재부팅한 컨트롤러 노드와 Ceph 및 이미지 서비스 데이터베이스에서 남은 데이터를 정리해야 합니다.

절차

1. 재부팅된 컨트롤러 노드에서 남은 데이터를 확인하고 지웁니다. 스테이징 저장소의 **glance-api.conf** 파일의 파일을 이미지 서비스 데이터베이스의 해당 이미지(예: < image_ID>.raw)와 비교합니다.
 - 이러한 해당 이미지에 가져오기 상태가 표시되면 이미지를 다시 생성해야 합니다.
 - 이미지에 활성화 상태가 표시되면 스테이징에서 데이터를 삭제하고 복사 가져오기를 다시 시작해야 합니다.
2. Ceph 저장소에서 남은 데이터를 확인하고 지웁니다. 스테이징 영역에서 정리한 이미지에는 이미지가 포함된 Ceph 저장소의 **stores** 속성에 일치하는 레코드가 있어야 합니다. Ceph의 이미지 이름은 이미지 서비스 데이터베이스의 이미지 ID입니다.
3. 이미지 서비스 데이터베이스를 지웁니다. 중단된 가져오기 작업에서 상태를 가져오는 모든 이미지를 지웁니다.

```
$ glance image-delete <image_id>
```

7.7. DCN에 RED HAT CEPH STORAGE 대시보드 배포

절차

Red Hat Ceph Storage 대시보드를 중앙 위치에 배포하려면 [오버클라우드 배포에 Red Hat Ceph Storage 대시보드 추가](#)를 참조하십시오. 이러한 단계는 중앙 위치를 배포하기 전에 완료해야 합니다.

Red Hat Ceph Storage 대시보드를 에지 위치에 배포하려면 중앙에 대해 완료한 것과 동일한 단계를 완료하지만 다음을 완료해야 합니다.

- 고가용성 가상 IP를 만들려면 로드 밸런싱을 위해 자체 솔루션을 배포해야 합니다. 에지 사이트는 haproxy 또는 pacemaker를 배포하지 않습니다. Red Hat Ceph Storage 대시보드를 에지 위치에 배포하면 스토리지 네트워크에 배포가 노출됩니다. 대시보드는 로드 밸런싱 솔루션 없이 별도의 IP 주소가 있는 3개의 DistributedComputeHCI 노드 각각에 설치됩니다.

Ceph 대시보드를 노출할 수 있는 가상 IP를 호스팅할 추가 네트워크를 만들 수 있습니다. 여러 스택에 대해 네트워크 리소스를 재사용해서는 안 됩니다. 네트워크 리소스 재사용에 대한 자세한 내용은 [여러 스택에서 네트워크 리소스 재사용](#)을 참조하십시오.

이 추가 네트워크 리소스를 생성하려면 제공된 **network_data_dashboard.yaml** heat 템플릿을 사용합니다. 생성된 네트워크의 이름은 **StorageDashboard** 입니다.

절차

1. Red Hat OpenStack Platform Director에 **스택**으로 로그인합니다.
2. 환경에 적합한 **DistributedComputeHCIDashboard** 역할 및 기타 역할을 생성합니다.

```
openstack overcloud roles generate DistributedComputeHCIDashboard -o ~/dnc0/roles.yaml
```

3. overcloud deploy 명령에 **roles.yaml** 및 **network_data_dashboard.yaml** 을 포함합니다.

```
$ openstack overcloud deploy --templates \
-r ~/<dcn>/<dcn_site_roles>.yaml \
-n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
-e <overcloud_environment_files> \
...
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-only.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/ceph-dashboard.yaml \
```



참고

배포에서는 스토리지 네트워크에서 대시보드가 활성화된 세 개의 IP 주소를 제공합니다.

검증

대시보드가 중앙 위치에서 작동하고 Ceph 클러스터에서 표시되는 데이터가 올바른지 확인하려면 [Ceph 대시보드 액세스](#)를 참조하십시오.

유사한 단계를 통해 대시보드가 에지 위치에서 작동하는지 확인할 수 있지만 에지 위치에 로드 밸런서가 없기 때문에 예외가 있습니다.

1. 선택한 스택과 관련된 대시보드 관리자 로그인 자격 증명을 검색합니다.

```
grep grafana_admin /home/stack/config-download/<stack>/cephadm/cephadm-extra-vars-heat.yaml
```

2. 선택한 스택과 관련된 인벤토리 내에서 **/home/stack/config-download/<stack>/cephadm/inventory.yaml**에서 DistributedComputeHCI 역할 호스트 목록을 찾아 **storage_ip** 값 3개를 모두 저장합니다. 아래 예제에서 처음 두 대시보드 IP는 172.16.11.84 및 172.16.11.87입니다.

```
DistributedComputeHCI:
  hosts:
    dcn1-distributed-compute-hci-0:
      ansible_host: 192.168.24.16
    ...
    storage_hostname: dcn1-distributed-compute-hci-0.storage.localdomain
    storage_ip: 172.16.11.84
    ...
    dcn1-distributed-compute-hci-1:
      ansible_host: 192.168.24.22
    ...
    storage_hostname: dcn1-distributed-compute-hci-1.storage.localdomain
    storage_ip: 172.16.11.87
```

3. 액세스할 수 있는 경우 이러한 IP 주소 중 하나에서 Ceph 대시보드가 활성화되어 있는지 확인할 수 있습니다. 이러한 IP 주소는 스토리지 네트워크에 있으며 라우팅되지 않습니다. 이러한 IP 주소를 사용할 수 없는 경우 확인을 위해 인벤토리에서 가져올 가상 IP 주소를 가져오도록 세 개의 IP 주소에 대한 로드 밸런서를 구성해야 합니다.

8장. 엣지에서 네트워크 트래픽 로드 밸런싱

엣지 사이트에서 로드 밸런서를 생성하여 트래픽 처리량을 늘리고 RHOSP(Red Hat OpenStack Platform) 로드 밸런싱 서비스(octavia)를 사용하여 대기 시간을 줄일 수 있습니다.

이 섹션에 포함된 항목은 다음과 같습니다.

- 서비스 가용성 영역 로드 밸런싱을 위한 네트워크 리소스 생성
- 로드 밸런싱 서비스 가용성 영역 생성
- 가용성 영역에서 로드 밸런서 생성

8.1. 서비스 가용성 영역 로드 밸런싱을 위한 네트워크 리소스 생성

RHOSP(Red Hat OpenStack Platform) 로드 밸런싱 서비스(octavia) 가용성 영역(AZ)을 생성하려면 RHOSP 관리자여야 합니다. **octavia-dcn-deployment.yaml**.

octavia-dcn-deployment.yaml을 실행하면 로드 밸런싱 서비스 AZ에 필요한 네트워크, 서브넷 및 라우터와 같은 네트워킹 리소스를 생성합니다. 각 AZ 이름 및 관리 네트워크를 지정한 구성 입력 파일 **octavia-dcn-parameters.yaml**에 플레이북을 제공합니다.

플레이북을 실행하고 필요한 네트워킹 리소스를 생성한 후 프로젝트(테넌트) 사용자가 DCN(Distributed Compute node) 로케일에 적합한 로드 밸런서를 생성하기 전에 실제 RHOSP 로드 밸런싱 서비스 AZ를 생성해야 합니다.

이 절차에서는 이름이 **az-central**, **az-dcn1**, **az-dcn2** 라는 3개의 로드 밸런싱 서비스 AZ에 필요한 네트워크 리소스를 생성하는 방법을 보여줍니다. 이러한 로드 밸런싱 서비스 AZ 이름은 Compute 서비스 AZ의 이름과 일치하며 이 배포에 사용되는 3개의 DCN의 이름이기도 합니다.

사전 요구 사항

- 생성하려는 모든 로드 밸런싱 서비스 AZ에 대해 하나의 Compute 서비스(nova) AZ가 있어야 합니다.
- 생성하려는 모든 로드 밸런싱 서비스 AZ에 대해 하나의 Networking 서비스(neutron) AZ가 있어야 합니다. 이러한 네트워킹 서비스 AZ는 Compute 서비스 AZ의 이름과 일치해야 합니다.
- 로드 밸런싱 서비스 공급자 드라이버는 amphora여야 합니다. OVN 공급자 드라이버는 AZ를 지원하지 않습니다.
- **admin** 역할이 있는 RHOSP 사용자여야 합니다.

프로세스

1. 자격 증명 파일을 가져옵니다.

예제

```
$ source ~/centralrc
```

2. **octavia-dcn-parameters.yaml** 파일을 생성하고 아래 표시된 구문을 사용하여 로드 밸런싱 서비스 AZ 및 해당 관리 네트워크를 추가하여 Ansible 플레이북에서 필요한 네트워킹 리소스를 생성할 수 있습니다.

octavia_controller_AZ_name 값은 모든 로드 밸런서 서비스가 실행되는 AZ의 이름입니다.

```

octavia_controller_availability_zone: <octavia_controller_AZ_name>
octavia_availability_zones:
  <octavia_controller_AZ_name>: # no cidr needed, it uses the already existing subnet
  <octavia_AZ_n>:
    lb_mgmt_subnet_cidr: <CIDR_address_n>
  <octavia_AZ_n2>:
    lb_mgmt_subnet_cidr: <CIDR_address_n2>

```



중요

지정한 로드 밸런싱 서비스 AZ의 이름은 기존 Compute 서비스 AZ의 이름과 일치해야 합니다. **openstack availability zone list --compute** 를 실행하여 Compute 서비스 AZ의 이름을 가져올 수 있습니다.

Ansible Playbook은 각 AZ에 대한 네트워크, 서브넷 및 라우터를 생성하고 이 규칙에 따라 **octavia-dcn-parameters.yaml** 에 지정하는 AZ 이름을 사용하여 이름을 지정합니다. **lb-mgmt-<AZ_name>-net**, **lb-mgmt-<AZ_name>-subnet**, **lb-mgmt-<AZ_name>-router** 예외는 **octavia_controller_AZ_name**: 플레이북에서 기존 부하 분산 관리 네트워크 및 서브넷, **lb-mgmt-net** 및 **lb-mgmt-subnet.NET**을 각각 사용하여 이름을 지정하는 관련 라우터를 생성합니다.

이 예에서는 **az-central**, **az-dcn1** 및 **az-dcn2**. **az-central** AZ는 기존 부하 분산 관리 네트워크인 **lb-mgmt-net** 을 사용합니다. 다른 두 AZ는 각각 **172.47.0.0/16** 및 **172.48.0.0/16** 을 사용합니다.

예제

```

octavia_controller_availability_zone: az-central
octavia_availability_zones:
  az-central: # no cidr needed; it uses the existing subnet
  az-dcn1:
    lb_mgmt_subnet_cidr: 172.47.0.0/16
  az-dcn2:
    lb_mgmt_subnet_cidr: 172.48.0.0/16

```

3. Ansible 플레이북 **octavia-dcn-deployment.yaml** 을 실행하고 **octavia-dcn-parameters.yaml** 에서 생성한 AZ 정의를 포함합니다.

예제

```

$ ansible-playbook -i overcloud-deploy/central/config-download/
central/tripleo-ansible-inventory.yaml \
/usr/share/ansible/tripleo-playbooks/octavia-dcn-deployment.yaml \
-e @octavia-dcn-parameters.yaml -e stack=central -v

```

검증

1. 필요한 **lb-mgmt-*** 서브넷이 있는지 확인합니다.

```
$ openstack subnet list -c Name -c Subnet
```

샘플 출력

```
+-----+-----+
```

```

| Name                | Subnet                |
+-----+-----+
| lb-mgmt-az-dcn2-subnet | 172.48.0.0/16 |
| segment5              | 10.0.20.0/24  |
| segment3              | 10.101.30.0/24 |
| segment2              | 10.101.20.0/24 |
| lb-mgmt-az-dcn1-subnet | 172.47.0.0/16  |
| heat_tempestconf_subnet | 192.168.199.0/24 |
| segment4              | 10.0.10.0/24   |
| lb-mgmt-subnet         | 172.24.0.0/16  |
| segment1              | 10.101.10.0/24 |
| lb-mgmt-backbone-subnet | 172.49.0.0/16  |
| segment6              | 10.0.30.0/24   |
+-----+-----+

```

2. 필요한 가상 라우터가 있는지 확인합니다.

```
$ openstack router list -c Name -c Status
```

샘플 출력

```

+-----+-----+
| Name                | Status |
+-----+-----+
| lb-mgmt-az-dcn2-router | ACTIVE |
| lb-mgmt-az-dcn1-router | ACTIVE |
| lb-mgmt-router        | ACTIVE |
+-----+-----+

```

다음 단계

- 로드 밸런싱 서비스를 위한 가용성 영역 생성

8.2. 로드 밸런싱 서비스를 위한 가용성 영역 생성

RHOSP(Red Hat OpenStack Platform) 로드 밸런싱 서비스(octavia)를 사용하면 RHOSP 관리자가 프로젝트 사용자가 분산 컴퓨팅 노드(DCN)에서 로드 밸런서를 생성하여 트래픽 처리량을 늘리고 대기 시간을 줄일 수 있는 AZ(가용성 영역)를 생성할 수 있습니다.

