Red Hat OpenStack Platform 17.1

컴퓨팅 셀을 사용하여 배포 확장

다중 셀 오버클라우드 생성 및 구성 가이드

Last Updated: 2024-11-22

Red Hat OpenStack Platform 17.1 컴퓨팅 셀을 사용하여 배포 확장

다중 셀 오버클라우드 생성 및 구성 가이드

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 가이드에서는 클라우드 관리자가 대규모 RHOSP(Red Hat OpenStack Platform) 배포에서 컴퓨팅 노드를 그룹화하도록 셀을 구성하고 관리하는 개념 및 절차를 제공합니다.

차례

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

Create Issue 양식을 사용하여 OpenShift (RHOSO) 또는 이전 Red Hat OpenStack Platform (RHOSP)의 Red Hat OpenStack Services 문서에 대한 피드백을 제공합니다. RHOSO 또는 RHOSP 문서에 대한 문제를 생성할 때 RHOSO Jira 프로젝트에 문제가 기록되어 피드백의 진행 상황을 추적할 수 있습니다.

문제 생성 양식을 완료하려면 Jira에 로그인해야 합니다. Red Hat Jira 계정이 없는 경우 https://issues.redhat.com 에서 계정을 생성할 수 있습니다.

1. 다음 링크를 클릭하여 **문제 생성** 페이지를 엽니다.
   https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12336920&summary=Documentation%20feedback:%20%3CAdd%20summary%20here%3E&i
   <Include+the+documentation+URL,+the%20chapter+or+section+number,+and+a+detailed+descrip

2. **요약** 및 **설명** 필드를 작성합니다. **설명** 필드에 문서 URL, 장 또는 섹션 번호, 문제에 대한 자세한 설명을 포함합니다. 양식의 다른 필드를 수정하지 마십시오.

3. **생성**을 클릭합니다.

1장. 다중 셀 오버클라우드 배포

셀을 사용하여 대규모 배포에서 컴퓨팅 노드를 각각 인스턴스 정보가 포함된 메시지 큐 및 전용 데이터베이스로 나눌 수 있습니다.

기본적으로 director는 모든 컴퓨팅 노드에 대해 단일 셀로 오버클라우드를 설치합니다. 이 셀에는 모든 Compute 서비스 및 데이터베이스와 모든 인스턴스 및 인스턴스 메타데이터가 포함됩니다. 대규모 배포의 경우 여러 셀이 있는 오버클라우드를 배포하여 더 많은 컴퓨팅 노드를 수용할 수 있습니다. 새 오버클라우드를 설치하거나 나중에 언제든지 셀을 추가할 수 있습니다.

다중 셀 배포에서 각 셀은 셀별 계산 서비스 및 데이터베이스의 독립형 복사본을 실행하고 해당 셀의 인스턴스에 대해서만 인스턴스 메타데이터를 저장합니다. 글로벌 정보 및 셀 매핑은 글로벌 컨트롤러 셀에 저장되며, 이는 셀 중 하나가 실패하는 경우 보안 및 복구를 제공합니다.

경고

기존 오버클라우드에 셀을 추가하는 경우 기본 셀의 컨덕터도 슈퍼 컨덕터의 역할을 수행합니다. 이는 배포 중인 셀과 오버클라우드의 성능에 부정적인 영향을 미칩니다. 또한 기본 셀을 오프라인 상태로 사용하는 경우 Super conductor도 오프라인 상태로 전환하여 전체 오버클라우드 배포를 중지합니다. 따라서 기존 오버클라우드를 확장하려면 기본 셀에 컴퓨팅 노드를 추가하지 마십시오. 대신 생성한 새 셀에 컴퓨팅 노드를 추가하여 기본 셀이 슈퍼 컨덕터 역할을 할 수 있습니다.

다중 셀 오버클라우드를 생성하려면 다음 작업을 수행해야 합니다.

1. 여러 셀을 처리하도록 오버클라우드를 구성하고 배포합니다.
2. 배포 중에 필요한 새 셀을 생성하고 프로비저닝합니다.
3. 각 셀에 컴퓨팅 노드를 추가합니다.
4. 각 Compute 셀을 가용성 영역에 추가합니다.

1.1. 사전 요구 사항

- 필요한 컨트롤러 노드 수와 함께 기본 오버클라우드를 배포했습니다.

1.2. 글로벌 구성 요소 및 서비스

다음 구성 요소는 컴퓨팅 셀 수에 관계없이 각 오버클라우드에 대해 컨트롤러 셀에 한 번 배포됩니다.

컴퓨팅 API

사용자에게 외부 REST API를 제공합니다.

컴퓨팅 스케줄러

인스턴스를 할당할 컴퓨팅 노드를 결정합니다.

배치 서비스

인스턴스에 Compute 리소스를 모니터링하고 할당합니다.

API 데이터베이스

Compute API 및 Compute 스케줄러 서비스에서 인스턴스에 대한 위치 정보를 추적하고 빌드되었지만 예약되지 않은 인스턴스에 임시 위치를 제공합니다.
다중 셀 배포에서 이 데이터베이스에는 각 셀에 대한 데이터베이스 연결을 지정하는 셀 매핑도 포함되어 있습니다.

cell0 데이터베이스

예약할 수 없는 인스턴스에 대한 정보의 전용 데이터베이스입니다.

슈퍼 컨덕터

이 서비스는 글로벌 서비스와 각 Compute 셀 사이를 조정하기 위해 다중 셀 배포에만 존재합니다. 이 서비스는 실패한 인스턴스 정보도 **cell0** 데이터베이스에 전송합니다.

1.3. 셀별 구성 요소 및 서비스

다음 구성 요소는 각 Compute 셀에 배포됩니다.

셀 데이터베이스

인스턴스에 대한 대부분의 정보를 포함합니다. 글로벌 API, 컨덕터, Compute 서비스에서 사용합니다.

컨덕터

글로벌 서비스에서 데이터베이스 쿼리 및 장기 실행 작업을 조정하고 컴퓨팅 노드를 직접 데이터베이스 액세스로부터 격리합니다.

