Red Hat OpenStack Platform 17.1

인스턴스의 고가용성 구성

컴퓨팅 인스턴스의 고가용성 구성

Last Updated: 2024-09-26

Red Hat OpenStack Platform 17.1 인스턴스의 고가용성 구성

컴퓨팅 인스턴스의 고가용성 구성

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 가이드에서는 인스턴스 HA(인스턴스 고가용성)를 관리하는 방법을 설명합니다. 인스턴스 HA를 사용하면 컴퓨팅 노드가 실패하면 RHOSP(Red Hat OpenStack Platform)가 다른 컴퓨팅 노드에서 인스턴스를 자동으로 비우고 다시 생성할 수 있습니다.

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

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/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
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. 인스턴스 HA 배포 도입 및 계획

Compute 인스턴스(인스턴스 HA)의 고가용성은 실패한 컴퓨팅 노드에서 인스턴스를 비우고 다른 컴퓨팅 노드에서 인스턴스를 다시 생성하는 데 사용할 수 있는 툴입니다.

인스턴스 HA는 공유 스토리지 또는 로컬 스토리지 환경에서 작동하므로 비어 있는 인스턴스가 고정 IP 주소 및 유동 IP 주소와 같은 동일한 네트워크 구성을 유지합니다. 다시 생성된 인스턴스는 새 컴퓨팅 노드 내에서 동일한 특성을 유지합니다.

1.1. 인스턴스 HA의 작동 방식

컴퓨팅 노드에 실패하면 오버클라우드 펜싱 에이전트가 노드를 펜싱한 다음 인스턴스 HA 에이전트가 실패한 컴퓨팅 노드에서 다른 컴퓨팅 노드로 인스턴스를 비웁니다.

컴퓨팅 노드가 실패하고 인스턴스 HA를 트리거하면 다음 이벤트가 발생합니다.

1. 오류가 발생할 때 **IPMI** 에이전트는 첫 번째 계층 펜싱을 수행합니다. 여기에는 노드를 물리적으로 재설정하여 오버클라우드에서 데이터 손상 또는 여러 동일한 인스턴스를 종료하고 방지합니다. 노드가 오프라인 상태이면 fenced로 간주됩니다.

2. 물리적 IPMI 펜싱 후 **fence-nova** 에이전트는 두 번째 계층 펜싱을 자동으로 수행하고 다음 명령을 실행하여 노드당 **"evacuate=yes"** 클러스터당 펜싱 노드를 표시합니다.

   ```
   $ attrd_updater -n evacuate -A name="evacuate" host="FAILEDHOST" value="yes"
   ```

   FAILEDHOST 는 실패한 컴퓨팅 노드의 이름입니다.

3. **nova-evacuate** 에이전트는 백그라운드에서 지속적으로 실행되며 주기적으로 **"evacuate=yes"** 속성이 있는 노드가 있는지 확인합니다. **nova-evacuate** 에서 펜싱된 노드에 이 속성이 포함되어 있음을 감지하면 에이전트는 노드를 비우기 시작합니다. 비우기 프로세스는 언제든지 수행할 수 있는 수동 인스턴스 비우기 프로세스와 유사합니다.

4. IPMI 재설정 후 실패한 노드가 다시 시작되면 해당 노드의 **nova-compute** 프로세스도 자동으로 시작됩니다. 노드가 이전에 펜싱되었기 때문에 Pacemaker에서 노드에 영향을 미치지 않을 때까지 새 인스턴스를 실행하지 않습니다.

5. Pacemaker에서 컴퓨팅 노드가 온라인 상태임을 감지하면 노드에서 **compute-unfence-trigger** 리소스 에이전트를 시작하여 노드를 해제하고 인스턴스를 다시 실행할 수 있습니다.

추가 리소스

- 인스턴스 비우기

1.2. 인스턴스 HA 배포 계획

인스턴스 HA를 배포하기 전에 규정 준수의 리소스 이름을 검토하고 환경에 따라 스토리지 및 네트워킹을 구성합니다.