로드 밸런싱 서비스 AZ를 생성하려면 두 단계가 필요합니다. RHOSP 관리자는 먼저 AZ 프로필을 생성한 다음 프로필을 사용하여 사용자에게 표시되는 로드 밸런싱 서비스 AZ를 생성해야 합니다.

AZ 프로필에는 다음이 있어야 합니다.

- Compute 서비스(nova) AZ의 이름입니다.
- 사용할 관리 네트워크입니다.
각 AZ에 대해 하나의 고유 네트워크인 여러 관리 네트워크가 있습니다. 중앙 AZ는 기존 부하 분산 관리 네트워크인 **lb-mgmt-net** 을 사용하고 추가 AZ는 해당 네트워크인 **lb-mgmt-<AZ_name>-net** 을 사용합니다(예: **lb-mgmt-az-dcn1-net**, **lb-#159-az-dcn2-net** 등).

사전 요구 사항

- **octavia-dcn-deployment.yaml** Ansible 플레이북을 실행하여 필요한 네트워킹 리소스가 생성된 DCN 환경이 있어야 합니다.
자세한 내용은 [서비스 가용성 영역 로드 밸런싱을 위한 네트워크 리소스 생성](#) 을 참조하십시오.
- 로드 밸런싱 서비스 공급자 드라이버는 amphora여야 합니다. OVN 공급자 드라이버는 AZ를 지원하지 않습니다.
- **admin** 역할이 있는 RHOSP 사용자여야 합니다.

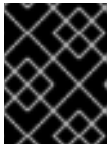
프로세스

1. 자격 증명 파일을 가져옵니다.

예제

```
$ source ~/centralrc
```

2. 로드 밸런싱 서비스 AZ의 이름을 수집하는 데 사용할 Compute 서비스 AZ의 이름을 수집합니다.



중요

생성하는 로드 밸런싱 서비스 AZ의 이름은 Compute 서비스 AZ의 이름과 일치해야 합니다.

```
$ openstack availability zone list --compute
```

샘플 출력

```
+-----+-----+
| Zone Name | Zone Status |
+-----+-----+
| az-central | available |
| az-dcn1    | available |
| az-dcn2    | available |
| internal   | available |
+-----+-----+
```

3. 로드 밸런싱 서비스 AZ를 생성하는 데 사용할 관리 네트워크의 ID를 수집합니다.

```
$ openstack network list -c Name -c ID
```

샘플 출력

```
+-----+-----+
| ID                               | Name                               |
+-----+-----+
| 10458d6b-e7c9-436f-92d9-711677c9d9fd | lb-mgmt-az-dcn2-net      |
| 662a94f5-51eb-4a4c-86c4-52dcbf471ef9 | lb-mgmt-net              |
| 6b97ef58-2a25-4ea5-931f-b7c07cd09474 | lb-mgmt-backbone-net     |
| 99f4215b-fad8-432d-8444-1f894154dc30 | heat_tempestconf_network |
| a2884aaf-846c-4936-9982-3083f6a71d9b | lb-mgmt-az-dcn1-net     |
```

```
| d7f7de6c-0e84-49e2-9042-697fa85d2532 | public |
| e887a9f9-15f7-4854-a797-033cedbfe5f3 | public2 |
+-----+-----+
```

4. AZ 프로필을 만듭니다. 이 단계를 반복하여 생성할 각 로드 밸런싱 서비스 AZ에 대한 AZ 프로필을 생성합니다.

```
$ openstack loadbalancer availabilityzoneprofile create \
--name <AZ_profile_name> --provider amphora --availability-zone-data '{"compute_zone": "
<compute_AZ_name>","management_network": "<lb_mgmt_AZ_net_UUID>"}
```

예 - az-central의 프로필 생성

이 예에서는 Compute AZ(**az-central**)에서 실행되는 컴퓨팅 노드의 관리 네트워크(**lb-mgmt-net**)를 사용하는 AZ 프로필(**az_profile_central**)이 생성됩니다.

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az_profile_central --provider amphora --availability-zone-data \
'{"compute_zone": "az-central","management_network": \
"662a94f5-51eb-4a4c-86c4-52dcbf471ef9"}'
```

5. 4단계를 반복하여 생성하려는 각 로드 밸런싱 서비스 AZ에 대한 AZ 프로필을 생성합니다.

예 - az-dcn1의 프로필 생성

이 예에서는 Compute AZ(**az-dcn1**)에서 실행되는 컴퓨팅 노드의 관리 네트워크(**lb-mgmt-az-dcn1-net**)를 사용하는 AZ 프로필(**az-profile-dcn1**)이 생성됩니다.

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn1 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn1","management-network": \
"a2884aaf-846c-4936-9982-3083f6a71d9b"}'
```

예 - az-dcn2의 프로필 생성

이 예에서는 Compute AZ(**az-dcn2**)에서 실행되는 컴퓨팅 노드의 관리 네트워크(**lb-mgmt-az-dcn2-net**)를 사용하는 AZ 프로필(**az-profile-dcn2**)이 생성됩니다.

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn2 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn2","management-network": \
"10458d6b-e7c9-436f-92d9-711677c9d9fd"}'
```

6. AZ 프로필을 사용하여 로드 밸런싱 서비스 AZ를 생성합니다. 각 AZ에 적절한 프로필을 사용하여 추가 AZ에 대해 이 단계를 반복합니다.

예 - AZ 생성: az-central

이 예에서는 AZ 프로필(**az-profile-central**)을 사용하여 로드 밸런싱 서비스 AZ(**az-central**)가 생성됩니다.

```
$ openstack loadbalancer availabilityzone create --name az-central \
--availabilityzoneprofile az-profile-central \
--description "AZ for Headquarters" --enable
```

예 - AZ 생성: az-dcn1

이 예에서는 AZ 프로파일(**az-profile-az-dcn1**)을 사용하여 로드 밸런싱 서비스 AZ(**az-dcn1**)가 생성됩니다.

```
$ openstack loadbalancer availabilityzone create --name az-dcn1 \
--availabilityzoneprofile az-profile-az-dcn1 \
--description "AZ for South Region" --enable
```

예 - AZ 생성: az-dcn2

이 예에서는 AZ 프로파일(**az-profile-az-dcn2**)을 사용하여 로드 밸런싱 서비스 AZ(**az-dcn2**)가 생성됩니다.

```
$ openstack loadbalancer availabilityzone create --name az-dcn2 \
--availabilityzoneprofile az-profile-az-dcn2 \
--description "AZ for North Region" --enable
```

검증

- AZ(**az-central**)가 생성되었는지 확인합니다. 각 AZ에 적절한 이름을 사용하여 추가 AZ에 대해 이 단계를 반복합니다.

예 - az-central 확인

```
$ openstack loadbalancer availabilityzone show az-central
```

샘플 출력

```
+-----+-----+
| Field          | Value                                |
+-----+-----+
| name           | az-central                          |
| availability_zone_profile_id | 5ed25d22-52a5-48ad-85ec-255910791623 |
| enabled        | True                                |
| description     | AZ for Headquarters                 |
+-----+-----+
```

예 - az-dcn1 확인

```
$ openstack loadbalancer availabilityzone show az-dcn1
```

샘플 출력

```
+-----+-----+
| Field          | Value                                |
+-----+-----+
| name           | az-dcn1                             |
| availability_zone_profile_id | e0995a82-8e67-4cea-b32c-256cd61f9cf3 |
| enabled        | True                                |
| description     | AZ for South Region                 |
+-----+-----+
```

예 - az-dcn2 확인

```
$ openstack loadbalancer availabilityzone show az-dcn2
```

샘플 출력

```
+-----+-----+
| Field          | Value                               |
+-----+-----+
| name           | az-dcn2                             |
| availability_zone_profile_id | 306a4725-7dac-4046-8f16-f2e668ee5a8d |
| enabled        | True                                |
| description     | AZ for North Region                 |
+-----+-----+
```

다음 단계

- [가용성 영역에서 로드 밸런서 생성](#)

추가 리소스

- [명령줄 인터페이스 참조에서 LoadBalancer availabilityzoneprofile create](#)
- [명령줄 인터페이스 참조에서 LoadBalancer availabilityzone 생성](#)

8.3. 가용성 영역에서 로드 밸런서 생성

RHOSP(Red Hat OpenStack Platform) 로드 밸런싱 서비스(octavia)를 사용하면 분산 컴퓨팅 노드(DCN) 환경에서 가용성 영역(AZ)에 로드 밸런서를 생성하여 트래픽 처리량을 늘리고 대기 시간을 줄일 수 있습니다.

사전 요구 사항

- RHOSP 관리자가 제공하는 로드 밸런싱 서비스 AZ가 있어야 합니다.
- 로드 밸런서와 연결된 VIP 네트워크는 로드 밸런서가 멤버인 AZ에서 사용할 수 있어야 합니다.

프로세스

1. 자격 증명 파일을 가져옵니다.

예제

```
$ source ~/centralrc
```

2. DCN 환경의 로드 밸런서를 생성하려면 **--availability-zone** 옵션과 함께 **loadbalancer create** 명령을 사용하고 적절한 AZ를 지정합니다.

예제

예를 들어 가용성 영역(**az-central**)의 공용 서브넷(**public_subnet**)에 종료되지 않은 HTTPS 로드 밸런서(**lb1**)를 생성하려면 다음 명령을 입력합니다.

```
$ openstack loadbalancer create --name lb1 --vip-subnet-id \
public_subnet --availability-zone az-central
```

3. 리스너, 풀, 상태 모니터 및 로드 밸런서 멤버를 추가하여 로드 밸런서를 계속 생성합니다.
자세한 내용은 [로드 밸런싱을 서비스로 구성](#) 가이드를 참조하십시오.

검증

- 로드 밸런서(lb1)가 가용성 영역(**az-central**)의 멤버인지 확인합니다.

예제

```
$ openstack loadbalancer show lb1
```

샘플 출력

```
+-----+-----+
| Field          | Value                                |
+-----+-----+
| admin_state_up | True                                |
| availability_zone | az-central                          |
| created_at      | 2023-07-12T16:35:05                |
| description     |                                     |
| flavor_id       | None                                |
| id              | 85c7e567-a0a7-4fcb-af89-a0bbc9abe3aa |
| listeners       |                                     |
| name            | lb1                                 |
| operating_status | ONLINE                              |
| pools           |                                     |
| project_id      | d303d3bda9b34d73926dc46f4d0cb4bc   |
| provider        | amphora                             |
| provisioning_status | ACTIVE                              |
| updated_at      | 2023-07-12T16:36:45                |
| vip_address     | 10.101.10.229                       |
| vip_network_id  | d7f7de6c-0e84-49e2-9042-697fa85d2532 |
| vip_port_id     | 7f916764-d171-4317-9c86-a1750a54b16e |
| vip_qos_policy_id | None                                |
| vip_subnet_id   | a421cbcf-c5db-4323-b7ab-1df20ee6acab |
| tags            |                                     |
+-----+-----+
```

추가 리소스

- [로드 밸런싱 서비스를 위한 가용성 영역 생성](#)
- [명령줄 인터페이스 참조의 LoadBalancer](#) https://access.redhat.com/documentation/en-us/red_hat_openshift_platform/17.1/html/command_line_interface_reference/loadbalancer
- [로드 밸런싱을 서비스 가이드로 구성](#)

9장. DISTRIBUTEDCOMPUTEHCI 노드 교체

하드웨어 유지 관리 중에 에지 사이트에서 DistributedComputeHCI 노드를 축소, 확장 또는 교체해야 할 수 있습니다. DistributedComputeHCI 노드를 교체하려면 교체 중인 노드에서 서비스를 제거하고 노드 수를 축소한 다음 해당 노드를 확장하는 절차를 따릅니다.

9.1. RED HAT CEPH STORAGE 서비스 제거

클러스터에서 HCI(하이퍼 컨버지드) 노드를 제거하기 전에 Red Hat Ceph Storage 서비스를 제거해야 합니다. Red Hat Ceph 서비스를 제거하려면 제거 중인 노드의 클러스터 서비스에서 **ceph-osd** 서비스를 비활성화 및 제거한 다음 **mon,mgr,osd** 서비스를 중지 및 비활성화해야 합니다.

프로세스

1. 언더클라우드에서 SSH를 사용하여 삭제하려는 DistributedComputeHCI 노드에 연결합니다.