메시지 큐

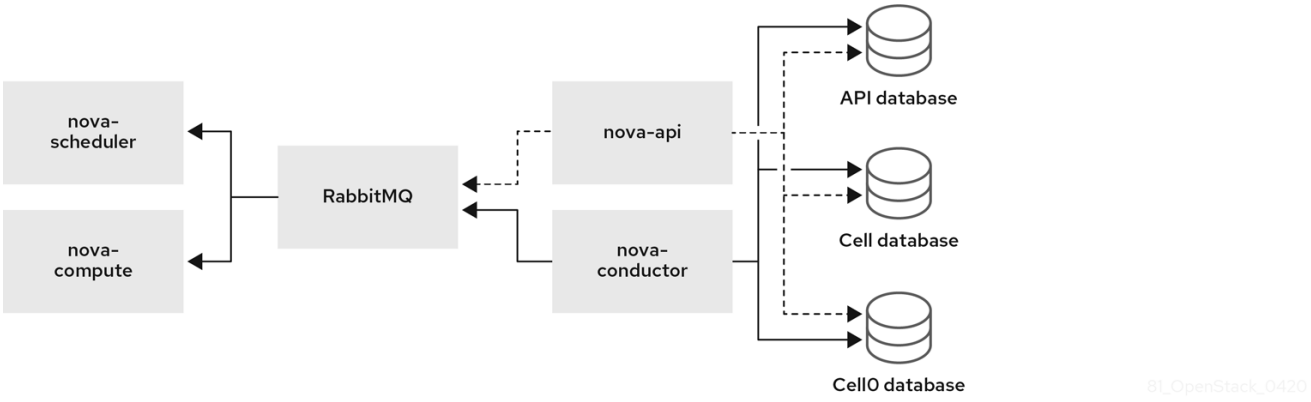
모든 서비스에서 셀 내에서 및 글로벌 서비스와 통신하기 위해 모든 서비스에서 사용하는 메시징 서비스입니다.

1.4. 셀 배포 아키텍처

director가 설치하는 기본 오버클라우드에는 모든 컴퓨팅 노드에 대한 단일 셀이 있습니다. 다음 아키텍처 다이어그램에 설명된 대로 셀을 추가하여 오버클라우드를 확장할 수 있습니다.

단일 셀 배포 아키텍처

다음 다이어그램은 기본 단일 셀 오버클라우드의 기본 구조 및 상호 작용의 예를 보여줍니다.

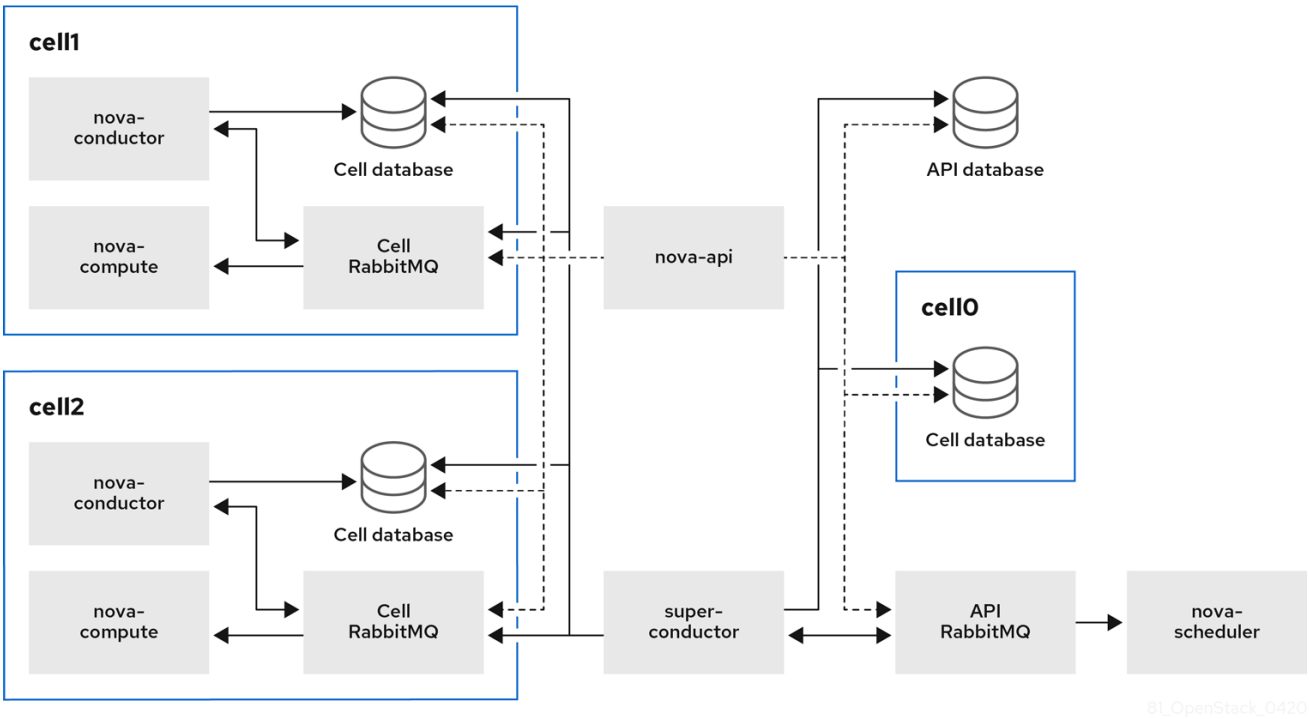


이 배포에서 모든 서비스는 단일 컨덕터를 사용하여 Compute API와 컴퓨팅 노드 간에 통신하고 단일 데이터베이스는 모든 라이브 인스턴스 데이터를 저장하도록 구성됩니다.

소규모 배포에서는 이 구성으로도 충분하지만 글로벌 API 서비스 또는 데이터베이스가 실패하면 전체 Compute 배포에서 고가용성 구성에 관계없이 정보를 보내거나 받을 수 없습니다.

다중 셀 배포 아키텍처

다음 다이어그램은 사용자 지정 다중 셀 오버클라우드의 기본 구조 및 상호 작용의 예를 보여줍니다.



이 배포에서 컴퓨팅 노드는 각각 자체 컨덕터, 데이터베이스 및 메시지 큐가 있는 여러 셀로 나뉩니다. 글로벌 서비스는 슈퍼 컨덕터를 사용하여 각 셀과 통신하며 글로벌 데이터베이스에는 전체 오버클라우드에 필요한 정보만 포함됩니다.

셀 수준 서비스는 글로벌 서비스에 직접 액세스할 수 없습니다. 이러한 격리는 셀 장애 발생 시 추가 보안 및 장애 시 안전하지 않은 기능을 제공합니다.

중요

"default"라는 첫 번째 셀에서 Compute 서비스를 실행하지 마십시오. 대신 컴퓨팅 노드를 포함하는 각 새 셀을 별도로 배포합니다.

1.5. 다중 셀 배포 고려 사항

다중 셀 배포의 최대 컴퓨팅 노드 수

모든 셀에서 최대 컴퓨팅 노드 수는 500개입니다.

셀 간 인스턴스 마이그레이션

한 셀의 호스트에서 다른 셀의 호스트로 인스턴스를 마이그레이션하는 것은 지원되지 않습니다. 이 제한은 다음 작업에 영향을 미칩니다.

- 콜드 마이그레이션
- 실시간 마이그레이션
- unshelve
- 크기 조정
- 비우기

서비스 할당량

계산 서비스 할당량은 데이터베이스의 정적이 아니라 각 리소스 사용량 지점에서 동적으로 계산됩니다. 다중 셀 배포에서 연결할 수 없는 셀은 사용 정보를 실시간으로 제공할 수 없으므로 셀에 다시 연결할 수 있을 때 할당량을 초과할 수 있습니다.
배치 서비스 및 API 데이터베이스를 사용하여 실패하거나 연결할 수 없는 셀을 견딜 수 있도록 할당량 계산을 구성할 수 있습니다.