- 컴퓨팅 노드 호스트 이름과 Pacemaker 원격 리소스 이름은 W3C 이름 지정 규칙을 준수해야 합니다. 자세한 내용은 W3C 문서의 네임스페이스 및 이름 및 토큰 선언을 참조하십시오.

- 일반적으로 인스턴스 HA를 사용하려면 인스턴스의 디스크 이미지에 대한 공유 스토리지를 구성해야 합니다. 따라서 **no-shared-storage** 옵션을 사용하려는 경우 비우는 동안 **InvalidSharedStorage** 오류가 발생할 수 있으며 다른 컴퓨팅 노드에서 인스턴스가 시작되지 않습니다.

그러나 모든 인스턴스가 OpenStack 블록 스토리지(**cinder**) 볼륨에서 부팅되도록 구성된 경우 인스턴스의 디스크 이미지에 대한 공유 스토리지를 구성할 필요가 없으며 **no-shared-storage** 옵션을 사용하여 모든 인스턴스를 비울 수 있습니다.

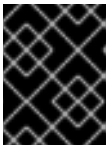
비우는 동안 인스턴스가 블록 스토리지 볼륨에서 부팅되도록 구성된 경우 비우는 인스턴스는 다른 컴퓨팅 노드의 동일한 볼륨에서 부팅됩니다. 따라서 OS 이미지와 애플리케이션 데이터가 OpenStack 블록 스토리지 볼륨에 저장되므로 비우는 인스턴스가 즉시 작업을 다시 시작합니다.

- Spine-Leaf 환경에 인스턴스 HA를 배포하는 경우 컨트롤러 및 컴퓨팅 노드에 대한 단일 **internal_api** 네트워크를 정의해야 합니다. 그런 다음 각 리프의 서브넷을 정의할 수 있습니다. Spine-Leaf 네트워크 구성에 대한 자세한 내용은 *Spine Leaf Networking* 가이드에서 역할 데이터 파일 생성 을 참조하십시오.

- Red Hat OpenStack Platform 13 이상에서는 director를 사용하여 오버클라우드 업그레이드의 일부로 인스턴스 HA를 업그레이드합니다. 오버클라우드 업그레이드에 대한 자세한 내용은 Red Hat OpenStack Platform 가이드의 마이너 업데이트 수행을 참조하십시오.

- vTPM 장치로 인스턴스를 비울 수 없습니다. vTPM 장치를 사용하여 인스턴스를 배포하는 경우 비워야 하는 다른 인스턴스에서 플레이버를 사용하거나 **evacuable** 속성으로 태그된 이미지를 사용해야 합니다. 비우기 위해 인스턴스를 지정하는 방법에 대한 자세한 내용은 인스턴스 HA로 비우도록 인스턴스 설계를 참조하십시오.

- 설치 후 director를 사용하여 인스턴스 HA 비활성화는 지원되지 않습니다. 배포에서 인스턴스 HA 구성 요소를 수동으로 제거하는 해결 방법은 컨트롤러 노드에서 인스턴스 HA 구성 요소를 제거하는 방법 문서를 참조하십시오. .



중요

이 해결방법은 프로덕션 환경에 대해 확인되지 않습니다. 프로덕션 환경에서 구현하기 전에 테스트 환경에서 절차를 확인해야 합니다.

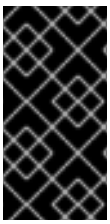
1.3. 인스턴스 HA 리소스 에이전트

인스턴스 HA는 **fence_compute**,**NovaEvacuate**, **comput-unfence-trigger** 리소스 에이전트를 사용하여 컴퓨팅 노드가 실패하면 인스턴스를 비우고 다시 생성합니다.

에이전트 이름	클러스터 내 이름	Role
fence_compute	**fence-nova**	노드를 사용할 수 없게 되면 비우기 위해 컴퓨팅 노드를 표시합니다.
NovaEvacuate	**nova-evacuate**	실패한 노드의 인스턴스를 비웁니다. 이 에이전트는 컨트롤러 노드 중 하나에서 실행됩니다.
Dummy	**compute-unfence-trigger**	펜싱된 노드를 해제하고 노드가 인스턴스를 다시 실행할 수 있습니다.