```
$ ssh tripleo-admin@<dcn-computeHCI-node>
```

2. cephadm 셸을 시작합니다. 제거 중인 호스트가 있는 사이트에 구성 파일 및 인증 키 파일을 사용합니다.

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring
```

3. 이후 단계에서 참조를 사용하기 위해 제거 중인 DistributedComputeHCI 노드와 연결된 OSD(오브젝트 스토리지 장치)를 기록합니다.

```
[ceph: root@dcn2-computeHCI2-1 ~]# ceph osd tree -c /etc/ceph/dcn2.conf
...
-3    0.24399    host dcn2-computeHCI2-1
  1    hdd 0.04880    osd.1          up 1.00000 1.00000
  7    hdd 0.04880    osd.7          up 1.00000 1.00000
 11    hdd 0.04880    osd.11         up 1.00000 1.00000
 15    hdd 0.04880    osd.15         up 1.00000 1.00000
 18    hdd 0.04880    osd.18         up 1.00000 1.00000
...
```

4. SSH를 사용하여 동일한 클러스터의 다른 노드에 연결하고 클러스터에서 모니터를 제거합니다.

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring

[ceph: root@dcn-computeHCI2-0]# ceph mon remove dcn2-computeHCI2-1 -c
/etc/ceph/dcn2.conf
removing mon.dcn2-computeHCI2-1 at [v2:172.23.3.153:3300/0,v1:172.23.3.153:6789/0],
there will be 2 monitors
```

5. SSH를 사용하여 클러스터에서 제거하려는 노드에 다시 로그인합니다.

6. **mgr** 서비스를 중지하고 비활성화합니다.

```
[tripleo-admin@dcn2-computeHCI2-1 ~]$ sudo systemctl --type=service | grep ceph
ceph-crash@dcn2-computeHCI2-1.service loaded active running Ceph crash dump
```

```

collector
ceph-mgr@dcn2-computehci2-1.service    loaded active    running    Ceph Manager

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl stop ceph-mgr@dcn2-computehci2-1

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl --type=service | grep ceph
ceph-crash@dcn2-computehci2-1.service loaded active running Ceph crash dump collector

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl disable ceph-mgr@dcn2-computehci2-1
Removed /etc/systemd/system/multi-user.target.wants/ceph-mgr@dcn2-computehci2-1.service.

```

7. cephadm 셸을 시작합니다.

```

$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring

```

8. 노드의 mgr 서비스가 클러스터에서 제거되었는지 확인합니다.

```

[ceph: root@dcn2-computehci2-1 ~]# ceph -s

cluster:
  id:      b9b53581-d590-41ac-8463-2f50aa985001
  health: HEALTH_WARN
          3 pools have too many placement groups
          mons are allowing insecure global_id reclaim

services:
  mon: 2 daemons, quorum dcn2-computehci2-2,dcn2-computehci2-0 (age 2h)
  mgr: dcn2-computehci2-2(active, since 20h), standbys: dcn2-computehci2-0 1
  osd: 15 osds: 15 up (since 3h), 15 in (since 3h)

data:
  pools: 3 pools, 384 pgs
  objects: 32 objects, 88 MiB
  usage: 16 GiB used, 734 GiB / 750 GiB avail
  pgs: 384 active+clean

```

1 mgr 서비스가 제거되면 mgr 서비스가 더 이상 나열되지 않습니다.

9. Red Hat Ceph Storage 사양을 내보냅니다.

```

[ceph: root@dcn2-computehci2-1 ~]# ceph orch ls --export > spec.yml

```

10. spec.yml 파일에서 사양을 편집합니다.

- spec.yml에서 호스트 <dcn-computehci-node>의 모든 인스턴스를 제거합니다.
- 다음에서 <dcn-computehci-node> 항목의 모든 인스턴스를 제거합니다.
 - service_type: osd

- service_type: mon

- service_type: host

11. Red Hat Ceph Storage 사양을 다시 적용합니다.

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch apply -i spec.yml
```

12. **ceph osd** 트리 를 사용하여 확인한 OSD를 제거합니다.

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch osd rm --zap 1 7 11 15 18
Scheduled OSD(s) for removal
```

13. 제거 중인 OSD의 상태를 확인합니다. 다음 명령에서 출력을 반환하지 않을 때까지 계속하지 마십시오.

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch osd rm status
OSD_ID HOST STATE PG_COUNT REPLACE FORCE
DRAIN_STARTED_AT
1 dcn2-computehci2-1 draining 27 False False 2021-04-23 21:35:51.215361
7 dcn2-computehci2-1 draining 8 False False 2021-04-23 21:35:49.111500
11 dcn2-computehci2-1 draining 14 False False 2021-04-23 21:35:50.243762
```

14. 제거 중인 호스트에 데몬이 남아 있지 않은지 확인합니다.

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch ps dcn2-computehci2-1
```

데몬이 여전히 있는 경우 다음 명령을 사용하여 해당 데몬을 제거할 수 있습니다.

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch host drain dcn2-computehci2-1
```

15. Red Hat Ceph Storage 클러스터에서 <dcn-computehci-node> 호스트를 제거합니다.

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch host rm dcn2-computehci2-1
Removed host 'dcn2-computehci2-1'
```

9.2. IMAGE 서비스(GLANCE) 서비스 제거

서비스에서 삭제할 때 노트에서 이미지 서비스를 제거합니다.

절차

- 이미지 서비스 서비스를 비활성화하려면 제거할 노트에서 **systemctl** 을 사용하여 해당 서비스를 비활성화합니다.

```
[root@dcn2-computehci2-1 ~]# systemctl stop tripleo_glance_api.service
[root@dcn2-computehci2-1 ~]# systemctl stop tripleo_glance_api_tls_proxy.service

[root@dcn2-computehci2-1 ~]# systemctl disable tripleo_glance_api.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_glance_api.service.
[root@dcn2-computehci2-1 ~]# systemctl disable tripleo_glance_api_tls_proxy.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_glance_api_tls_proxy.service.
```

9.3. BLOCK STORAGE(CINDER) 서비스 제거

서비스에서 삭제할 때 DistributedComputeHCI 노드에서 **cinder-volume** 및 **etcd** 서비스를 제거해야 합니다.

절차

1. 제거 중인 노드에서 **cinder-volume** 서비스를 식별하고 비활성화합니다.

```
(central) [stack@site-undercloud-0 ~]$ openstack volume service list --service cinder-volume
| cinder-volume | dcn2-computehci2-1@tripleo_ceph | az-dcn2 | enabled | up | 2022-03-23T17:41:43.000000 |
(central) [stack@site-undercloud-0 ~]$ openstack volume service set --disable dcn2-computehci2-1@tripleo_ceph cinder-volume
```

2. 스택의 다른 DistributedComputeHCI 노드에 로그인합니다.

```
$ ssh tripleo-admin@dcn2-computehci2-0
```

3. 제거 중인 노드와 연결된 **cinder-volume** 서비스를 삭제합니다.

```
[root@dcn2-computehci2-0 ~]# podman exec -it cinder_volume cinder-manage service
remove cinder-volume dcn2-computehci2-1@tripleo_ceph
Service cinder-volume on host dcn2-computehci2-1@tripleo_ceph removed.
```

4. 제거 중인 노드에서 **tripleo_cinder_volume** 서비스를 중지하고 비활성화합니다.

```
[root@dcn2-computehci2-1 ~]# systemctl stop tripleo_cinder_volume.service
[root@dcn2-computehci2-1 ~]# systemctl disable tripleo_cinder_volume.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_cinder_volume.service
```

9.4. DISTRIBUTEDCOMPUTEHCI 노드 삭제

프로비저닝된 매개변수를 **false** 값으로 설정하고 스택에서 노드를 제거합니다. **nova-compute** 서비스를 비활성화하고 관련 네트워크 에이전트를 삭제합니다.

프로세스

1. **baremetal-deployment.yaml** 파일을 복사합니다.

```
cp /home/stack/dcn2/overcloud-baremetal-deploy.yaml \
/home/stack/dcn2/baremetal-deployment-scaledown.yaml
```

2. **baremetal-deployment-scaledown.yaml** 파일을 편집합니다. 제거하려는 호스트를 식별하고 프로비저닝된 매개변수를 **false** 값을 갖도록 설정합니다.

```
instances:
...
- hostname: dcn2-computehci2-1
  provisioned: false
```

3. 스택에서 노드를 삭제합니다.

-

```
openstack overcloud node delete --stack dcn2 --baremetal-deployment
/home/stack/dcn2/baremetal_deployment_scaledown.yaml
```

4. 선택 사항: 노드를 재사용하려면 ironic을 사용하여 디스크를 정리합니다. 노드에서 Ceph OSD를 호스팅하는 경우 이 작업이 필요합니다.

```
openstack baremetal node manage $UUID
openstack baremetal node clean $UUID --clean-steps [{"interface":"deploy", "step":
"erase_devices_metadata"}]
openstack baremetal provide $UUID
```

5. 중앙 사이트를 다시 배포합니다. 초기 구성에 사용한 모든 템플릿을 포함합니다.

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/control-plane/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e /home/stack/central/dcn_ceph.yaml \
-e /home/stack/central/glance_update.yaml
```

9.5. 삭제된 DISTRIBUTEDCOMPUTEHCI 노드 교체

9.5.1. 삭제된 DistributedComputeHCI 노드 교체

DCN 배포에 새 HCI 노드를 추가하려면 추가 노드로 에지 스택을 재배포하고 해당 스택의 **ceph** 내보내기를 수행한 다음 중앙 위치에 대한 스택 업데이트를 수행해야 합니다. 중앙 위치의 스택 업데이트는 에지 사이트와 관련된 구성을 추가합니다.

사전 요구 사항

노드 수는 노드를 교체하거나 새 노드를 추가할 스택의 nodes_data.yaml 파일에서 적합합니다.

절차

1. 배포 스크립트에서 호출한 템플릿 중 하나에서 **EtcdInitialClusterState** 매개변수를 **existing** 로 설정해야 합니다.

```
parameter_defaults:
  EtcdInitialClusterState: existing
```

2. 스택 관련 배포 스크립트를 사용하여 재배포합니다.

```
(undercloud) [stack@site-undercloud-0 ~]$ ./overcloud_deploy_dcn2.sh
...
Overcloud Deployed without error
```

3. 스택에서 Red Hat Ceph Storage 데이터를 내보냅니다.

```
(undercloud) [stack@site-undercloud-0 ~]$ sudo -E openstack overcloud export ceph --stack
dcn1,dcn2 --config-download-dir /var/lib/mistral --output-file
~/central/dcn2_scale_up_ceph_external.yaml
```

4. 중앙 위치에 대한 배포 스크립트에서 dcn_ceph_external.yaml을 새로 생성된 dcn2_scale_up_ceph_external.yaml로 바꿉니다.
5. 중앙에서 스택 업데이트를 수행합니다.

```
(undercloud) [stack@site-undercloud-0 ~]$ ./overcloud_deploy.sh
...
Overcloud Deployed without error
```

9.6. 교체된 DISTRIBUTEDCOMPUTEHCI 노드의 기능 확인

1. **status** 필드의 값이 **활성화되어** 있고 **State** 필드의 값이 **up** 인지 확인합니다.

```
(central) [stack@site-undercloud-0 ~]$ openstack compute service list -c Binary -c Host -c
Zone -c Status -c State
```

Binary	Host	Zone	Status	State
...				
nova-compute	dcn1-compute1-0.redhat.local	az-dcn1	enabled	up
nova-compute	dcn1-compute1-1.redhat.local	az-dcn1	enabled	up
nova-compute	dcn2-computehci2-0.redhat.local	az-dcn2	enabled	up
nova-compute	dcn2-computehci2-0.redhat.local	az-dcn2	enabled	up
nova-compute	dcn2-computescaleout2-0.redhat.local	az-dcn2	enabled	up
nova-compute	dcn2-computehci2-2.redhat.local	az-dcn2	enabled	up
...				