API 데이터베이스

Compute API 데이터베이스는 모든 셀에 대해 항상 전역적이며 각 셀에 대해 복제할 수 없습니다.

콘솔 프록시

콘솔 토큰 권한 부여는 셀 데이터베이스에 저장되므로 각 셀에 대한 콘솔 프록시를 구성해야 합니다. 각 콘솔 프록시 서버는 해당 셀 데이터베이스의 **database.connection** 정보에 액세스해야 합니다.

컴퓨팅 메타데이터 API

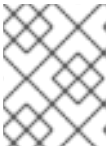
여러 셀 환경의 모든 셀에 동일한 네트워크를 사용하는 경우 셀 간에 브리지할 수 있도록 Compute 메타데이터 API를 전역적으로 실행해야 합니다. Compute 메타데이터 API가 전역적으로 실행되면 **api_database.connection** 정보에 액세스해야 합니다.
라우팅된 네트워크를 사용하여 여러 셀 환경을 배포하는 경우 성능 및 데이터 격리를 개선하기 위해 각 셀에서 Compute 메타데이터 API를 별도로 실행해야 합니다. Compute 메타데이터 API가 각 셀에서 실행되는 경우 **neutron-metadata-agent** 서비스는 해당 **nova-api-metadata** 서비스를 가리켜야 합니다.

NovaLocalMetadataPerCell 매개변수를 사용하여 Compute 메타데이터 API가 실행되는 위치를 제어합니다.

2장. 동일한 네트워크를 사용하여 다중 셀 환경 구성 및 배포

동일한 네트워크를 사용하여 여러 셀을 처리하도록 RHOSP(Red Hat OpenStack Platform) 배포를 구성하려면 다음 작업을 수행해야 합니다.

1. 오버클라우드 스택의 컨트롤 플레인에서 매개변수 정보를 추출합니다.
2. 셀 역할 파일을 생성합니다. 셀의 컴퓨팅 노드에 기본 **Compute** 역할을 사용하고 셀 컨트롤러 노드에 대해 전용 **CellController** 역할을 사용할 수 있습니다. 각 셀 스택의 사용자 지정 역할과 같은 다중 셀 환경에서 사용할 사용자 지정 역할을 생성할 수도 있습니다. 사용자 지정 역할 생성에 대한 자세한 내용은 Composable 서비스 및 사용자 지정 역할을 참조하십시오.
3. **CellController** 역할에 대한 호스트를 지정합니다.



참고

다중 셀 환경에 대한 사용자 지정 역할을 생성한 경우 사용자 지정 역할에 대한 호스트도 지정해야 합니다.

4. 각 셀을 구성합니다.
5. 각 셀 스택을 배포합니다.

2.1. 오버클라우드 스택 컨트롤 플레인에서 매개변수 정보 추출

기본 오버클라우드 스택에서 **default** 라는 첫 번째 셀에서 매개변수 정보를 추출합니다.

절차

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

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

3. 오버클라우드 스택의 **기본** 셀에서 다중 셀 배포를 위한 새로운 공통 환경 파일로 셀 구성 및 암호 정보를 내보냅니다.

```
(undercloud)$ openstack overcloud cell export --control-plane-stack overcloud \
 -f --output-file common/default_cell_export.yaml \
 --working-dir /home/stack/overcloud-deploy/overcloud/
```

이 명령은 **EndpointMap**,**HostsEntry**,**AllNodesConfig**,**GlobalConfig** 매개변수 및 암호 정보를 공통 환경 파일로 내보냅니다.

작은 정보

환경 파일이 이미 있는 경우 **--force-overwrite** 또는 **-f** 옵션을 사용하여 명령을 입력합니다.

2.2. 셀 역할 파일 생성

스택에서 동일한 네트워크를 사용하고 사용자 지정 역할이 필요하지 않은 경우 모든 셀 스택에서 사용할 공통 셀 역할 파일을 생성할 수 있습니다.

절차

- **Compute** 및 **CellController** 역할을 포함하는 **cell_roles_data.yaml** 이라는 새 역할 데이터 파일을 생성합니다.

```
(undercloud)$ openstack overcloud roles generate \
 --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
 -o common/cell_roles_data.yaml Compute CellController
```

2.3. CELLCONTROLLER 역할에 대한 호스트 지정

CellController 역할에 베어 메탈 노드를 지정하려면 **CellController** 역할의 노드를 태그하도록 리소스 클래스로 베어 메탈 노드를 구성해야 합니다.

작은 정보

여러 셀 환경에 대한 사용자 지정 역할을 생성한 경우 이 절차에 따라 사용자 지정 역할의 이름으로 셀 컨트롤러 이름을 대체하여 사용자 지정 역할의 리소스 클래스를 구성할 수 있습니다.

참고

다음 절차는 아직 프로비저닝되지 않은 새 오버클라우드 노드에 적용됩니다. 이미 프로비저닝된 기존 오버클라우드 노드에 리소스 클래스를 할당하려면 오버클라우드를 축소하여 노드를 프로비저닝 해제한 다음 오버클라우드를 확장하여 새 리소스 클래스 할당으로 노드를 다시 프로비저닝합니다. 자세한 내용은 오버클라우드 노드 스케일링을 참조하십시오.

프로세스

1. 노드 정의 템플릿에 **node.json** 또는 **node.yaml** 을 추가하여 **cellController** 역할의 베어 메탈 노드를 등록합니다. 자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드용 노드 등록을 참조하십시오.

2. 노드 하드웨어를 검사합니다.

```
(undercloud)$ openstack overcloud node introspect \
 --all-manageable --provide
```

   자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 베어 메탈 노드 하드웨어의 인벤토리 생성* 을 참조하십시오.

3. 노드 목록을 검색하여 UUID를 확인합니다.

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

4. 사용자 지정 셀 컨트롤러 리소스 클래스를 사용하여 셀 컨트롤러로 지정할 각 베어 메탈 노드에 태그를 지정합니다.

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.CELL-CONTROLLER <node>
```

   - & **lt;node** >를 베어 메탈 노드의 이름 또는 UUID로 바꿉니다.

5. 노드 정의 파일 **overcloud-baremetal-deploy.yaml** 에 **CellController** 역할을 추가하고 노드에 할당할 예측 노드 배치, 리소스 클래스, 네트워크 토폴로지 또는 기타 속성을 정의합니다.