2장. 인스턴스 HA 설치 및 구성

RHOSP(Red Hat OpenStack Platform) director를 사용하여 HA(인스턴스 고가용성)를 배포합니다. 그러나 새 오버클라우드에서 새 인스턴스 HA 배포를 구성하려면 추가 단계를 수행해야 합니다. 단계를 완료하면 인스턴스 HA가 사용자 지정 역할이 있는 컴퓨팅 노드의 하위 집합에서 실행됩니다.



중요

인스턴스 HA는 RHOSP HCI(하이퍼 컨버지드 인프라) 환경에서 지원되지 않습니다. RHOSP HCI 환경에서 인스턴스 HA를 사용하려면 인스턴스 HA를 사용하려면 Compute 노드 하위 집합을 ComputeInstanceHA 역할로 지정해야 합니다. Red Hat Ceph Storage 서비스는 인스턴스 HA를 호스팅하는 컴퓨팅 노드에서 호스팅해서는 안 됩니다.

중요

표준 또는 사용자 지정 역할을 사용하는 기존 오버클라우드와 같이 다른 환경에서 인스턴스 HA를 활성화하려면 배포와 관련된 절차만 수행하고 그에 따라 템플릿을 조정합니다.

2.1. 인스턴스 HA 역할 및 프로필 구성

인스턴스 HA를 배포하기 전에 **roles-data.yaml** 파일에 Instance HA 역할을 추가하고 인스턴스 HA 프로필로 관리할 각 컴퓨팅 노드에 태그를 지정한 다음 **overcloud-baremetal-deploy.yaml** 파일에 추가합니다. 특정 역할에 대한 오버클라우드 노드 지정에 대한 자세한 내용은 프로필 일치를 통해 역할에 대한 오버클라우드 노드 설계를 참조하십시오. 예를 들어 **computeiha** 프로필을 사용하여 노드를 구성할 수 있습니다.

프로세스

1. 등록된 각 노드의 기존 기능을 확인합니다.

```
(undercloud)$ openstack baremetal node show <node> -f json -c properties | jq -r
.properties.capabilities
```

2. 노드의 기존 기능에 **profile:computeiha** 를 추가하여 역할 프로필에 일치시킬 각 베어 메탈 노드에 프로필 기능을 할당합니다.

```
(undercloud)$ openstack baremetal node set  <node> --property
capabilities="profile:computeiha,<capability_1>,...,<capability_n>"
```

   - <node>를 베어 메탈 노드의 ID로 바꿉니다.
   - <capability_1> 및 <capability_n>까지 모든 기능을 1단계에서 확인한 각 기능으로 바꿉니다.

3. 아직 정의되지 않은 경우 **overcloud-baremetal-deploy.yaml** 파일에 역할을 추가합니다.

4. **overcloud-baremetal-deploy.yaml** 을 편집하여 역할의 노드에 할당할 프로필을 정의합니다.

```
- name: ComputeInstanceHA
  count: 2
  hostname_format: compute-%index%
  defaults:
    profile: computeiha
    network_config:
      template: /home/stack/composable_roles/network/nic-configs/compute.j2
```

```
  networks:
  - network: ctlplane
    vif: true
  - network: internal_api
  - network: tenant
  - network: storage
```

5. 오버클라우드 노드를 프로비저닝합니다.

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

- <stack>을 베어 메탈 노드를 프로비저닝한 스택의 이름으로 바꿉니다. 기본값은 **overcloud** 입니다.
- <deployment_file>을 배포 명령으로 포함할 생성된 heat 환경 파일의 이름으로 교체합니다 (예: **/home/stack/templates/overcloud-baremetal-deployed.yaml**).