2. 모든 네트워크 에이전트가 **up** 상태인지 확인합니다.

```
(central) [stack@site-undercloud-0 ~]$ openstack network agent list -c "Agent Type" -c Host -
c Alive -c State
```

Agent Type	Host	Alive	State
DHCP agent	dcn3-compute3-1.redhat.local	:-)	UP
Open vSwitch agent	central-computehci0-1.redhat.local	:-)	UP
DHCP agent	dcn3-compute3-0.redhat.local	:-)	UP
DHCP agent	central-controller0-2.redhat.local	:-)	UP
Open vSwitch agent	dcn3-compute3-1.redhat.local	:-)	UP
Open vSwitch agent	dcn1-compute1-1.redhat.local	:-)	UP
Open vSwitch agent	central-computehci0-0.redhat.local	:-)	UP
DHCP agent	central-controller0-1.redhat.local	:-)	UP
L3 agent	central-controller0-2.redhat.local	:-)	UP
Metadata agent	central-controller0-1.redhat.local	:-)	UP

```
| Open vSwitch agent | dcn2-computescaleout2-0.redhat.local | :- ) | UP |
| Open vSwitch agent | dcn2-computehci2-5.redhat.local | :- ) | UP |
| Open vSwitch agent | central-computehci0-2.redhat.local | :- ) | UP |
| DHCP agent | central-controller0-0.redhat.local | :- ) | UP |
| Open vSwitch agent | central-controller0-1.redhat.local | :- ) | UP |
| Open vSwitch agent | dcn2-computehci2-0.redhat.local | :- ) | UP |
| Open vSwitch agent | dcn1-compute1-0.redhat.local | :- ) | UP |
...
```

3. Ceph 클러스터의 상태를 확인합니다.

- a. SSH를 사용하여 새 DistributedComputeHCI 노드에 연결하고 Ceph 클러스터의 상태를 확인합니다.

```
[root@dcn2-computehci2-5 ~]# podman exec -it ceph-mon-dcn2-computehci2-5 \
ceph -s -c /etc/ceph/dcn2.conf
```

- b. 새 노드에 대한 ceph mon 및 ceph mgr 서비스가 모두 있는지 확인합니다.

```
services:
  mon: 3 daemons, quorum dcn2-computehci2-2,dcn2-computehci2-0,dcn2-
computehci2-5 (age 3d)
  mgr: dcn2-computehci2-2(active, since 3d), standbys: dcn2-computehci2-0, dcn2-
computehci2-5
  osd: 20 osds: 20 up (since 3d), 20 in (since 3d)
```

- c. 'ceph osd tree'를 사용하여 ceph osds의 상태를 확인합니다. 새 노드의 모든 osds가 STATUS에 있는지 확인합니다.

```
[root@dcn2-computehci2-5 ~]# podman exec -it ceph-mon-dcn2-computehci2-5 ceph
osd tree -c /etc/ceph/dcn2.conf
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1 0.97595 root default
-5 0.24399 host dcn2-computehci2-0
0 hdd 0.04880 osd.0 up 1.00000 1.00000
4 hdd 0.04880 osd.4 up 1.00000 1.00000
8 hdd 0.04880 osd.8 up 1.00000 1.00000
13 hdd 0.04880 osd.13 up 1.00000 1.00000
17 hdd 0.04880 osd.17 up 1.00000 1.00000
-9 0.24399 host dcn2-computehci2-2
3 hdd 0.04880 osd.3 up 1.00000 1.00000
5 hdd 0.04880 osd.5 up 1.00000 1.00000
10 hdd 0.04880 osd.10 up 1.00000 1.00000
14 hdd 0.04880 osd.14 up 1.00000 1.00000
19 hdd 0.04880 osd.19 up 1.00000 1.00000
-3 0.24399 host dcn2-computehci2-5
1 hdd 0.04880 osd.1 up 1.00000 1.00000
7 hdd 0.04880 osd.7 up 1.00000 1.00000
11 hdd 0.04880 osd.11 up 1.00000 1.00000
15 hdd 0.04880 osd.15 up 1.00000 1.00000
18 hdd 0.04880 osd.18 up 1.00000 1.00000
-7 0.24399 host dcn2-computehciscscaleout2-0
2 hdd 0.04880 osd.2 up 1.00000 1.00000
6 hdd 0.04880 osd.6 up 1.00000 1.00000
```

```

9 hdd 0.04880    osd.9          up 1.00000 1.00000
12 hdd 0.04880    osd.12         up 1.00000 1.00000
16 hdd 0.04880    osd.16         up 1.00000 1.00000

```

4. 새 DistributedComputeHCI 노드의 **cinder-volume** 서비스가 상태 'enabled'이고 상태 'up'인지 확인합니다.

```

(central) [stack@site-undercloud-0 ~]$ openstack volume service list --service cinder-volume
-c Binary -c Host -c Zone -c Status -c State
+-----+-----+-----+-----+-----+
| Binary | Host | Zone | Status | State |
+-----+-----+-----+-----+-----+
| cinder-volume | hostgroup@tripleo_ceph | az-central | enabled | up |
| cinder-volume | dcn1-compute1-1@tripleo_ceph | az-dcn1 | enabled | up |
| cinder-volume | dcn1-compute1-0@tripleo_ceph | az-dcn1 | enabled | up |
| cinder-volume | dcn2-computehci2-0@tripleo_ceph | az-dcn2 | enabled | up |
| cinder-volume | dcn2-computehci2-2@tripleo_ceph | az-dcn2 | enabled | up |
| cinder-volume | dcn2-computehci2-5@tripleo_ceph | az-dcn2 | enabled | up |
+-----+-----+-----+-----+-----+

```



참고

cinder-volume 서비스의 상태가 **다운** 된 경우 노드에서 서비스가 시작되지 않았습니다.

5. ssh를 사용하여 새 DistributedComputeHCI 노드에 연결하고 'systemctl'을 사용하여 Glance 서비스의 상태를 확인합니다.

```

[root@dcn2-computehci2-5 ~]# systemctl --type service | grep glance
tripleo_glance_api.service          loaded active running glance_api container
tripleo_glance_api_healthcheck.service loaded activating start start glance_api
healthcheck
tripleo_glance_api_tls_proxy.service loaded active running
glance_api_tls_proxy container

```

9.7. DISTRIBUTEDCOMPUTEHCI 상태 문제 해결

EtcdInitialClusterState 매개변수 값을 **existing** 로 설정하지 않고 대체 노드를 배포한 경우 **openstack volume service list** 를 실행할 때 교체된 노드의 **cinder-volume** 서비스가 **down** 으로 표시됩니다.

절차

1. 교체 노드에 로그인하고 etcd 서비스의 로그를 확인합니다. 로그에 **etcd** 서비스가 **/var/log/containers/stdouts/etcd.log** 로그 파일에 일치하지 않는 클러스터 ID를 보고하고 있는지 확인합니다.

```

2022-04-06T18:00:11.834104130+00:00 stderr F 2022-04-06 18:00:11.834045 E | rafthttp:
request cluster ID mismatch (got 654f4cf0e2cfb9fd want 918b459b36fe2c0c)

```

2. **EtcdInitialClusterState** 매개변수를 배포 템플릿의 기존 값으로 설정하고 배포 스크립트를 재실행합니다.
3. SSH를 사용하여 교체 노드에 연결하고 root로 다음 명령을 실행합니다.

```
[root@dcn2-computehci2-4 ~]# systemctl stop tripleo_etcd  
[root@dcn2-computehci2-4 ~]# rm -rf /var/lib/etcd/*  
[root@dcn2-computehci2-4 ~]# systemctl start tripleo_etcd
```

4. **/var/log/containers/stdouts/etcd.log** 로그 파일을 다시 확인하여 노드가 클러스터에 성공적으로 참여했는지 확인합니다.

```
2022-04-06T18:24:22.130059875+00:00 stderr F 2022-04-06 18:24:22.129395 I |  
etcdserver/membership: added member 96f61470cd1839e5 [https://dcn2-computehci2-  
4.internalapi.redhat.local:2380] to cluster 654f4cf0e2cfb9fd
```

5. cinder-volume 서비스의 상태를 확인하고 **openstack volume service list** 를 실행할 때 교체 노드에서 읽기되는지 확인합니다.

10장. 키 관리자를 사용하여 배포

Red Hat OpenStack Platform 16.1.2 릴리스 이전의 엣지 사이트를 배포한 경우 이 기능을 구현하려면 **roles.yaml**을 다시 생성해야 합니다. 기능을 구현하려면 **DCN** 사이트의 배포에 사용된 **roles.yaml** 파일을 다시 생성합니다.

```
$ openstack overcloud roles generate DistributedComputeHCI DistributedComputeHCIScaleOut -o
~/dcn0/roles_data.yaml
```

10.1. 키 관리자를 사용하여 엣지 사이트 배포

엣지 사이트에서 키 관리자(**barbican**) 서비스에 대한 액세스를 포함하려면 중앙 위치에서 **barbican**을 구성해야 합니다. **barbican** 설치 및 구성에 대한 자세한 내용은 [Barbican 배포](#)를 참조하십시오.

- **/usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml** 을 포함하여 **DCN** 사이트에서 **barbican**에 대한 액세스를 구성할 수 있습니다.

```
openstack overcloud deploy \
  --stack dcn0 \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/dcn0/roles_data.yaml \
  ....
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml
```

11장. NOVA에 GLANCE 이미지 사전 연결

로컬 임시 스토리지를 사용하도록 **OpenStack Compute**를 구성하면 **Glance** 이미지가 캐시되어 인스턴스 배포를 가속화합니다. 인스턴스에 필요한 이미지가 아직 캐시되지 않은 경우 인스턴스를 생성할 때 컴퓨팅 노드의 로컬 디스크로 다운로드됩니다.

Glance 이미지를 다운로드하는 프로세스는 이미지 크기 및 대역폭 및 대기 시간과 같은 네트워크 특성에 따라 가변적인 시간이 걸립니다.

인스턴스를 시작하려고 하면 로컬인 **Ceph** 클러스터의 에서 이미지를 사용할 수 없는 경우 다음 메시지와 함께 인스턴스를 시작할 수 없습니다.

```
Build of instance 3c04e982-c1d1-4364-b6bd-f876e399325b aborted: Image 20c5ff9d-5f54-4b74-830f-88e78b9999ed is unacceptable: No image locations are accessible
```

Compute 서비스 로그에 다음이 표시됩니다.

```
'Image %s is not on my ceph and [workarounds]/ never_download_image_if_on_rbd=True; refusing to fetch and upload.'
```

DCN 배포에 대해 기본적으로 **true** 로 설정된 **never_download_image_if_on_rbd** 라는 **nova.conf** 구성 파일의 매개 변수로 인해 인스턴스를 시작하지 못합니다. **dcn-storage.yaml** 파일에서 찾을 수 있는 **heat** 매개 변수 **NovaDisableImageDownloadToRbd** 를 사용하여 이 값을 제어할 수 있습니다.

오버클라우드를 배포하기 전에 **NovaDisableImageDownloadToRbd** 의 값을 **false** 로 설정하면 다음이 수행됩니다.

- **Compute** 서비스(**nova**)는 로컬에서 사용할 수 없는 경우 중앙 위치에서 사용 가능한 이미지를 자동으로 스트리밍합니다.
- **Glance** 이미지의 **COW** 복사본을 사용하지 않습니다.
- **Compute(nova)** 스토리지에는 이를 사용하는 인스턴스 수에 따라 동일한 이미지의 여러 사본이 포함될 수 있습니다.

- **WAN 링크와 nova 스토리지 풀** 을 모두 포화할 수 있습니다.

Red Hat은 이 값을 **true**로 설정하고 인스턴스를 시작하기 전에 필요한 이미지를 로컬에서 사용할 수 있도록 하는 것이 좋습니다. 예지에서 이미지를 사용할 수 있도록 하는 방법에 대한 자세한 내용은 [A.1.3절. “이미지를 새 사이트에 복사”](#) 을 참조하십시오.

로컬 이미지의 경우 **tripleo_nova_image_cache.yml ansible** 플레이북을 사용하여 향후 배포될 가능성이 있는 일반적으로 사용되는 이미지 또는 이미지를 사용하여 **VM** 생성 속도를 높일 수 있습니다.

11.1. TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE 플레이북 실행

사전 요구 사항

- 셸 환경에서 올바른 **API**에 대한 인증 자격 증명입니다.

각 단계에서 명령을 제공하기 전에 올바른 인증 파일을 제공했는지 확인해야 합니다.

프로세스

1. 오버클라우드 스택에 대한 **ansible** 인벤토리 디렉토리를 생성합니다.

```
$ mkdir inventories

$ find ~/overcloud-deploy/*/config-download \
  -name tripleo-ansible-inventory.yaml \
  while read f; do cp $f inventories/$(basename $(dirname $f)).yaml; done
```

2. 사전 캐시하려는 이미지 **ID** 목록을 생성합니다.