```
- name: Controller
  count: 3
- name: Compute
  count: 3
  defaults:
    network_config:
      template: /home/stack/templates/nic-config/<cell_topology_file>
  instances:
  - hostname: cell1-compute-%index%
    name: computecell1
  - hostname: cell1-compute-%index%
    name: computecell2
  - hostname: cell1-compute-%index%
    name: computecell3
- name: CellController
  count: 1
  defaults:
    resource_class: baremetal.CELL-CONTROLLER
    network_config:
      template: /home/stack/templates/nic-config/<role_topology_file>
  instances:
  - hostname: cell1-cellcontroller-%index%
    name: cellcontroller
```

- & **lt;cell_topology_file** >을 셀 스택에 사용할 네트워크 토폴로지 파일의 이름으로 바꿉니다 (예: **compute.j2**).

- < **role_topology_file** >을 **CellController** 역할에 사용할 네트워크 토폴로지 파일의 이름으로 바꿉니다(예: **cell_controller_net_top.j2**).
  기존 네트워크 토폴로지를 재사용하거나 역할 또는 셀에 대한 새 사용자 지정 네트워크 인터페이스 템플릿을 생성할 수 있습니다. 자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리의* 사용자 지정 네트워크 인터페이스 템플릿을 참조하십시오. 기본 네트워크 정의 설정을 사용하려면 역할 정의에 **network_config** 를 포함하지 마십시오.

노드 정의 파일에서 노드 속성을 구성하는 데 사용할 수 있는 속성에 대한 자세한 내용은 베어 메탈 노드 프로비저닝 속성을 참조하십시오. 노드 정의 파일의 예는 노드 정의 파일 예제 를 참조하십시오.

6. 역할의 새 노드를 프로비저닝합니다.

```
(undercloud)$ openstack overcloud node provision \
[--stack <stack>] \
[--network-config \]
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

- 선택 사항: **<stack** >을 베어 메탈 노드가 프로비저닝되는 스택의 이름으로 바꿉니다. 기본값은 **overcloud** 입니다.

- 선택 사항: **--network-config** 선택적 인수를 포함하여 네트워크 정의를 **cli-overcloud-node-network-config.yaml** Ansible 플레이북에 제공합니다. **network_config** 속성을 사용하여 노드 정의 파일에 네트워크 정의를 정의하지 않은 경우 기본 네트워크 정의가 사용됩니다.

- < **deployment_file** >을 배포 명령에 포함할 heat 환경 파일의 이름으로 교체합니다(예: **/home/stack/templates/overcloud-baremetal-deployed.yaml**).

7. 별도의 터미널에서 프로비저닝 진행 상황을 모니터링합니다. 프로비저닝이 성공하면 노드 상태가 **available** 에서 **active** 로 변경됩니다.

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

8. **--network-config** 옵션 없이 provisioning 명령을 실행한 경우 NIC 템플릿 파일을 가리키도록 **network-environment.yaml** 파일에서 < **Role>NetworkConfigTemplate** 매개변수를 구성합니다.

```
parameter_defaults:
  ComputeNetworkConfigTemplate: /home/stack/templates/nic-configs/compute.j2
  CellControllerNetworkConfigTemplate: /home/stack/templates/nic-
configs/<role_topology_file>
  ControllerNetworkConfigTemplate: /home/stack/templates/nic-configs/controller.j2
```

- < **role_topology_file** >을 **CellController** 역할의 네트워크 토폴로지를 포함하는 파일 이름으로 바꿉니다(예: **cell_controller_net_top.j2**). 기본 네트워크 토폴로지를 사용하려면 **compute.j2** 로 설정합니다.

2.4. 동일한 네트워크를 사용하여 각 셀 스택 구성 및 배포

각 셀 스택을 배포에서 추가 셀로 식별하도록 구성해야 합니다.

프로세스

1. 새 셀에 대한 새 디렉토리를 만듭니다.

```
(undercloud)$ mkdir cells
```

2. 셀 디렉터리의 각 추가 셀에 대한 새 환경 파일을 생성합니다(예: **/ cells /cell1.yaml**).

3. 각 환경 파일에 다음 매개변수를 추가하고 배포의 각 셀에 대한 매개변수 값을 업데이트합니다.

```
parameter_defaults:
  # Disable network creation in order to use the `network_data.yaml` file from the overcloud
stack,
  # and create ports for the nodes in the separate stacks on the existing networks.
  ManageNetworks: false

  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # The DNS names for the VIPs for the cell
  CloudName: cell1.ooo.test
  CloudNameInternal: cell1.internalapi.ooo.test
  CloudNameStorage: cell1.storage.ooo.test
  CloudNameStorageManagement: cell1.storagemgmt.ooo.test
  CloudNameCtlplane: cell1.ctlplane.ooo.test
```

4. 다른 환경 파일과 함께 스택에 환경 파일을 추가하고 셀 스택을 배포합니다.

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack cell1 \
 -e [your environment files] \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml \
 -r $HOME/common/cell_roles_data.yaml \
 -e $HOME/common/default_cell_export.yaml \
 -e $HOME/cells/cell1.yaml
```

모든 셀 스택이 배포될 때까지 각 셀 스택에 대해 이 단계를 반복합니다.

2.5. 다음 단계

- Compute 서비스 내에서 셀 생성 및 관리

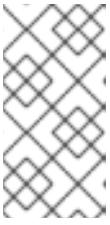
3장. 라우팅된 네트워크를 사용하여 다중 셀 환경 구성 및 배포

중요

이 섹션의 콘텐츠는 이 릴리스에서 *기술 프리뷰*로 제공되므로 Red Hat에서 완전히 지원되지 않습니다. 테스트 용도로만 사용해야 하며 프로덕션 환경에 배포해서는 안 됩니다. 자세한 내용은 기술 프리뷰 를 참조하십시오.

라우팅된 네트워크로 여러 셀을 처리하도록 RHOSP(Red Hat OpenStack) 배포를 구성하려면 다음 작업을 수행해야 합니다.

1. 오버클라우드 스택에서 셀 네트워크 라우팅을 위해 컨트롤 플레인을 준비합니다.
2. 오버클라우드 스택의 컨트롤 플레인에서 매개변수 정보를 추출합니다.
3. 셀 스택에서 셀 네트워크 라우팅을 구성합니다.
4. 각 스택에 대한 셀 역할 파일을 생성합니다. 기본 **Compute** 역할을 셀의 컴퓨팅 노드의 기반으로 사용하고 전용 **CellController** 역할을 셀 컨트롤러 노드의 기반으로 사용할 수 있습니다. 다중 셀 환경에서 사용할 사용자 지정 역할을 생성할 수도 있습니다. 사용자 지정 역할 생성에 대한 자세한 내용은 Composable 서비스 및 사용자 지정 역할을 참조하십시오.
5. 생성한 각 사용자 지정 역할에 대한 호스트를 지정합니다.



참고

이 절차는 단일 컨트롤 플레인 네트워크가 있는 환경에 대한 것입니다. 환경에 스파인 리프 네트워크 환경과 같은 여러 컨트롤 플레인 네트워크가 있는 경우 노드를 각 리프에 태그할 수 있도록 각 리프 네트워크의 각 역할에 대한 호스트도 지정해야 합니다. 자세한 내용은 리프 노드에 대한 역할 지정을 참조하십시오.

6. 각 셀을 구성합니다.
7. 각 셀 스택을 배포합니다.

3.1. 사전 요구 사항

- 라우팅된 네트워크에 대해 언더클라우드를 설정했습니다. 자세한 내용은 언더클라우드의 라우팅된 스파인-리프트 구성을 참조하십시오.

3.2. 셀 네트워크 라우팅을 위한 컨트롤 플레인 및 기본 셀 준비

오버클라우드 스택이 셀과 통신하도록 오버클라우드 스택에서 경로를 구성해야 합니다. 이를 위해 기본 스택의 모든 네트워크와 서브넷을 정의하는 네트워크 데이터 파일을 생성하고 이 파일을 사용하여 오버클라우드 스택과 셀 스택을 모두 배포합니다.

절차

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

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

3. 공통 스택 구성에 사용할 새 디렉터리를 생성합니다.