2.2. 인스턴스 HA를 사용하여 오버클라우드에서 펜싱 활성화

펜싱 정보를 사용하여 환경 파일을 생성하여 오버클라우드의 모든 컨트롤러 및 컴퓨팅 노드에서 펜싱을 활성화합니다.

절차

1. ***~/templates*** 와 같은 액세스 가능한 위치에 환경 파일을 생성하고 다음 콘텐츠를 포함합니다.

```
parameter_defaults:
  EnableFencing: true
  FencingConfig:
    devices:
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:c7
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6230
        passwd: password
        lanplus: 1
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:cb
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6231
        passwd: password
        lanplus: 1
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:cf
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6232
```

```
      passwd: password
      lanplus: 1
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:d3
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6233
        passwd: password
        lanplus: 1
    - agent: fence_ipmilan
      host_mac: 00:ec:ad:cb:3c:d7
      params:
        login: admin
        ipaddr: 192.168.24.1
        ipport: 6234
        passwd: password
        lanplus: 1
```

2. 컴퓨팅 인스턴스에 공유 스토리지를 사용하지 않는 경우 생성한 환경 파일에 다음 매개변수를 추가합니다.

```
parameter_defaults:
  ExtraConfig:
    tripleo::instanceha::no_shared_storage: true
```

추가 리소스

- 1.2절. "인스턴스 HA 배포 계획"
- STONITH를 사용하여 컨트롤러 노드 펜싱

2.3. 인스턴스 HA를 사용하여 오버클라우드 배포

오버클라우드를 이미 배포한 경우 생성한 추가 인스턴스 HA 파일을 사용하여 **openstack overcloud deploy** 명령을 다시 실행할 수 있습니다. 언더클라우드를 생성한 후 언제든지 오버클라우드에 대한 인스턴스 HA를 구성할 수 있습니다.

사전 요구 사항

- 인스턴스 HA 역할 및 프로필을 구성했습니다.
- 오버클라우드에 펜싱을 활성화했습니다.

프로세스

- **openstack overcloud deploy** 명령을 **-e** 옵션과 함께 사용하여 **compute-instanceha.yaml** 환경 파일을 포함하고 추가 환경 파일을 포함합니다.

```
$ openstack overcloud deploy --templates \
 -e <fencing_environment_file> \
 -r my_roles_data.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/compute-instanceha.yaml
```

& **lt;fencing_environment_file** >을 환경에 적절한 파일 이름으로 바꿉니다.

참고

- **compute-instanceha.yaml** 환경 파일을 수정하지 마십시오.
- 오버클라우드 배포에 포함할 각 환경 파일의 전체 경로를 포함합니다.

배포 후 각 컴퓨팅 노드에는 **STONITH** 장치 및 **pacemaker_remote** 서비스가 포함됩니다.

2.4. 인스턴스 HA 비우기 테스트

인스턴스 HA가 인스턴스를 올바르게 비우도록 테스트하기 위해 컴퓨팅 노드에서 비우기를 트리거하고 인스턴스 HA 에이전트가 다른 컴퓨팅 노드에서 인스턴스를 성공적으로 비우고 다시 생성되었는지 확인합니다.

주의

다음 절차에서는 컴퓨팅 노드를 의도적으로 충돌하여 인스턴스 HA가 있는 인스턴스 자동 비우기를 트리거합니다.

사전 요구 사항

- 인스턴스 HA는 컴퓨팅 노드에 배포됩니다.

절차

1. 오버클라우드에서 하나 이상의 인스턴스를 시작합니다.

```
stack@director $ . overcloudrc
stack@director $ openstack server create --image cirros --flavor 2 test-failover
stack@director $ openstack server list -c Name -c Status
```

2. 인스턴스를 호스팅하는 컴퓨팅 노드에 로그인하고 **root** 사용자로 변경합니다. **compute-n** 을 컴퓨팅 노드의 이름으로 교체합니다.

```
stack@director $ . stackrc
stack@director $ ssh -l tripleo-admin compute-n
tripleo-admin@compute-n $ su -
```