- a. 사용 가능한 이미지의 포괄적인 목록을 검색합니다.

```
$ source centralrc

$ openstack image list
+-----+
| ID              | Name      | Status |
+-----+-----+-----+
```

```
| 07bc2424-753b-4f65-9da5-5a99d8383fe6 | image_0 | active |
| d5187afa-c821-4f22-aa4b-4e76382bef86 | image_1 | active |
+-----+-----+-----+
```

b.

nova_cache_args.yml 이라는 **ansible** 플레이북 인수 파일을 생성하고 사전 캐시할 이미지의 ID를 추가합니다.

```
---
tripleo_nova_image_cache_images:
  - id: 07bc2424-753b-4f65-9da5-5a99d8383fe6
  - id: d5187afa-c821-4f22-aa4b-4e76382bef86
```

3.

tripleo_nova_image_cache.yml **ansible** 플레이북을 실행합니다.

```
$ source centralrc

$ ansible-playbook -i inventories \
  --extra-vars "@nova_cache_args.yml" \
  /usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

11.2. 성능 고려 사항

ansible forks 매개변수를 사용하여 동시에 다운로드하려는 이미지 수를 지정할 수 있습니다. 기본값은 5 입니다. **forks** 매개변수의 값을 늘려 이 이미지를 배포할 시간을 줄일 수 있지만 네트워크 증가 및 **glance-api** 로드와 균형을 조정해야 합니다.

다음과 같이 **--forks** 매개변수를 사용하여 동시성을 조정합니다.

```
ansible-playbook -i inventory.yaml \
  --forks 10 \
  --extra-vars "@nova_cache_args.yml" \
  /usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

11.3. DCN 사이트에 대한 이미지 배포 최적화

Glance 이미지 배포에 프록시를 사용하여 **WAN** 트래픽을 줄일 수 있습니다. 프록시를 구성하는 경우:

•

Glance 이미지는 프록시 역할을 하는 단일 컴퓨팅 노드에 다운로드됩니다.

-

프록시는 **glance** 이미지를 인벤토리의 다른 컴퓨팅 노드에 재배포합니다.

다음 매개변수를 **nova_cache_args.yml** ansible 인수 파일에 배치하여 프록시 노드를 구성할 수 있습니다.

tripleo_nova_image_cache_use_proxy 매개변수를 **true** 로 설정하여 이미지 캐시 프록시를 활성화합니다.

이미지 프록시는 보안 복사 **scp** 를 사용하여 인벤토리의 다른 노드에 이미지를 배포합니다. **SCP**는 **DCN** 사이트 간 **WAN**과 같이 대기 시간이 긴 네트워크보다 비효율적입니다. 플레이북 대상을 단일 **DCN** 위치로 제한하는 것이 좋습니다. 이 위치는 단일 스택과 관련이 있습니다.

tripleo_nova_image_cache_proxy_hostname 매개변수를 사용하여 이미지 캐시 프록시를 선택합니다. 기본 프록시는 **ansible** 인벤토리 파일의 첫 번째 **compute** 노드입니다.

tripleo_nova_image_cache_plan 매개변수를 사용하여 플레이북 인벤토리를 단일 사이트로 제한합니다.

```
tripleo_nova_image_cache_use_proxy: true
tripleo_nova_image_cache_proxy_hostname: dcn0-novacompute-1
tripleo_nova_image_cache_plan: dcn0
```

11.4. NOVA-CACHE 정리 구성

다음 두 조건이 모두 **true**인 경우 백그라운드 프로세스는 주기적으로 실행하여 **nova** 캐시에서 이미지를 제거합니다.

-

이미지는 인스턴스에서 사용하지 않습니다.

-

이미지 기간이 **nova** 매개변수 **remove_unused_original_minimum_age_seconds** 값보다 큼니다.

remove_unused_original_minimum_age_seconds 매개변수의 기본값은 **86400** 입니다. 값은 초 단위로 표시되며 24시간과 동일합니다. 초기 배포 중에 **NovalImageCacheTTL** **tripleo-heat-templates** 매개변수를 사용하거나 클라우드의 스택 업데이트 중에 이 값을 제어할 수 있습니다.

```
parameter_defaults:
  NovalImageCacheTTL: 604800 # Default to 7 days for all compute roles
```

Compute2Parameters:

NovalmImageCacheTTL: 1209600 # Override to 14 days for the Compute2 compute role

컴퓨팅 노드에 이미 존재하는 이미지를 미리 캐시하도록 플레이북에 지시하면 **ansible**에서 변경 사항을 보고하지 않지만 이미지의 사용 기간은 **0**으로 재설정됩니다. 이미지 캐시를 유지하려면 **NovalmImageCacheTTL** 매개변수 값보다 **ansible** 플레이를 더 자주 실행합니다.

12장. DCN용 TLS-E

분산 컴퓨팅 노드 인프라를 위해 설계된 클라우드에서 TLS(전송 계층 보안)를 활성화할 수 있습니다. 공용 액세스에만 TLS를 활성화하거나 TLS-e를 사용하여 모든 네트워크에서 TLS를 활성화하여 모든 내부 및 외부 데이터 흐름에서 암호화를 허용하는 옵션이 있습니다.

에지 사이트에 공용 엔드포인트가 없으므로 에지 스택에서 공용 액세스를 활성화할 수 없습니다. 공용 액세스를 위한 TLS에 대한 자세한 내용은 [Overcloud Public Endpoints](#)에서 [SSL/TLS 활성화](#)를 참조하십시오.

12.1. TLS-E를 사용하여 분산 컴퓨팅 노드 아키텍처 배포

사전 요구 사항

IdM(Red Hat Identity Manager)을 사용하여 RHOSP(Red Hat OpenStack Platform) 분산 컴퓨팅 노드 아키텍처에서 TLS를 구성하는 경우 Red Hat Identity Manager에 배포된 Red Hat Enterprise Linux 버전을 기반으로 다음 작업을 수행합니다.

Red Hat Enterprise Linux 8.4

1.

Red Hat Identity Management 노드에서 `ipa-ext.conf` 파일에서 신뢰할 수 있는 서브넷을 **ACL**에 허용했습니다.

```
acl "trusted_network" {
    localnets;
    localhost;
    192.168.24.0/24;
    192.168.25.0/24;
};
```

1.

`/etc/named/ipa-options-ext.conf` 파일에서 재귀 및 쿼리 캐시를 허용합니다.

```
allow-recursion { trusted_network; };
allow-query-cache { trusted_network; };
```

2.

'named-pkcs11' 서비스를 다시 시작하십시오.

```
systemctl restart named-pkcs11
```

Red Hat Enterprise Linux 8.2

RHEL(Red Hat Enterprise Linux) 8.2에 IdM(Red Hat Identity Manager)이 있는 경우 Red Hat Enterprise Linux를 업그레이드한 다음 RHEL 8.4에 대한 지침을 따라야 합니다.

Red Hat Enterprise Linux 7.x

RHEL(Red Hat Enterprise Linux) 7.x에 Red Hat IdM(Identity Manager)이 있는 경우 도메인 이름에 대한 액세스 제어 명령(ACI)을 수동으로 추가해야 합니다. 예를 들어 도메인 이름이 **redhat.local**인 경우 Red Hat Identity Manager에서 다음 명령을 실행하여 ACI를 구성합니다.

```
ADMIN_PASSWORD=redhat_01
DOMAIN_LEVEL_1=local
DOMAIN_LEVEL_2=redhat

cat << EOF | ldapmodify -x -D "cn=Directory Manager" -w ${ADMIN_PASSWORD}
dn: cn=dns,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}
changetype: modify
add: aci
aci: (targetattr = "aaaarecord || arecord || cnamerecord || idnsname || objectclass || ptrrecord")
(targetfilter = "(&(objectclass=idnsrecord)(|(aaaarecord=)(arecord=)(cnamerecord=)(ptrrecord=)
(idnsZoneActive=TRUE)))")(version 3.0; acl "Allow hosts to read DNS A/AAA/CNAME/PTR records";
allow (read,search,compare) userdn =
"ldap:///fqdn=*,cn=computers,cn=accounts,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}";)
EOF
```

중앙 위치와 엣지 위치 간의 디자인 차이로 인해 엣지 스택에 다음 파일을 포함하지 마십시오.

tls-everywhere-endpoints-dns.yaml

이 파일은 엣지 사이트에서 무시되며, 설정한 끝점은 중앙 스택에서 내보낸 엔드포인트에 의해 재정의됩니다.

haproxy-public-tls-certmonger.yaml

이 파일은 엣지에 공용 엔드포인트가 없기 때문에 배포에 실패합니다.

추가 정보:

- [Ansible을 사용하여 TLS-e 구현](#)

13장. 외부 액세스를 위한 CEPH 키 생성



주의

이 기능의 콘텐츠는 이 릴리스에서 *문서 프리뷰*로 사용할 수 있으므로 **Red Hat**에서 완전히 확인하지 않습니다. 테스트 용도로만 사용하고 프로덕션 환경에서 사용하지 마십시오.

Ceph 스토리지에 대한 외부 액세스는 로컬이 아닌 모든 사이트에서 **Ceph**에 액세스할 수 있습니다. 옛지의 **Ceph** 스토리지가 중앙 위치에서 외부인 것처럼 센트 위치에 있는 **Ceph** 스토리지는 에지(DCN) 사이트의 외부입니다.

Ceph 스토리지와 함께 중앙 또는 **DCN** 사이트를 배포할 때 로컬 및 외부 액세스 모두에 기본 **openstack** 인증 키를 사용할 수 있습니다. 또는 로컬이 아닌 사이트에서 액세스하기 위해 별도의 키를 생성할 수 있습니다.

외부 사이트에 대한 액세스에 추가 **Ceph** 키를 사용하려면 각 키의 이름이 동일해야 합니다. 키 이름은 다음 예제의 외부입니다.

로컬이 아닌 사이트에서 액세스하는 데 별도의 키를 사용하는 경우 로컬 액세스를 중단하지 않고 보안 이벤트에 대한 응답으로 외부 키를 취소 및 다시 발행할 수 있는 추가 보안 이점이 있습니다. 그러나 외부 액세스를 위해 별도의 키를 사용하면 교차 가용성 영역 백업 및 오프라인 볼륨 마이그레이션과 같은 일부 기능에 대한 액세스 권한이 손실됩니다. 보안 상태 요구 사항을 원하는 기능 세트와 균형을 조정해야 합니다.

기본적으로 중앙 및 모든 **DCN** 사이트의 키가 공유됩니다.

13.1. 외부 액세스를 위한 CEPH 키 생성

로컬이 아닌 액세스에 대한 외부 키를 생성하려면 다음 단계를 완료합니다.

프로세스

1.

외부 액세스를 위한 **Ceph** 키를 만듭니다. 이 키는 민감합니다. 다음을 사용하여 키를 생성할

수 있습니다.

```
python3 -c 'import os,struct,time,base64; key = os.urandom(16) ; \
header = struct.pack("<hiih", 1, int(time.time()), 0, len(key)) ; \
print(base64.b64encode(header + key).decode())'
```

2.

배포 중인 스택의 디렉터리에서 키에 대한 이전 명령의 출력을 사용하여 다음과 같은 내용이 포함된 **ceph_keys.yaml** 환경 파일을 생성합니다.

```
parameter_defaults:
  CephExtraKeys:
    - name: "client.external"
      caps:
        mgr: "allow *"
        mon: "profile rbd"
        osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
        key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
        mode: "0600"
```

3.

사이트 배포에 **ceph_keys.yaml** 환경 파일을 포함합니다. 예를 들어 **ceph_keys.yaml** 환경 파일을 사용하여 중앙 사이트를 배포하려면 다음과 같은 명령을 실행합니다.

```
overcloud deploy \
  --stack central \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  ....
  -e ~/central/ceph_keys.yaml
```

13.2. 외부 CEPH 키 사용

이미 배포된 키만 사용할 수 있습니다. 외부 키를 사용하여 사이트를 배포하는 방법에 대한 자세한 내용은 [13.1절. “외부 액세스를 위한 Ceph 키 생성”](#)을 참조하십시오. 이는 중앙 및 에지 사이트 모두에 대해 수행해야 합니다.

•

중앙에서 제공하는 외부 키를 사용할 에지 사이트를 배포하는 경우 다음을 완료합니다.

1.

에지 사이트에 대한 **dcn_ceph_external.yaml** 환경 파일을 생성합니다. 포함할 배포된 키를 지정하려면 **cephx-key-client-name** 옵션을 포함해야 합니다.

```
sudo -E openstack overcloud export ceph \
  --stack central \
  --cephx-key-client-name external \
  --output-file ~/dcn-common/dcn_ceph_external.yaml
```

2.

에지 사이트에서 중앙 사이트의 **Ceph** 클러스터에 액세스할 수 있도록 **dcn_ceph_external.yaml** 파일을 포함합니다. **ceph_keys.yaml** 파일을 포함하여 에지 사이트에서 **Ceph** 클러스터의 외부 키를 배포합니다.