```
(undercloud)$ mkdir common
```

4. 기본 **network_data_subnets_routed.yaml** 파일을 **공통** 디렉터리에 복사하여 오버클라우드 스택의 구성 가능 네트워크를 추가합니다.

```
(undercloud)$ cp /usr/share/openstack-tripleo-heat-
templates/network_data_subnets_routed.yaml
~/common/network_data_routed_multi_cell.yaml
```

구성 가능 네트워크에 대한 자세한 내용은 *Director 설치 및 사용* 가이드의 Composable 네트워크를 참조하십시오.

5. 네트워크의 **/common/network_data_routed_multi_cell.yaml** 에서 구성을 업데이트하고, 쉽게 식별할 수 있도록 셀 서브넷 이름을 업데이트합니다(예: **internal_api_leaf1** 을 **internal_api_cell1**).

6. 각 역할에 대한 NIC 템플릿의 인터페이스에 < **network_name>InterfaceRoutes** 가 포함되어 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
-
  type: vlan
  vlan_id:
    get_param: InternalApiNetworkVlanID
  addresses:
  -
    ip_netmask:
      get_param: InternalApiIpSubnet
  routes:
    get_param: InternalApiInterfaceRoutes
```

7. **network_data_routed_multi_cell.yaml** 파일을 다른 환경 파일과 함께 오버클라우드 스택에 추가하고 오버클라우드를 배포합니다.

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack overcloud \
 -e [your environment files]
 -n /home/stack/common/network_data_routed_multi_cell.yaml \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml
```

3.3. 오버클라우드 스택 컨트롤 플레인에서 매개변수 정보 추출

기본 오버클라우드 스택에서 **default** 라는 첫 번째 셀에서 매개변수 정보를 추출합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.

2. **stackrc** 파일을 소싱합니다.

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

3. 오버클라우드 스택의 **기본** 셀에서 다중 셀 배포를 위한 새로운 공통 환경 파일로 셀 구성 및 암호 정보를 내보냅니다.

```
(undercloud)$ openstack overcloud cell export --control-plane-stack overcloud \
 -f --output-file common/default_cell_export.yaml \
 --working-dir /home/stack/overcloud-deploy/overcloud/
```

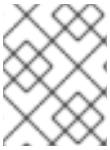
이 명령은 **EndpointMap**,**HostsEntry**,**AllNodesConfig**,**GlobalConfig** 매개변수 및 암호 정보를 공통 환경 파일로 내보냅니다.

작은 정보

환경 파일이 이미 있는 경우 **--force-overwrite** 또는 **-f** 옵션을 사용하여 명령을 입력합니다.

3.4. 라우팅된 네트워크에 대한 셀 역할 파일 생성

각 스택에서 다른 네트워크를 사용하는 경우 사용자 지정 셀 역할을 포함하는 각 셀 스택에 대한 셀 역할 파일을 생성합니다.



참고

각 사용자 지정 역할에 대한 플레이버를 생성해야 합니다. 자세한 내용은 셀 역할에 대한 호스트 지정을 참조하십시오.

절차

1. cell 스택에 필요한 다른 역할과 함께 **cellController** 역할을 포함하는 새 역할 데이터 파일을 생성합니다. 다음 예제에서는 **CellController** 및 **Compute**: 역할을 포함하는 역할 데이터 파일 **cell1_roles_data.yaml** 을 생성합니다.

```
(undercloud)$ openstack overcloud roles generate \
  --roles-path /usr/share/openstack-tripleo-heat-templates/roles \
  -o cell1/cell1_roles_data.yaml \
  Compute:ComputeCell1 \
  CellController:CellControllerCell1
```

2. 새 셀 역할 파일의 각 역할 정의에 **HostnameFormatDefault** 를 추가합니다.

```
- name: ComputeCell1
  ...
  HostnameFormatDefault: '%stackname%-compute-cell1-%index%'
  ServicesDefault:
  ...
  networks:
  ...
- name: CellControllerCell1
  ...
  HostnameFormatDefault: '%stackname%-cellcontrol-cell1-%index%'
  ServicesDefault:
  ...
  networks:
  ...
```

3. 아직 없는 경우 Networking 서비스(neutron) DHCP 및 Metadata 에이전트를 **Compute Cell1** 및 **cellControllerCell1** 역할에 추가합니다.

```
- name: ComputeCell1
  ...
  HostnameFormatDefault: '%stackname%-compute-cell1-%index%'
  ServicesDefault:
  - OS::TripleO::Services::NeutronDhcpAgent
  - OS::TripleO::Services::NeutronMetadataAgent
  ...
  networks:
  ...
- name: CellControllerCell1
  ...
  HostnameFormatDefault: '%stackname%-cellcontrol-cell1-%index%'
  ServicesDefault:
  - OS::TripleO::Services::NeutronDhcpAgent
  - OS::TripleO::Services::NeutronMetadataAgent
  ...
  networks:
  ...
```

4. **network_data_routed_multi_cell.yaml** 에서 구성한 서브넷을 **ComputeCell1** 및 cell **ControllerCell1** 역할에 추가합니다.

```
- name: ComputeCell1
  ...
  networks:
    InternalApi:
      subnet: internal_api_subnet_cell1
    Tenant:
      subnet: tenant_subnet_cell1
    Storage:
      subnet: storage_subnet_cell1
...
- name: CellControllerCell1
  ...
  networks:
    External:
      subnet: external_subnet
    InternalApi:
      subnet: internal_api_subnet_cell1
    Storage:
      subnet: storage_subnet_cell1
    StorageMgmt:
      subnet: storage_mgmt_subnet_cell1
    Tenant:
      subnet: tenant_subnet_cell1
```

3.5. 셀 역할에 대한 호스트 지정

셀 역할에 대한 베어 메탈 노드를 지정하려면 셀 역할의 노드를 태그하는 데 사용할 리소스 클래스로 베어 메탈 노드를 구성해야 합니다. 다음 절차에 따라 **cellcontrollercell1** 역할에 대한 베어 메탈 리소스 클래스를 생성합니다. 셀 컨트롤러 이름을 사용자 지정 역할의 이름으로 대체하여 사용자 지정 역할에 대해 이 절차를 반복합니다.

참고

다음 절차는 아직 프로비저닝되지 않은 새 오버클라우드 노드에 적용됩니다. 이미 프로비저닝된 기존 오버클라우드 노드에 리소스 클래스를 할당하려면 오버클라우드를 축소하여 노드를 프로비저닝 해제한 다음 오버클라우드를 확장하여 새 리소스 클래스 할당으로 노드를 다시 프로비저닝합니다. 자세한 내용은 오버클라우드 노드 스케일링을 참조하십시오.

프로세스

1. 노드 정의 템플릿에 **node.json** 또는 **node.yaml** 을 추가하여 **cellcontrollercell1** 역할의 베어 메탈 노드를 등록합니다. 자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드용 노드 등록을 참조하십시오.

2. 노드 하드웨어를 검사합니다.