3. 컴퓨팅 노드를 충돌합니다.

```
root@compute-n $ echo c > /proc/sysrq-trigger
```

4. 노드가 다시 시작될 때까지 몇 분 정도 기다린 다음 컴퓨팅 노드의 인스턴스가 다른 컴퓨팅 노드에 다시 생성되는지 확인합니다.

```
stack@director $ openstack server list -c Name -c Status
stack@director $ openstack compute service list
```

2.5. 인스턴스 HA로 비우도록 인스턴스 지정

기본적으로 인스턴스 HA는 실패한 노드의 모든 인스턴스를 비웁니다. 특정 이미지 또는 플레이버로만 인스턴스를 비우도록 인스턴스 HA를 구성할 수 있습니다.

사전 요구 사항

- 인스턴스 HA가 오버클라우드에 배포됩니다.

절차

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

```
$ source ~/overcloudrc
```

3. 다음 옵션 중 하나를 사용합니다.
   - 이미지에 태그를 지정합니다.

```
(overcloud) $ openstack image set --tag evacuable <image_id>
```

     & **lt;image_id** >를 비우기 원하는 이미지의 ID로 바꿉니다.
   - 플레이버에 태그를 지정합니다.

```
(overcloud) $ openstack flavor set --property evacuable=true <flavor_id>
```

     & **lt;flavor_id** >를 비우려는 플레이버의 ID로 바꿉니다.

2.6. 추가 리소스

- director를 사용하여 Red Hat OpenStack Platform 설치 및 관리
- 구성 가능 서비스 및 사용자 지정 역할

3장. 인스턴스 HA를 사용하여 언더클라우드 및 오버클라우드에서 유지보수 수행

언더클라우드 및 오버클라우드에서 유지보수를 수행하려면 오버클라우드를 시작할 때 최소한의 문제를 확인하기 위해 특정 순서로 언더클라우드 및 오버클라우드 노드를 종료하고 시작해야 합니다. 노드를 중지하고 노드에서 Pacemaker 리소스를 비활성화하여 특정 컴퓨팅 또는 컨트롤러 노드에서 유지보수를 수행할 수도 있습니다.

3.1. 사전 요구 사항

- 인스턴스 HA가 활성화된 실행 중인 언더클라우드 및 오버클라우드.

3.2. 언더클라우드 및 오버클라우드 종료 순서

Red Hat OpenStack Platform 환경을 종료하려면 다음 순서로 오버클라우드 및 언더클라우드를 종료해야 합니다.

1. 오버클라우드 컴퓨팅 노드에서 인스턴스 종료
2. 컴퓨팅 노드 종료
3. 컨트롤러 노드에서 고가용성 및 OpenStack Platform 서비스를 모두 중지합니다.
4. Ceph Storage 노드 종료
5. 컨트롤러 노드 종료
6. 언더클라우드 종료

3.2.1. 오버클라우드 컴퓨팅 노드에서 인스턴스 종료

Red Hat OpenStack Platform 환경 종료 과정의 일환으로 Compute 노드를 종료하기 전에 컴퓨팅 노드의 모든 인스턴스를 종료합니다.

사전 요구 사항

- 활성 Compute 서비스가 있는 오버클라우드

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. 오버클라우드의 인증 정보 파일을 가져옵니다.

```
$ source ~/overcloudrc
```

3. 오버클라우드에서 실행 중인 인스턴스를 확인합니다.

```
$ openstack server list --all-projects
```

4. 오버클라우드에서 각 인스턴스를 중지합니다.

```
$ openstack server stop <INSTANCE>
```

-

오버클라우드의 모든 인스턴스를 중지할 때까지 각 인스턴스에 대해 이 단계를 반복합니다.

3.2.2. 오버클라우드 컴퓨팅 노드에서 인스턴스 HA 서비스 중지

Red Hat OpenStack Platform 환경 종료 과정의 일환으로 인스턴스를 중지하고 컴퓨팅 노드를 종료하기 전에 컴퓨팅 노드에서 실행되는 모든 인스턴스 HA 서비스를 종료해야 합니다.