•

에지 사이트를 배포한 후 중앙 위치를 업데이트할 때 중앙 위치가 **dcn** 외부 키를 사용해야 합니다.

1.

CephClientUserName 매개변수가 내보내는 키와 일치하는지 확인합니다. 다음 예에 표시된 대로 **external** 이름을 사용하는 경우 다음과 유사한 **glance_update.yaml**을 생성합니다.

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  CephClusterName: central
  GlanceBackendID: central
  GlanceMultistoreConfig:
    dcn0:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn0 rbd glance store'
      CephClientUserName: 'external'
      CephClusterName: dcn0
      GlanceBackendID: dcn0
    dcn1:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn1 rbd glance store'
      CephClientUserName: 'external'
      CephClusterName: dcn1
      GlanceBackendID: dcn1
```

2.

openstack overcloud export ceph 명령을 사용하여 중앙 위치에서 **DCN** 에지 액세스의 외부 키를 포함합니다. 이렇게 하려면 **--stack** 인수에 대해 쉼표로 구분된 스택 목록을 제공하고 **cephx-key-client-name** 옵션을 포함해야 합니다.

```
sudo -E openstack overcloud export ceph \
--stack dcn0,dcn1,dcn2 \
--cephx-key-client-name external \
--output-file ~/central/dcn_ceph_external.yaml
```

3.

원래 템플릿을 사용하여 중앙 사이트를 다시 배포하고 새로 생성된 **dcn_ceph_external.yaml** 및 **glance_update.yaml** 파일을 포함합니다.

```
openstack overcloud deploy \
--stack central \
```

```
--templates /usr/share/openstack-tripleo-heat-templates/ \  
-r ~/central/central_roles.yaml \  
...  
-e /usr/share/openstack-tripleo-heat-  
templates/environments/cephadm/cephadm.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \  
-e ~/central/central-images-env.yaml \  
-e ~/central/role-counts.yaml \  
-e ~/central/site-name.yaml  
-e ~/central/ceph.yaml \  
-e ~/central/ceph_keys.yaml \  
-e ~/central/glance.yaml \  
-e ~/central/dcn_ceph_external.yaml
```

부록 A. 배포 마이그레이션 옵션

이 섹션에는 **DCN** 스토리지의 관련 검증과 아키텍처 마이그레이션 또는 변경 사항이 포함되어 있습니다.

A.1. 엣지 스토리지 검증

Glance 다중 저장소 및 인스턴스 생성을 테스트하여 중앙 및 엣지 사이트의 배포가 작동하는지 확인합니다.

로컬 파일 시스템에서 사용할 수 있거나 웹 서버에서 사용할 수 있는 이미지를 **Glance**로 가져올 수 있습니다.



참고

중앙 위치에서 이미지를 사용하는 인스턴스가 없는 경우에도 중앙 사이트에 이미지 복사본을 항상 저장합니다.

사전 요구 사항

1.

glance stores-info 명령을 사용하여 이미지 서비스를 통해 사용할 수 있는 저장소를 확인합니다. 다음 예제에서는 **central**, **dcn1**, **dcn2**의 세 가지 저장소를 사용할 수 있습니다. 이는 각각 중앙 위치 및 엣지 사이트의 **Glance** 저장소에 해당합니다.

```
$ glance stores-info
+-----+
| Property | Value |
+-----+
| stores | [{"default": "true", "id": "central", "description": "central rbd glance |
| | store"}, {"id": "dcn0", "description": "dcn0 rbd glance store"}, |
| | {"id": "dcn1", "description": "dcn1 rbd glance store"}] |
+-----+
```

A.1.1. 로컬 파일에서 가져오기

이미지를 중앙 위치의 저장소에 먼저 업로드한 다음 이미지를 원격 사이트에 복사해야 합니다.

1.

이미지 파일이 **RAW** 형식인지 확인합니다. 이미지가 원시 형식이 아닌 경우 이미지를 이미지 서비스로 가져오기 전에 이미지를 변환해야 합니다.

```
file cirros-0.5.1-x86_64-disk.img
cirros-0.5.1-x86_64-disk.img: QEMU QCOW2 Image (v3), 117440512 bytes

qemu-img convert -f qcow2 -O raw cirros-0.5.1-x86_64-disk.img cirros-0.5.1-x86_64-disk.raw
```

Import the image into the default back end at the central site:

```
glance image-create \
--disk-format raw --container-format bare \
--name cirros --file cirros-0.5.1-x86_64-disk.raw \
--store central
```

A.1.2. 웹 서버에서 이미지 가져오기

이미지가 웹 서버에서 호스팅되는 경우 **GlanceImageImportPlugins** 매개변수를 사용하여 이미지를 여러 저장소에 업로드할 수 있습니다.

이 절차에서는 기본 이미지 변환 플러그인이 **Glance**에서 활성화되어 있다고 가정합니다. 이 기능은 **QCOW2** 파일 형식을 **Ceph RBD**에 가장 적합한 **RAW** 이미지로 자동 변환합니다. **glance image-show ID | grep disk_format** 을 실행하여 **Glance** 이미지가 **RAW** 형식인지 확인할 수 있습니다.

절차

1.

glance 명령의 **image-create-via-import** 매개 변수를 사용하여 웹 서버에서 이미지를 가져옵니다. **--stores** 매개변수를 사용합니다.

```
# glance image-create-via-import \
--disk-format qcow2 \
--container-format bare \
--name cirros \
--uri http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img \
--import-method web-download \
--stores central,dcn1
```

이 예에서는 **qcow2 cirros** 이미지가 공식 **Cirros** 사이트에서 다운로드되고 **Glance**에 따라 **RAW**로 변환되고 **--stores** 매개변수에서 지정하는 대로 중앙 사이트 및 에지 사이트 1로 가져옵니다.

또는 **--stores** 를 **--all-stores True** 로 교체하여 모든 저장소에 이미지를 업로드할 수 있습니다.

A.1.3. 이미지를 새 사이트에 복사

기존 이미지를 중앙 위치에서 에지 사이트로 복사할 수 있으므로 새로 설정된 위치에서 이전에 생성된 이미지에 액세스할 수 있습니다.

1.

복사 작업에 **glance** 이미지의 **UUID**를 사용합니다.

```
ID=$(openstack image show cirros -c id -f value)
```

```
glance image-import $ID --stores dcn0,dcn1 --import-method copy-image
```



참고

이 예에서 **--stores** 옵션은 **cirros** 이미지가 중앙 사이트에서 에지 사이트 **dcn1** 및 **dcn2**로 복사되도록 지정합니다. 또는 현재 이미지가 없는 모든 저장소에 이미지를 업로드하는 **--all-stores True** 옵션을 사용할 수 있습니다.

2.

이미지 사본이 각 저장소에 있는지 확인합니다. 속성 맵의 마지막 항목인 **stores** 키는 **central,dcn0,dcn1**로 설정됩니다.

```
$ openstack image show $ID | grep properties
| properties      | direct_url=rbid://d25504ce-459f-432d-b6fa-79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap, locations=[{u'url':
u'rbid://d25504ce-459f-432d-b6fa-79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata': {u'store': u'central'}}, {u'url': u'rbid://0c10d6b5-a455-4c4d-
bd53-8f2b9357c3c7/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata':
{u'store': u'dcn0'}}, {u'url': u'rbid://8649d6c3-dcb3-4aae-8c19-8c2fe5a853ac/images/8083c7e7-
32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata': {u'store': u'dcn1'}}],
os_glance_failed_import=', os_glance_importing_to_stores=', os_hash_algo='sha512,
os_hash_value=b795f047a1b10ba0b7c95b43b2a481a59289dc4cf2e49845e60b194a911819d
3ada03767bbba4143b44c93fd7f66c96c5a621e28dff51d1196dae64974ce240e,
os_hidden=False, stores=central,dcn0,dcn1 |
```



참고

해당 사이트에 사용하는 **VM**이 없는 경우에도 항상 중앙 사이트에 이미지 사본을 저장합니다.

A.1.4. 에지 사이트의 인스턴스가 이미지 기반 볼륨으로 부팅될 수 있는지 확인

에지 사이트의 이미지를 사용하여 영구 루트 볼륨을 생성할 수 있습니다.

절차

1.

볼륨으로 생성할 이미지의 ID를 식별하고 해당 ID를 **openstack volume create** 명령에 전달합니다.

```
IMG_ID=$(openstack image show cirros -c id -f value)
openstack volume create --size 8 --availability-zone dcn0 pet-volume-dcn0 --image $IMG_ID
```

2.

새로 생성된 볼륨의 볼륨 ID를 확인하고 **openstack server create** 명령에 전달합니다.

```
VOL_ID=$(openstack volume show -f value -c id pet-volume-dcn0)
openstack server create --flavor tiny --key-name dcn0-key --network dcn0-network --security-group basic --availability-zone dcn0 --volume $VOL_ID pet-server-dcn0
```

3.

dcn0 에지 사이트의 **ceph-mon** 컨테이너에서 **rbd** 명령을 실행하여 **volumes** 풀을 나열하여 볼륨이 이미지를 기반으로 하는지 확인할 수 있습니다.

```
$ sudo podman exec ceph-mon-$HOSTNAME rbd --cluster dcn0 -p volumes ls -l
NAME                               SIZE PARENT                               FMT PROT LOCK
volume-28c6fc32-047b-4306-ad2d-de2be02716b7 8 GiB images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap 2   excl
$
```

4.

인스턴스의 루트 볼륨의 **cinder** 스냅샷을 만들 수 있는지 확인합니다. 정리된 스냅샷을 생성하기 위해 서버가 데이터를 정지하도록 중지되었는지 확인합니다. 인스턴스가 꺼져 있을 때 볼륨 상태가 **in-use** 상태로 유지되므로 **--force** 옵션을 사용합니다.

```
openstack server stop pet-server-dcn0
openstack volume snapshot create pet-volume-dcn0-snap --volume $VOL_ID --force
openstack server start pet-server-dcn0
```

5.

dcn0 Ceph 클러스터에 **volumes** 풀의 콘텐츠를 나열하여 새로 생성된 스냅샷을 표시합니다.

```
$ sudo podman exec ceph-mon-$HOSTNAME rbd --cluster dcn0 -p volumes ls -l
NAME                               SIZE PARENT                               FMT PROT LOCK
volume-28c6fc32-047b-4306-ad2d-de2be02716b7 8 GiB
images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap 2   excl
volume-28c6fc32-047b-4306-ad2d-de2be02716b7@snapshot-a1ca8602-6819-45b4-a228-b4cd3e5adf60 8 GiB images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap 2 yes
```

A.1.5. 사이트 간에 이미지 스냅샷을 생성하고 복사할 수 있는지 확인

1.

dcn0 사이트에서 새 이미지를 만들 수 있는지 확인합니다. 데이터를 정지하여 정리된 스냅샷을 생성하기 위해 서버가 중지되었는지 확인합니다.

```
NOVA_ID=$(openstack server show pet-server-dcn0 -f value -c id)
openstack server stop $NOVA_ID
openstack server image create --name cirros-snapshot $NOVA_ID
openstack server start $NOVA_ID
```

2.

dcn0 에지 사이트의 이미지를 **glance**의 기본 백엔드인 **hub** 위치로 다시 복사합니다.

```
IMAGE_ID=$(openstack image show cirros-snapshot -f value -c id)
glance image-import $IMAGE_ID --stores central --import-method copy-image
```

Glance 다중 저장소 작업에 대한 자세한 내용은 [여러 저장소를 사용하는 이미지 서비스를 참조하십시오](#).

A.2. 스파인 및 리프 배포로 마이그레이션

기존 네트워크 구성이 있는 기존 클라우드를 스파인 리프 아키텍처가 있는 하나로 마이그레이션할 수 있습니다. 이를 위해 다음 조건이 필요합니다.

- 모든 베어 메탈 포트에는 실제 네트워크 속성 값이 **ctiplane** 로 설정되어 있어야 합니다.
- **enable_routed_networks** 매개변수가 추가되고 **undercloud.conf**에서 **true** 로 설정되고 **undercloud** 설치 명령인 **openstack undercloud install** 을 다시 실행합니다.

언더클라우드가 다시 배포되면 오버클라우드는 단일 리프 리프 0이 있는 스파인 리프 로 간주됩니다. 다음 단계를 통해 배포에 남아 있는 프로비저닝을 추가할 수 있습니다.

1. 언더클라우드의 라우팅된 스파인-리프트 구성에 표시된 대로 **undercloud.conf**에 원하는 서브넷을 추가합니다.
2. 언더클라우드 설치 명령 **openstack undercloud install** 을 다시 실행합니다.
3. 원하는 추가 네트워크 및 역할을 오버클라우드 템플릿 **network_data.yaml** 및 **roles_data.yaml** 에 각각 추가합니다.



참고

네트워크 구성 파일에서 `{{network.name}}InterfaceRoutes` 매개변수를 사용하는 경우 `NetworkDeploymentActions` 매개변수에 **UPDATE** 값이 포함되어 있는지 확인해야 합니다.