```
(undercloud)$ openstack overcloud node introspect \
 --all-manageable --provide
```

   자세한 내용은 director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 베어 메탈 노드 하드웨어의 인벤토리 생성 을 참조하십시오.

3. 노드 목록을 검색하여 UUID를 확인합니다.

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

4. 사용자 지정 셀 컨트롤러 리소스 클래스를 사용하여 셀 컨트롤러로 지정할 각 베어 메탈 노드에 태그를 지정합니다.

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.CELL-CONTROLLER <node>
```

   - & **lt;node** >를 베어 메탈 노드의 이름 또는 UUID로 바꿉니다.

5. 노드 정의 파일 **overcloud-baremetal-deploy.yaml** 에 **cellcontrollercell1** 역할을 추가하고 노드에 할당할 예측 노드 배치, 리소스 클래스, 네트워크 토폴로지 또는 기타 속성을 정의합니다.

```
- name: cellcontrollercell1
  count: 1
  defaults:
    resource_class: baremetal.CELL1-CONTROLLER
    network_config:
      template: /home/stack/templates/nic-config/<role_topology_file>
  instances:
  - hostname: cell1-cellcontroller-%index%
    name: cell1controller
```

   - < **role_topology_file** >을 **cellcontrollercell1** 역할에 사용할 네트워크 토폴로지 파일의 이름으로 바꿉니다(예: **cell1_controller_net_top.j2**). 기존 네트워크 토폴로지를 재사용하거나 역할 또는 셀에 대한 새 사용자 지정 네트워크 인터페이스 템플릿을 생성할 수 있습니다. 자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리의* 사용자 지정 네트워크 인터페이스 템플릿을 참조하십시오. 기본 네트워크 정의 설정을 사용하려면 역할 정의에 **network_config** 를 포함하지 마십시오.
     노드 정의 파일에서 노드 속성을 구성하는 데 사용할 수 있는 속성에 대한 자세한 내용은 베어 메탈 노드 프로비저닝 속성을 참조하십시오. 노드 정의 파일의 예는 노드 정의 파일 예제 를 참조하십시오.

6. 역할의 새 노드를 프로비저닝합니다.

```
(undercloud)$ openstack overcloud node provision \
[--stack <stack>] \
[--network-config \]
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

- 선택 사항: **<stack** >을 베어 메탈 노드가 프로비저닝되는 스택의 이름으로 바꿉니다. 기본값은 **overcloud** 입니다.
- 선택 사항: **--network-config** 선택적 인수를 포함하여 네트워크 정의를 **cli-overcloud-node-network-config.yaml** Ansible 플레이북에 제공합니다. **network_config** 속성을 사용하여 노드 정의 파일에 네트워크 정의를 정의하지 않은 경우 기본 네트워크 정의가 사용됩니다.
- < **deployment_file** >을 배포 명령에 포함할 heat 환경 파일의 이름으로 교체합니다(예: **/home/stack/templates/overcloud-baremetal-deployed.yaml**).

7. 별도의 터미널에서 프로비저닝 진행 상황을 모니터링합니다. 프로비저닝이 성공하면 노드 상태가 **available** 에서 **active** 로 변경됩니다.

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

8. **--network-config** 옵션 없이 provisioning 명령을 실행한 경우 NIC 템플릿 파일을 가리키도록 **network-environment.yaml** 파일에서 < **Role>NetworkConfigTemplate** 매개변수를 구성합니다.

```
parameter_defaults:
  ComputeNetworkConfigTemplate: /home/stack/templates/nic-configs/compute.j2
  CellControllerCell1NetworkConfigTemplate: /home/stack/templates/nic-
configs/<role_topology_file>
  ControllerNetworkConfigTemplate: /home/stack/templates/nic-configs/controller.j2
```

- < **role_topology_file** >을 **cellcontrollercell1** 역할의 네트워크 토폴로지를 포함하는 파일 이름으로 바꿉니다(예: **cell1_controller_net_top.j2**). 기본 네트워크 토폴로지를 사용하려면 **controller.j2** 로 설정합니다.

3.6. 라우팅된 네트워크를 사용하여 각 셀 스택 구성 및 배포

다음 절차에 따라 하나의 셀 스택인 **cell1** 을 구성합니다. 모든 셀 스택을 배포할 때까지 배포하려는 추가 셀 스택에 대해 절차를 반복합니다.

절차

1. 셀별 매개변수(예: **/home/stack/cell1/cell1.yaml**)에 대한 추가 셀에 대한 새 환경 파일을 생성합니다.
2. 환경 파일에 다음 매개변수를 추가합니다.

```
resource_registry:
  OS::TripleO::CellControllerCell1::Net::SoftwareConfig: /home/stack/templates/nic-
configs/cellcontroller.yaml
  OS::TripleO::ComputeCell1::Net::SoftwareConfig: /home/stack/templates/nic-
configs/compute.yaml
```

```
parameter_defaults:
  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # Enable local metadata API for each cell
  NovaLocalMetadataPerCell: True

  #Disable network creation in order to use the `network_data.yaml` file from the overcloud
stack,
  # and create ports for the nodes in the separate stacks on the existing networks.
  ManageNetworks: false

  # Specify that this is an additional cell
  NovaAdditionalCell: True

  # The DNS names for the VIPs for the cell
  CloudDomain: redhat.local
  CloudName: cell1.redhat.local
  CloudNameInternal: cell1.internalapi.redhat.local
  CloudNameStorage: cell1.storage.redhat.local
  CloudNameStorageManagement: cell1.storagemgmt.redhat.local
  CloudNameCtlplane: cell1.ctlplane.redhat.local
```

3. 글로벌 컨트롤러 대신 각 셀에서 Compute 메타데이터 API를 실행하려면 셀 환경 파일에 다음 매개변수를 추가합니다.

```
parameter_defaults:
  NovaLocalMetadataPerCell: True
```

4. 셀 환경 파일에 셀의 VIP(가상 IP 주소) 정보를 추가합니다.