사전 요구 사항

- 활성 Compute 서비스가 있는 오버클라우드
- 컴퓨팅 노드에서 인스턴스 HA가 활성화됨

절차

1. Pacemaker를 실행하는 오버클라우드 노드에 **root** 사용자로 로그인합니다.
2. 각 컴퓨팅 노드에서 Pacemaker 원격 리소스를 비활성화합니다.
   a. 컴퓨팅 노드에서 Pacemaker 원격 리소스를 확인합니다.

   ```
   # pcs resource status
   ```

   이러한 리소스는 **ocf::pacemaker:remote** 에이전트를 사용하며 일반적으로 **overcloud-novacomputeiha-0** 과 같은 컴퓨팅 노드 호스트 형식으로 이름이 지정됩니다.

   b. 각 Pacemaker 원격 리소스를 비활성화합니다. 다음 예제에서는 **overcloud-novacomputeiha-0** 에 대한 리소스를 비활성화하는 방법을 보여줍니다.

   ```
   # pcs resource disable overcloud-novacomputeiha-0
   ```

3. 컴퓨팅 노드 STONITH 장치를 비활성화합니다.
   a. 컴퓨팅 노드 STONITH 장치를 식별합니다.

   ```
   # pcs stonith status
   ```

   b. 각 컴퓨팅 노드 STONITH 장치를 비활성화합니다.

   ```
   # pcs stonith disable <STONITH_DEVICE>
   ```

3.2.3. 컴퓨팅 노드 종료

Red Hat OpenStack Platform 환경 종료 과정의 일환으로 각 컴퓨팅 노드에 로그인하여 종료합니다.

사전 요구 사항

- 컴퓨팅 노드에서 모든 인스턴스 종료

절차

1. 컴퓨팅 노드에 **root** 사용자로 로그인합니다.

2. 노드를 종료합니다.

```
# shutdown -h now
```

3. 모든 컴퓨팅 노드를 종료할 때까지 각 컴퓨팅 노드에 대해 다음 단계를 수행합니다.

3.2.4. 컨트롤러 노드에서 서비스 중지

Red Hat OpenStack Platform 환경 종료 과정의 일환으로 컨트롤러 노드에서 서비스를 중지한 후 노드를 종료합니다. 여기에는 Pacemaker 및 systemd 서비스가 포함됩니다.

사전 요구 사항

- 활성 Pacemaker 서비스가 있는 오버클라우드

절차

1. **root** 사용자로 컨트롤러 노드에 로그인합니다.
2. Pacemaker 클러스터를 중지합니다.

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

   이 명령은 모든 노드에서 클러스터를 중지합니다.

3. Pacemaker 서비스가 중지될 때까지 기다린 후 서비스가 중지되었는지 확인합니다.
   a. Pacemaker 상태를 확인합니다.

```
# pcs status
```

   b. Podman에서 Pacemaker 서비스가 실행되고 있지 않은지 확인합니다.

```
# podman ps --filter "name=.*-bundle.*"
```

4. Red Hat OpenStack Platform 서비스를 중지합니다.

```
# systemctl stop 'tripleo_*'
```

5. 서비스가 중지될 때까지 기다린 후 Podman에서 서비스가 더 이상 실행되지 않는지 확인합니다.

```
# podman ps
```

3.2.5. Ceph Storage 노드 종료

Red Hat OpenStack Platform 환경 종료 과정의 일환으로 Ceph Storage 서비스를 비활성화한 다음 각 Ceph Storage 노드에 로그인하여 종료합니다.

사전 요구 사항

- 정상적인 Ceph Storage 클러스터
- Ceph MON 서비스는 독립 실행형 Ceph MON 노드 또는 컨트롤러 노드에서 실행 중입니다.

절차

1. 컨트롤러 노드 또는 독립 실행형 Ceph MON 노드와 같은 Ceph MON 서비스를 실행하는 노드에 **root** 사용자로 로그인합니다.