```
NetworkDeploymentActions: ['CREATE','UPDATE'])
```

4.

마지막으로 클라우드 배포와 관련된 모든 **heat** 템플릿이 포함된 오버클라우드 설치 스크립트를 다시 실행합니다.

A.3. 다중 스택 배포로 마이그레이션

기존 배포를 중앙 사이트로 처리하고 추가 에지 사이트를 추가하여 단일 스택 배포에서 다중 스택 배포로 마이그레이션할 수 있습니다.

기존 스택을 분할할 수 없습니다. 필요한 경우 기존 스택을 축소하여 컴퓨팅 노드를 제거할 수 있습니다. 그런 다음 이러한 컴퓨팅 노드를 에지 사이트에 추가할 수 있습니다.



참고

이 작업은 모든 컴퓨팅 노드가 제거되면 워크로드 중단을 생성합니다.

A.4. 에지 사이트 간 백업 및 복원

에지 사이트 및 가용성 영역의 **DCN(Distributed Compute Node)** 아키텍처에서 **Block Storage** 서비스(**cinder**) 볼륨을 백업하고 복원할 수 있습니다. **cinder-backup** 서비스는 중앙 가용성 영역(**AZ**)에서 실행되며 백업은 중앙 **AZ**에 저장됩니다. 블록 스토리지 서비스는 **DCN** 사이트에 백업을 저장하지 않습니다.

사전 요구 사항

- 선택적 **Block Storage** 백업 서비스를 배포합니다. 자세한 내용은 [블록 스토리지 볼륨 백업의 블록 스토리지 백업 서비스 배포](#)를 참조하십시오.
- **Block Storage(cinder) REST API** 마이크로 버전 3.51 이상

- 모든 사이트에서 공통 **openstack cephx** 클라이언트 이름을 사용해야 합니다. 자세한 내용은 **DCN(Distributed Compute Node) 아키텍처 배포**의 **외부 액세스용 Ceph 키 생성**을 참조하십시오.

프로세스

1.

첫 번째 **DCN** 사이트에 볼륨 백업을 생성합니다.

```
$ cinder --os-volume-api-version 3.51 backup-create --name <volume_backup> --availability-zone <az_central> <edge_volume>
```

- <volume_backup>**을 볼륨 백업의 이름으로 바꿉니다.
- <az_central>**을 **cinder-backup** 서비스를 호스팅하는 중앙 가용성 영역의 이름으로 바꿉니다.
- <edge_volume>**을 백업할 볼륨의 이름으로 바꿉니다.



참고

Ceph 인증 키에 문제가 있는 경우 호스트에서 컨테이너로 인증 키를 복사하도록 **cinder-backup** 컨테이너를 다시 시작해야 할 수 있습니다.

2.

두 번째 **DCN** 사이트의 새 볼륨으로 백업을 복원합니다.

```
$ cinder --os-volume-api-version 3.51 create --availability-zone <az_2> --name <new_volume> --backup-id <volume_backup> <volume_size>
```

- <az_2>**를 백업을 복원하려는 가용성 영역의 이름으로 바꿉니다.
- <new_volume>**을 새 볼륨의 이름으로 바꿉니다.
- <volume_backup>**을 이전 단계에서 생성한 볼륨 백업 이름으로 바꿉니다.

- <volume_size>를 원래 볼륨의 크기와 같거나 큰 값(GB)으로 바꿉니다.

A.5. DCN 환경 채택 및 준비

오버클라우드 채택을 위해 다음 작업을 수행해야 합니다.

- 각 사이트는 중앙 위치에서 시작하여 별도로 완전히 업그레이드됩니다.
- 중앙 위치 스택의 경우 오버클라우드에 네트워크 및 호스트 프로비저닝 구성 내보내기를 채택합니다. 'suggestion:-0+0'
- 새 컨테이너 및 추가 호환성 구성을 정의합니다.

채택 후 다음 작업을 수행하는 업그레이드 준비 스크립트를 실행해야 합니다.

- 오버클라우드 계획을 **OpenStack Platform 17.1**로 업데이트
- 업그레이드를 위해 노드를 준비합니다.

이 업그레이드 절차의 기간 및 영향에 대한 자세한 내용은 [업그레이드 기간 및 영향](#)을 참조하십시오.

사전 요구 사항

- 모든 노드는 **ACTIVE** 상태입니다.

```
$ openstack baremetal node list
```

MAINTENANCE 상태인 노드가 있는 경우 해당 노드를 **ACTIVE** 로 설정합니다.

```
$ openstack baremetal node maintenance unset <node_uuid>
```

- `<node_uuid>`를 노드의 UUID로 바꿉니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. 언더클라우드 업그레이드 중에 내보낸 다음 파일에 오버클라우드 업그레이드에 필요한 구성이 포함되어 있는지 확인합니다. `~/overcloud-deploy` 디렉터리에서 다음 파일을 찾을 수 있습니다.

- `tripleo-<stack>-passwords.yaml`
- `tripleo-<stack>-network-data.yaml`
- `tripleo-<stack>-virtual-ips.yaml`
- `tripleo-<stack>-baremetal-deployment.yaml`



참고

언더클라우드 업그레이드 후 파일이 생성되지 않은 경우 **Red Hat** 지원에 문의하십시오.



중요

다중 셀 환경이 있는 경우 파일을 각 셀 스택에 복사하는 예는 [다중 셀 환경에 대한 Overcloud 채택](#)을 검토합니다.

4. 기본 스택에서 **passwords.yaml** 파일을 `~/overcloud-deploy/<stack>` 디렉터리에 복사합니다. 환경의 각 스택에서 이 단계를 반복합니다.

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-passwords.yaml ~/overcloud-
deploy/<stack>/<stack>-passwords.yaml
```

•

<stack>을 스택 이름으로 바꿉니다.

5.

중앙 위치에서 준비 및 채택을 수행하는 경우 **network-data.yaml** 파일을 **stack** 사용자의 홈 디렉터리에 복사하고 네트워크를 배포합니다. 중앙 위치에 대해서만 이 작업을 수행합니다.

```
$ cp /home/stack/overcloud-deploy/central/tripleo-central-network-data.yaml ~/
$ mkdir /home/stack/overcloud_adopt
$ openstack overcloud network provision --debug \
--output /home/stack/overcloud_adopt/generated-networks-deployed.yaml tripleo-central-
network-data.yaml
```

자세한 내용은 **director**를 사용하여 [Red Hat OpenStack Platform 설치 및 관리에서 오버클라우드 프로비저닝 및 배포를 참조하십시오](#).

6.

중앙 위치에서 준비 및 채택을 수행하는 경우 **virtual-ips.yaml** 파일을 **stack** 사용자의 홈 디렉터리에 복사하고 네트워크 **VIP**를 프로비저닝합니다. 중앙 위치에 대해서만 이 작업을 수행합니다.

```
$ cp /home/stack/overcloud-deploy/central/tripleo-central-virtual-ips.yaml ~/
$ openstack overcloud network vip provision --debug \
--stack <stack> --output \
/home/stack/overcloud_adopt/generated-vip-deployed.yaml tripleo-central-virtual-ips.yaml
```

7.

기본 스택에서 **baremetal-deployment.yaml** 파일을 **stack** 사용자의 홈 디렉터리에 복사하고 오버클라우드 노드를 프로비저닝합니다. 환경의 각 스택에서 이 단계를 반복합니다.

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-baremetal-deployment.yaml ~/
$ openstack overcloud node provision --debug --stack <stack> \
--output /home/stack/overcloud_adopt/baremetal-central-deployment.yaml \
tripleo-<stack>-baremetal-deployment.yaml
```



참고

이는 오버클라우드 채택의 최종 단계입니다. 오버클라우드 채택을 완료하는 데 10분 이상 걸리면 **Red Hat** 지원에 문의하십시오.

8.

컨테이너를 준비하려면 다음 단계를 완료합니다.

a.

언더클라우드 업그레이드에 사용한 **containers-prepare-parameter.yaml** 파일을 백업합니다.

```
$ cp containers-prepare-parameter.yaml \
containers-prepare-parameter.yaml.orig
```

b.

스크립트를 실행하기 전에 **containers-prepare-parameter.yaml** 파일을 업데이트하기 전에 다음 환경 변수를 정의합니다.

- **NAMESPACE:** UBI9 이미지의 네임스페이스입니다. 예를 들어 **NAMESPACE="namespace":"example.redhat.com:5002",'**
 - **EL8_NAMESPACE:** UBI8 이미지의 네임스페이스입니다.
 - **NEUTRON_DRIVER:** 사용할 **OpenStack Networking(neutron)** 컨테이너를 사용하고 결정할 드라이버입니다. 원래 스택을 배포하는 데 사용한 컨테이너 유형으로 설정합니다. 예를 들어, **OVN** 기반 컨테이너를 사용하려면 **NEUTRON_DRIVER="neutron_driver":"ovn"** 으로 설정합니다.
 - **EL8_TAGS:** UBI8 이미지의 태그(예: **EL8_TAGS="tag":"17.1",'**
 - "17.1"을 콘텐츠 뷰에서 사용하는 태그로 바꿉니다.
 - **EL9_TAGS:** UBI9 이미지의 태그(예: **EL9_TAGS="tag":"17.1",'**
 - "17.1"을 콘텐츠 뷰에서 사용하는 태그로 바꿉니다.
- tag** 매개변수에 대한 자세한 내용은 *Red Hat OpenStack Platform* 배포 사용자 지정의 컨테이너 이미지 준비 매개변수를 참조하십시오.
- **CONTROL_PLANE_ROLES:** **--role** 옵션을 사용하는 컨트롤 플레인 역할 목록입니다(예: **--role ControllerOpenstack, --role Database, --role Messaging, --role**

Networker, --role CephStorage). 환경에서 컨트롤 플레인 역할 목록을 보려면 다음 명령을 실행합니다.

```
$ export STACK=<stack> \
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/${STACK}/tripleo-ansible-inventory.yaml \
| grep -vi compute
```

o

& lt;stack>을 스택 이름으로 바꿉니다.

•

COMPUTE_ROLES: --role 옵션을 사용하는 **Compute** 역할 목록입니다(예: --**Compute-1**). 해당 환경의 **Compute** 역할 목록을 보려면 다음 명령을 실행합니다.

```
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/${STACK}/tripleo-ansible-inventory.yaml \
| grep -i compute
```

•

CEPH_OVERRIDE: Red Hat Ceph Storage를 배포한 경우 **Red Hat Ceph Storage 5** 컨테이너 이미지를 지정합니다. 예를 들면 다음과 같습니다.

```
CEPH_OVERRIDE="ceph_image":"rhceph-5-rhel8","ceph_tag":"<latest>,"
```

o

& lt;latest>를 최신 **ceph_tag** 버전으로 바꿉니다(예: 5-499).

다음은 **containers-prepare-parameter.yaml** 파일 구성의 예입니다.

```
NAMESPACE="namespace":"registry.redhat.io/rhosp-rhel9",'
EL8_NAMESPACE="namespace":"registry.redhat.io/rhosp-rhel8",'
NEUTRON_DRIVER="neutron_driver":"ovn",'
EL8_TAGS="tag":"17.1",'
EL9_TAGS="tag":"17.1",'
CONTROL_PLANE_ROLES="--role Controller"
COMPUTE_ROLES="--role Compute"
CEPH_TAGS="ceph_tag":"5",'
```

c.

다음 스크립트를 실행하여 **containers-prepare-parameter.yaml** 파일을 업데이트합니다.



주의

Red Hat Ceph Storage를 배포하는 경우 다음 명령을 실행하기 전에 **CEPH_OVERRIDE** 환경 변수가 올바른 값으로 설정되어 있는지 확인합니다. 이렇게 하지 않으면 **Red Hat Ceph Storage**를 업그레이드할 때 문제가 발생합니다.