```
parameter_defaults:
  ...
  VipSubnetMap:
    InternalApi: internal_api_cell1
    Storage: storage_cell1
    StorageMgmt: storage_mgmt_cell1
    External: external_subnet
```

이렇게 하면 셀 컨트롤러 노드가 연결된 L2 네트워크 세그먼트와 연결된 서브넷에 가상 IP 주소가 생성됩니다.

5. 다른 환경 파일과 함께 스택에 환경 파일을 추가하고 셀 스택을 배포합니다.

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack cell1 \
 -e [your environment files] \
 -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
 -e /home/stack/templates/overcloud-networks-deployed.yaml \
 -e /home/stack/templates/overcloud-vip-deployed.yaml \
 -r /home/stack/cell1/cell1_roles_data.yaml \
 -n /home/stack/common/network_data_spine_leaf.yaml \
 -e /home/stack/common/default_cell_export.yaml \
 -e /home/stack/cell1/cell1.yaml
```

3.7. 배포 후 새 셀 서브넷 추가

다중 셀 환경을 배포한 후 오버클라우드 스택에 새 셀 서브넷을 추가하려면 **'UPDATE'** 를 포함하도록 **NetworkDeploymentActions** 값을 업데이트해야 합니다.

절차

1. 오버클라우드 스택의 환경 파일에 다음 구성을 추가하여 새 셀 서브넷으로 네트워크 구성을 업데이트합니다.

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

2. 새 셀 서브넷의 구성을 **/common/network_data_routed_multi_cell.yaml** 에 추가합니다.

3. 오버클라우드 스택을 배포합니다.

```
(undercloud)$ openstack overcloud deploy --templates \
 --stack overcloud \
 -n /home/stack/common/network_data_routed_multi_cell.yaml \
 -e [your environment files]
```

4. 선택 사항: **NetworkDeploymentActions** 를 다음 배포의 기본값으로 재설정합니다.

```
parameter_defaults:
  NetworkDeploymentActions: ['CREATE']
```

3.8. 다음 단계

- Compute 서비스 내에서 셀 생성 및 관리

4장. COMPUTE 서비스 내에서 셀 생성 및 관리

셀 스택을 사용하여 오버클라우드를 배포한 후 Compute 서비스 내에 셀을 생성해야 합니다. Compute 서비스 내에서 셀을 생성하려면 글로벌 API 데이터베이스에 셀 및 메시지 큐 매핑에 대한 항목을 생성합니다. 그런 다음 컨트롤러 노드 중 하나에서 셀 호스트 검색을 실행하여 셀 노드에 컴퓨팅 노드를 추가할 수 있습니다.

셀을 생성하려면 다음 작업을 수행해야 합니다.

1. **nova-manage** 유틸리티를 사용하여 글로벌 API 데이터베이스에 셀 및 메시지 큐 매핑 레코드를 생성합니다.
2. 각 셀에 컴퓨팅 노드를 추가합니다.
3. 각 셀의 가용성 영역을 생성합니다.
4. 각 셀의 모든 컴퓨팅 노드를 셀의 가용성 영역에 추가합니다.

4.1. 사전 요구 사항

- 여러 셀이 있는 오버클라우드를 구성하고 배포했습니다.

4.2. COMPUTE 서비스 내에서 셀 생성

새 셀 스택을 사용하여 오버클라우드를 배포한 후 Compute 서비스 내에 셀을 생성해야 합니다. Compute 서비스 내에서 셀을 생성하려면 글로벌 API 데이터베이스에 셀 및 메시지 큐 매핑에 대한 항목을 생성합니다. 참고: 만들고 시작하는 각 셀에 대해 이 프로세스를 반복해야 합니다. Ansible 플레이북의 단계를 자동화할 수 있습니다. Ansible 플레이북의 예는 OpenStack 커뮤니티 설명서 의 셀 생성 및 컴퓨팅 노드 검색 섹션을 참조하십시오. 커뮤니티 문서는 그대로 제공되며 공식적으로 지원되지 않습니다.

절차

1. 컨트롤 플레인 및 셀 컨트롤러의 IP 주소를 가져옵니다.

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r .ctlplane_ip)
$ CELL_CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/cell1/config-download/cell1/tripleo-ansible-inventory.yaml --host <cell_controller_node> | jq -r .ctlplane_ip)
```

   - & **lt;controller_node&** gt;를 컨트롤러 노드의 이름으로 바꿉니다(예: **controller-0**).
   - & **lt;cell_controller_node** >를 셀 컨트롤러 노드의 이름으로 바꿉니다(예: **cell1-controller-0**).

2. 모든 컨트롤러 노드에 셀 정보를 추가합니다. 이 정보는 언더클라우드의 셀 엔드포인트에 연결하는 데 사용됩니다. 다음 예제에서는 접두사 **cell1** 을 사용하여 셀 시스템만 지정하고 컨트롤러 시스템을 제외합니다.

```
(undercloud)$ CELL_INTERNALAPI_INFO=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 egrep cell1.*\.internalapi /etc/hosts)
(undercloud)$ ansible -i /home/stack/overcloud-deploy/overcloud/config-
```

```
download/overcloud/tripleo-ansible-inventory.yaml \
 Controller -b -m lineinfile -a "dest=/etc/hosts line=\"$CELL_INTERNALAPI_INFO\""
```

3. **transport_url** 매개변수에서 컨트롤러 셀의 메시지 큐 끝점과 **database.connection** 매개변수에서 컨트롤러 셀에 대한 데이터베이스 연결을 가져옵니다.

```
(undercloud)$ CELL_TRANSPORT_URL=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 sudo crudini --get /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf \
 DEFAULT transport_url)
(undercloud)$ CELL_MYSQL_VIP=$(ssh tripleo-admin@${CELL_CTRL_IP} \
 sudo crudini --get /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf \
 database connection | awk -F[@/] '{print $4}')
```

4. 글로벌 컨트롤러 노드 중 하나에 로그인하고 셀을 생성합니다.

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman \
 exec -i -u root nova_api \
 nova-manage cell_v2 create_cell --name cell1 \
 --database_connection "{scheme}://{username}:{password}@$CELL_MYSQL_VIP/nova?
{query}" \
 --transport-url "$CELL_TRANSPORT_URL"
```

5. 셀이 생성되고 셀 목록에 표시되는지 확인합니다.

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_cells --verbose
```

6. 컨트롤러 노드에서 컴퓨팅 서비스를 다시 시작합니다.

```
$ ansible -i /usr/bin/tripleo-ansible-inventory Controller -b -a \
"systemctl restart tripleo_nova_api tripleo_nova_conductor tripleo_nova_scheduler"
```

7. 셀 컨트롤러 서비스가 프로비저닝되었는지 확인합니다.

```
(overcloud)$ openstack compute service list -c Binary -c Host -c Status -c State
+----------------+------------------------+---------+-------+
| Binary         | Host                   | Status  | State |
+----------------+------------------------+---------+-------+
| nova-conductor | controller-0.ostest    | enabled | up    |
| nova-scheduler | controller-0.ostest    | enabled | up    |
| nova-conductor | cellcontroller-0.ostest | enabled | up    |
| nova-compute   | compute-0.ostest       | enabled | up    |
| nova-compute   | compute-1.ostest       | enabled | up    |
+----------------+------------------------+---------+-------+
```

4.3. 셀에 컴퓨팅 노드 추가

컨트롤러 노드 중 하나에서 셀 호스트 검색을 실행하여 컴퓨팅 노드를 검색하고 node-to-cell 매핑으로 API 데이터베이스를 업데이트합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. 셀의 컨트롤 플레인의 IP 주소를 가져옵니다.

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r .ctlplane_ip)
```

   - & **lt;controller_node&** gt;를 컨트롤러 노드의 이름으로 바꿉니다(예: **controller-0**).