2. 클러스터 상태를 확인합니다. 다음 예제에서 **podman** 명령은 컨트롤러 노드의 Ceph MON 컨테이너 내에서 상태 점검을 실행합니다.

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

   상태가 **HEALTH_OK**인지 확인합니다.

3. 클러스터에 **noout**, **norecover**, **norebalance**, **nobackfill**, **nodown**, **pause** 플래그를 설정합니다. 다음 예제에서 **podman** 명령은 컨트롤러 노드의 Ceph MON 컨테이너를 통해 이러한 플래그를 설정합니다.

```
# sudo podman exec -it ceph-mon-controller-0 ceph osd set noout
# sudo podman exec -it ceph-mon-controller-0 ceph osd set norecover
# sudo podman exec -it ceph-mon-controller-0 ceph osd set norebalance
# sudo podman exec -it ceph-mon-controller-0 ceph osd set nobackfill
# sudo podman exec -it ceph-mon-controller-0 ceph osd set nodown
# sudo podman exec -it ceph-mon-controller-0 ceph osd set pause
```

4. 각 Ceph Storage 노드를 종료합니다.

   a. **root** 사용자로 Ceph Storage 노드에 로그인합니다.

   b. 노드를 종료합니다.

```
# shutdown -h now
```

   c. 모든 Ceph Storage 노드를 종료할 때까지 각 Ceph Storage 노드에 대해 다음 단계를 수행합니다.

5. 독립 실행형 Ceph MON 노드를 종료합니다.

   a. 독립 실행형 Ceph MON 노드에 **root** 사용자로 로그인합니다.

   b. 노드를 종료합니다.

```
# shutdown -h now
```

   c. 독립 실행형 Ceph MON 노드를 모두 종료할 때까지 각 독립 실행형 Ceph MON 노드에 대해 다음 단계를 수행합니다.

추가 리소스

- "시스템을 종료하고 전체 ceph 클러스터를 가져오는 절차는 무엇입니까?"

3.2.6. 컨트롤러 노드 종료

Red Hat OpenStack Platform 환경 종료 과정의 일환으로 각 컨트롤러 노드에 로그인하여 종료합니다.

사전 요구 사항

- Pacemaker 클러스터 중지
- 컨트롤러 노드에서 모든 Red Hat OpenStack Platform 서비스를 중지합니다.

절차

1. **root** 사용자로 컨트롤러 노드에 로그인합니다.
2. 노드를 종료합니다.

```
# shutdown -h now
```

3. 모든 컨트롤러 노드를 종료할 때까지 각 컨트롤러 노드에 대해 다음 단계를 수행합니다.

3.2.7. 언더클라우드 종료

Red Hat OpenStack Platform 환경 종료 과정의 일환으로 언더클라우드 노드에 로그인하여 언더클라우드를 종료합니다.

사전 요구 사항

- 실행 중인 언더클라우드

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. 언더클라우드를 종료합니다.

```
$ sudo shutdown -h now
```

3.3. 시스템 유지보수 수행

언더클라우드와 오버클라우드를 완전히 종료한 후 해당 환경의 시스템에 대한 유지 관리를 수행한 다음 언더클라우드 및 오버클라우드를 시작합니다.

3.4. 언더클라우드 및 오버클라우드 시작 순서

Red Hat OpenStack Platform 환경을 시작하려면 다음 순서로 언더클라우드 및 오버클라우드를 시작해야 합니다.

1. 언더클라우드를 시작합니다.
2. 컨트롤러 노드를 시작합니다.
3. Ceph Storage 노드를 시작합니다.
4. 컴퓨팅 노드를 시작합니다.
5. 오버클라우드 컴퓨팅 노드에서 인스턴스를 시작합니다.

3.4.1. 언더클라우드 시작

Red Hat OpenStack Platform 환경 시작 과정의 일환으로 언더클라우드 노드의 전원을 켜고 언더클라우드에 로그인한 다음, 언더클라우드 서비스를 확인합니다.