```
$ python3 /usr/share/openstack-tripleo-heat-templates/tools/multi-rhel-container-image-prepare.py \
    ${COMPUTE_ROLES} \
    ${CONTROL_PLANE_ROLES} \
    --enable-multi-rhel \
    --excludes collectd \
    --excludes nova-libvirt \
    --minor-override "
    ${EL8_TAGS}${EL8_NAMESPACE}${CEPH_OVERRIDE}${NEUTRON_DRIVER}"no_tag\":"not_used\" \
    --major-override "
    ${EL9_TAGS}${NAMESPACE}${CEPH_OVERRIDE}${NEUTRON_DRIVER}"no_tag\":"not_used\" \
    --output-env-file \
    /home/stack/containers-prepare-parameter.yaml
```

multi-rhel-container-image-prepare.py 스크립트는 다음 매개변수를 지원합니다.

--output-env-file

기본 **ContainerImagePrepare** 값이 포함된 환경 파일을 씁니다.

--local-push-destination

로컬 레지스트리에 업로드를 트리거합니다.

--enable-registry-login

시스템이 컨테이너를 가져오기 전에 원격 레지스트리에 로그인할 수 있는 플래그를 활성화합니다. **--local-push-destination** 을 사용하지 않고 대상 시스템에서 원격 레지스트리에 대한 네트워크 연결이 있는 경우 이 플래그를 사용합니다. 원격 레지스트리에 대한 네트워크 연결이 없을 수 있는 오버클라우드에는 이 플래그를 사용하지 마십시오.

--enable-multi-rhel

multi-rhel를 활성화합니다.

--excludes

제외할 서비스를 나열합니다.

--major-override

주요 릴리스에 대한 덮어쓰기 매개변수를 나열합니다.

--minor-override

마이너 릴리스에 대한 덮어쓰기 매개변수를 나열합니다.

--role

역할 목록입니다.

--role-file

role_data.yaml 파일

d.

Red Hat Ceph Storage를 배포하는 경우 **containers-prepare-parameter.yaml** 파일을 열어 **Red Hat Ceph Storage 5** 컨테이너 이미지가 지정되고 **Red Hat Ceph Storage 6** 컨테이너 이미지에 대한 참조가 없는지 확인합니다.

9.

director가 배포한 **Red Hat Ceph Storage** 배포가 있는 경우 **ceph_params.yaml** 이라는 파일을 생성하고 다음 콘텐츠를 포함합니다.

```
parameter_defaults:
  CephSpecFqdn: true
  CephConfigPath: "/etc/ceph"
  CephAnsibleRepo: "rhceph-5-tools-for-rhel-8-x86_64-rpms"
  DeployedCeph: true
```

중요

RHOSP 업그레이드가 완료된 후에는 **ceph_params.yaml** 파일을 제거하지 마십시오. 이 파일은 **director**가 배포한 **Red Hat Ceph Storage** 환경에 있어야 합니다. 또한 **openstack overcloud deploy** 를 실행할 때마다 **ceph_params.yaml** 파일을 포함해야 합니다(예: **-e ceph_params.yaml**).



참고

Red Hat Ceph Storage 배포에 짧은 이름이 포함된 경우 **CephSpecFqdn** 매개변수를 **false** 로 설정해야 합니다. **true** 로 설정하면 인벤토리에서 짧은 이름과 도메인 이름을 모두 사용하여 생성되므로 **Red Hat Ceph Storage** 업그레이드가 실패합니다.

10.

templates 디렉터리에 **upgrades-environment.yaml** 이라는 환경 파일을 생성하고 다음 콘텐츠를 포함합니다.

```
parameter_defaults:
  ExtraConfig:
    nova::workarounds::disable_compute_service_check_for_ffu: true
  DnsServers: ["<dns_servers>"]
  DockerInsecureRegistryAddress: <undercloud_FQDN>
  UpgradeInitCommand: |
    sudo subscription-manager repos --disable=*
    if $( grep -q 9.2 /etc/os-release )
    then
      sudo subscription-manager repos --enable=rhel-9-for-x86_64-baseos-eus-rpms --
enable=rhel-9-for-x86_64-appstream-eus-rpms --enable=rhel-9-for-x86_64-highavailability-
eus-rpms --enable=openstack-17.1-for-rhel-9-x86_64-rpms --enable=fast-datapath-for-rhel-9-
x86_64-rpms
      sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-
tools-for-rhel-9-x86_64-rpms
      sudo subscription-manager release --set=9.2
    else
      sudo subscription-manager repos --enable=rhel-8-for-x86_64-baseos-tus-rpms --
enable=rhel-8-for-x86_64-appstream-tus-rpms --enable=rhel-8-for-x86_64-highavailability-
tus-rpms --enable=openstack-17.1-for-rhel-8-x86_64-rpms --enable=fast-datapath-for-rhel-8-
x86_64-rpms
      sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-
tools-for-rhel-8-x86_64-rpms
      sudo subscription-manager release --set=8.4
    fi

    if $(sudo podman ps | grep -q ceph )
    then
      sudo dnf -y install cephadm
    fi
```

•

< **dns_servers** >를 DNS 서버 IP 주소의 쉼표로 구분된 목록으로 바꿉니다(예: ["10.0.0.36", "10.0.0.37"]).

•

< **undercloud_FQDN** >을 언더클라우드 호스트의 **FQDN**(정규화된 도메인 이름)으로 바꿉니다(예: "undercloud-0.ctlplane.redhat.local:8787").

환경 파일에서 구성할 수 있는 업그레이드 매개변수에 대한 자세한 내용은 [Upgrade 매개변수](#)를 참조하십시오.

11.

에지 위치에서 준비 및 채택을 수행하는 경우 **AuthCloudName** 매개변수를 중앙 위치의 이름으로 설정합니다.

```
parameter_defaults:
  AuthCloudName: central
```

12.

여러 이미지 서비스(**glance**) 저장소를 배포하는 경우 **copy-image**에 대한 이미지 서비스 API 정책을 설정하여 모든 규칙을 허용합니다.

```
parameter_defaults:
  GlanceApiPolicies: {glance-copy_image: {key 'copy-image', value: ""}}
```

13.

언더클라우드에서 **templates** 디렉터리에 **overcloud_upgrade_prepare.sh** 라는 파일을 생성합니다.

- 환경의 각 스택에 대해 이 파일을 생성해야 합니다. 이 파일에는 오버클라우드 배포 파일의 원본 내용과 사용자 환경과 관련된 환경 파일이 포함되어 있습니다.

- DCN 에지 위치에 대해 **overcloud_upgrade_prepare.sh** 를 생성하는 경우 다음 템플릿을 포함해야 합니다.

- 내보낸 중앙 사이트 매개 변수가 포함된 환경 템플릿입니다. 이 파일은 **/home/stack/overcloud-deploy/centra/central-export.yaml** 에서 찾을 수 있습니다.

- **generated-networks-deployed.yaml.yaml** - 중앙 위치에서 **openstack overcloud network provision** 명령을 실행하여 결과 파일입니다.

- **generated-vip-deployed.yaml.yaml** 중앙 위치에서 **openstack overcloud network vip provision** 명령을 실행한 결과 파일입니다. + 예를 들면 다음과 같습니다.

```
#!/bin/bash
openstack overcloud upgrade prepare --yes \
  --timeout 460 \
  --templates /usr/share/openstack-tripleo-heat-templates \
  --ntp-server 192.168.24.1 \
  --stack <stack> \
```

```

-r /home/stack/roles_data.yaml \
-e /home/stack/templates/internal.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-
ovs.yaml \
-e /home/stack/templates/network/network-environment.yaml \
-e /home/stack/templates/inject-trust-anchor.yaml \
-e /home/stack/templates/hostnames.yml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
-e /home/stack/templates/nodes_data.yaml \
-e /home/stack/templates/debug.yaml \
-e /home/stack/templates/firstboot.yaml \
-e /home/stack/templates/upgrades-environment.yaml \
-e /home/stack/overcloud-params.yaml \
-e /home/stack/overcloud-deploy/<stack>/overcloud-network-environment.yaml \
-e /home/stack/overcloud-adopt/<stack>-passwords.yaml \
-e /home/stack/overcloud_adopt/<stack>-baremetal-deployment.yaml \
-e /home/stack/overcloud_adopt/generated-networks-deployed.yaml \
-e /home/stack/overcloud_adopt/generated-vip-deployed.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-hw-machine-
type-upgrade.yaml \
-e /home/stack/skip_rhel_release.yaml \
-e ~/containers-prepare-parameter.yaml

```



참고

다중 셀 환경이 있는 경우 각 셀 스택에 대해 **overcloud_upgrade_prepare.sh** 파일을 생성하는 예는 다중 셀 환경에 대해 **Overcloud** 채택 을 검토합니다.

a.

원래 **network-environment.yaml** 파일 (/home/stack/templates/network/network-environment.yaml)에서 **OS::TripleO::*::Net::SoftwareConfig** 를 가리키는 모든 **resource_registry** 리소스를 제거합니다.

b.

overcloud_upgrade_prepare.sh 파일에서 환경과 관련된 다음 옵션을 포함합니다.

o

업그레이드별 매개변수(-e)가 있는 환경 파일(**upgrades-environment.yaml**)입니다.

o

새 컨테이너 이미지 위치(-e)가 포함된 환경 파일(**containers-prepare-parameter.yaml**)입니다. 대부분의 경우 언더클라우드에서 사용하는 환경 파일과 동일합니다.

- 환경 파일(-e)에 릴리스 매개 변수가 있는 **skip_rhel_release.yaml**.
- 배포와 관련된 모든 사용자 지정 구성 환경 파일(-e)입니다.
- 해당하는 경우 **--roles-file** 을 사용하여 사용자 지정 역할(**roles_data**) 파일을 사용합니다.
- **Ceph** 배포의 경우 환경 파일(**ceph_params.yaml**)은 **Ceph** 매개변수(-e)를 사용합니다.
- 오버클라우드 채택 중에 생성된 파일(**network-deployed.yaml**, **vip-deployed.yaml**, **baremetal-deployment.yaml**) (-e)
- 해당하는 경우 **IPA** 서비스(-e)가 포함된 환경 파일(**ipa-environment.yaml**)입니다.
- 구성 가능 네트워크를 사용하는 경우 **--network-file** 을 사용하여 (**network_data**) 파일입니다.



참고

오버클라우드 배포 파일 또는 **overcloud_upgrade_prepare.sh** 파일에 **network-isolation.yaml** 파일을 포함하지 마십시오. 네트워크 격리는 **network_data.yaml** 파일에 정의됩니다.

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



참고

환경의 모든 **RHEL 8** 컴퓨팅 노드가 **RHEL 9**로 업그레이드될 때까지 **nova-hw-machine-type-upgrade.yaml** 파일을 템플릿에 포함해야 합니다. 이 파일이 제외되면 **/var/log/containers/nova** 디렉터리의 **nova_compute.log** 에 오류가 표시됩니다. 모든 **RHEL 8** 컴퓨팅 노드를 **RHEL 9**로 업그레이드한 후 구성에서 이 파일을 제거하고 스택을 업데이트할 수 있습니다.

a.

director가 배포한 **Red Hat Ceph Storage** 사용 사례에서 업그레이드 중인 배포에서 **NFS**를 통해 **CephFS**를 사용하여 **Shared File Systems** 서비스(**manila**)를 활성화한 경우 **overcloud_upgrade_prepare.sh** 스크립트 파일 끝에 추가 환경 파일을 지정해야 합니다. 스크립트의 끝에 환경 파일을 추가해야 합니다. 이 파일은 스크립트 앞부분에서 지정된 다른 환경 파일을 재정의하기 때문입니다.

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-
ansible/manila-cephfs-ganeshasha-config.yaml
```

b.

외부 **Red Hat Ceph Storage** 사용 사례에서 업그레이드하는 배포에서 **CephFS**를 통해 **NFS**를 통해 **Shared File Systems** 서비스(**manila**)를 활성화한 경우 **overcloud_upgrade_prepare.sh** 스크립트의 연결된 환경 파일이 **tripleo** 기반 **ceph-nfs** 역할을 가리키는지 확인해야 합니다. 있는 경우 다음 환경 파일을 제거합니다.

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-
ansible/manila-cephfs-ganeshasha-config.yaml
```

다음 환경 파일을 추가합니다.

```
-e /usr/share/openstack-tripleo-heat-templates/environments/manila-
cephfs-ganeshasha-config.yaml
```

14.

환경의 각 스택에 대해 업그레이드 준비 스크립트를 실행합니다.

```
$ source stackrc
$ chmod 755 /home/stack/overcloud_upgrade_prepare.sh
$ sh /home/stack/overcloud_upgrade_prepare.sh
```



참고

다중 셀 환경이 있는 경우 각 셀 스택에 대해 생성한 각 **overcloud_upgrade_prepare.sh** 파일에 대해 스크립트를 실행해야 합니다. 예를 들어 **다중 셀 환경에 대한 Overcloud 채택**을 참조하십시오.

15.

업그레이드 준비가 완료될 때까지 기다립니다.

16.

컨테이너 이미지를 다운로드합니다.

```
$ openstack overcloud external-upgrade run --stack <stack> --tags  
container_image_prepare
```