3. 셀에 컴퓨팅 호스트를 노출하고 할당합니다.

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
  nova-manage cell_v2 discover_hosts --by-service --verbose
```

4. 컴퓨팅 호스트가 셀에 할당되었는지 확인합니다.

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
  nova-manage cell_v2 list_hosts
```

4.4. 셀 가용성 영역 생성

해당 셀의 컴퓨팅 노드에서 생성된 인스턴스가 동일한 셀의 다른 컴퓨팅 노드로만 마이그레이션되도록 각 셀에 대한 가용성 영역(AZ)을 생성해야 합니다. 셀 간에 인스턴스를 마이그레이션하는 것은 지원되지 않습니다.

AZ 셀을 생성한 후 셀의 모든 컴퓨팅 노드를 AZ 셀에 추가해야 합니다. 기본 셀은 Compute 셀과 다른 가용성 영역에 있어야 합니다.

절차

1. **overcloudrc** 파일을 소싱합니다.

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

2. 셀의 AZ를 만듭니다.

```
(overcloud)# openstack aggregate create \
 --zone <availability_zone> \
 <aggregate_name>
```

   - & **lt;availability_zone** >을 가용성 영역에 할당할 이름으로 바꿉니다.
   - & **lt;aggregate_name** >을 호스트 집계에 할당할 이름으로 바꿉니다.

3. 선택 사항: 가용성 영역에 메타데이터를 추가합니다.

```
(overcloud)# openstack aggregate set --property <key=value> \
  <aggregate_name>
```

   - &lt **;key=value&gt**;를 메타데이터 키-값 쌍으로 바꿉니다. 필요에 따라 많은 키-값 속성을 추가할 수 있습니다.
   - & **lt;aggregate_name&** gt;을 가용성 영역 호스트 집계의 이름으로 바꿉니다.

4. 셀에 할당된 컴퓨팅 노드 목록을 검색합니다.

```
$ ssh tripleo-admin@${CTRL_IP} sudo podman exec -i -u root nova_api \
 nova-manage cell_v2 list_hosts
```

5. 셀에 할당된 컴퓨팅 노드를 셀 가용 영역에 추가합니다.

```
(overcloud)# openstack aggregate add host <aggregate_name> \
 <host_name>
```

- 컴퓨팅 노드를 추가할 가용성 영역 호스트 집계의 이름으로 < **aggregate_** name>을 바꿉니다.
- & **lt;host_name** >을 가용성 영역에 추가할 컴퓨팅 노드의 이름으로 바꿉니다.

참고

- 이 단계에서 셀이 생성되지 않기 때문에 **OS::TripleO::Services::NovaAZConfig** 매개변수를 사용하여 배포 중에 AZ를 자동으로 생성할 수 없습니다.
- 셀 간에 인스턴스를 마이그레이션하는 것은 지원되지 않습니다. 인스턴스를 다른 셀로 이동하려면 이전 셀에서 삭제하고 새 셀에서 다시 만들어야 합니다.

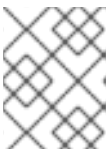
호스트 집계 및 가용성 영역에 대한 자세한 내용은 호스트 집계 생성 및 관리를 참조하십시오.

4.5. 셀에서 컴퓨팅 노드 삭제

셀에서 컴퓨팅 노드를 삭제하려면 셀에서 모든 인스턴스를 삭제하고 배치 데이터베이스에서 호스트 이름을 삭제해야 합니다.

절차

1. 셀의 컴퓨팅 노드에서 모든 인스턴스를 삭제합니다.



참고

셀 간에 인스턴스를 마이그레이션하는 것은 지원되지 않습니다. 인스턴스를 삭제하고 다른 셀에서 다시 생성해야 합니다.

2. 글로벌 컨트롤러 중 하나에서 셀에서 모든 컴퓨팅 노드를 삭제합니다.

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-
download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r
.ctlplane_ip)

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_hosts

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 delete_host --cell_uuid <uuid> --host <compute>
```

- & **lt;controller_node&** gt;를 컨트롤러 노드의 이름으로 바꿉니다(예: **controller-0**).

3. 배치 서비스에서 셀의 리소스 공급자를 삭제합니다. 나중에 다른 셀에 동일한 호스트 이름을 추가하려면 호스트 이름을 사용할 수 있는지 확인합니다.

```
(undercloud)$ source ~/overcloudrc

(overcloud)$ openstack resource provider list
+--------------------------------------+--------------------------------------+------------+
| uuid                                 | name                                 | generation |
+--------------------------------------+--------------------------------------+------------+
| 9cd04a8b-5e6c-428e-a643-397c9bebcc16 | computecell1-novacompute-0.site1.test |
11 |
+--------------------------------------+--------------------------------------+------------+

(overcloud)$ openstack resource provider  \
 delete 9cd04a8b-5e6c-428e-a643-397c9bebcc16
```

4.6. 셀 삭제

셀에서 컴퓨팅 노드 삭제에 설명된 대로 먼저 셀에서 모든 인스턴스 및 컴퓨팅 노드를 삭제해야 합니다. 그런 다음 셀 자체와 셀 스택을 삭제합니다.

프로세스

1. 글로벌 컨트롤러 중 하나에서 셀을 삭제합니다.

```
$ CTRL_IP=$(ansible-inventory -i /home/stack/overcloud-deploy/overcloud/config-
download/overcloud/tripleo-ansible-inventory.yaml --host <controller_node> | jq -r
.ctlplane_ip)

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 list_cells

$ ssh tripleo-admin@${CTRL_IP} sudo podman  \
 exec -i -u root nova_api \
 nova-manage cell_v2 delete_cell --cell_uuid <uuid>
```

- & **lt;controller_node&** gt;를 컨트롤러 노드의 이름으로 바꿉니다(예: **controller-0**).

2. 임시 Heat 프로세스를 시작하고 heat 환경을 내보냅니다.

```
(undercloud)$ openstack tripleo launch heat --heat-dir /home/stack/overcloud-
deploy/cell1/heat-launcher --restore-db
(undercloud)$ export OS_CLOUD=heat
```

3. 오버클라우드에서 셀 스택을 삭제합니다.

```
$ openstack stack delete <stack name> --wait --yes
```

참고

컨트롤러 및 컴퓨팅 셀에 대해 별도의 셀 스택을 배포한 경우 먼저 Compute 셀 스택을 삭제한 다음 컨트롤러 셀 스택을 삭제합니다.

4. 셀 스택 삭제가 완료되면 언더클라우드에서 임시 Heat 프로세스를 제거합니다.

```
(undercloud)$ openstack tripleo launch heat --kill
```