사전 요구 사항

- 언더클라우드의 전원이 꺼집니다.

프로세스

- 언더클라우드의 전원을 켜고 언더클라우드가 부팅될 때까지 기다립니다.

검증

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

```
$ source ~/stackrc
```

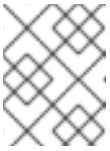
3. 언더클라우드에서 서비스를 확인합니다.

```
$ systemctl list-units 'tripleo_*'
```

4. **tripleo-ansible-inventory.yaml** 이라는 정적 인벤토리 파일을 확인합니다.

```
$ validation run --group pre-introspection -i <inventory_file>
```

- < **inventory_file** >을 Ansible 인벤토리 파일의 이름 및 위치로 바꿉니다(예: **~/tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml**).



참고

검증을 실행하면 출력의 **Reasons** 열이 79자로 제한됩니다. 검증 결과를 전체로 보려면 검증 로그 파일을 확인합니다.

5. 모든 서비스와 컨테이너가 활성 상태이고 정상 상태인지 확인합니다.

```
$ validation run --validation service-status --limit undercloud -i <inventory_file>
```

추가 리소스

- 검증 프레임워크 사용

3.4.2. 컨트롤러 노드 시작

Red Hat OpenStack Platform 환경 시작 과정의 일환으로 각 컨트롤러 노드의 전원을 켜고 노드에서 Pacemaker가 아닌 서비스를 확인합니다.

사전 요구 사항

- 컨트롤러 노드의 전원이 꺼집니다.

프로세스

- 각 컨트롤러 노드의 전원을 켭니다.

검증

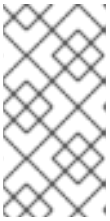
1. 각 컨트롤러 노드에 **root** 사용자로 로그인합니다.
2. 컨트롤러 노드에서 서비스를 확인합니다.

```
$ systemctl -t service
```

Pacemaker 기반이 아닌 서비스만 실행 중입니다.

3. Pacemaker 서비스가 시작될 때까지 기다린 후 서비스가 시작되었는지 확인합니다.

```
$ pcs status
```



참고

환경에서 인스턴스 HA를 사용하는 경우 Compute 노드를 시작하거나 **pcs stonith 로 수동 unfence 작업을 수행할 때까지 Pacemaker 리소스가 시작되지 않습니다 <compute_node> 명령을 확인합니다**. 인스턴스 HA를 사용하는 각 컴퓨팅 노드에서 이 명령을 실행해야 합니다.

3.4.3. Ceph Storage 노드 시작

Red Hat OpenStack Platform 환경 시작 과정의 일환으로 Ceph MON 및 Ceph Storage 노드의 전원을 켜고 Ceph Storage 서비스를 활성화합니다.

사전 요구 사항

- 전원이 꺼진 Ceph Storage 클러스터
- Ceph MON 서비스는 전원이 꺼진 독립 실행형 Ceph MON 노드 또는 전원이 켜진 컨트롤러 노드에서 활성화됩니다.

절차

1. 환경에 독립 실행형 Ceph MON 노드가 있는 경우 각 Ceph MON 노드의 전원을 켭니다.
2. 각 Ceph Storage 노드의 전원을 켭니다.
3. 컨트롤러 노드 또는 독립 실행형 Ceph MON 노드와 같은 Ceph MON 서비스를 실행하는 노드에 **root** 사용자로 로그인합니다.
4. 클러스터 노드의 상태를 확인합니다. 다음 예제에서 **podman** 명령은 컨트롤러 노드의 Ceph MON 컨테이너 내에서 상태 점검을 실행합니다.

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

각 노드의 전원이 켜져 있고 연결되어 있는지 확인합니다.

5. 클러스터의 **noout**, **norecover**, **norebalance**, **nobackfill**, **nodown** 및 **pause** 플래그를 설정 해제합니다. 다음 예제에서 **podman** 명령은 컨트롤러 노드의 Ceph MON 컨테이너를 통해 이러한 플래그를 설정 해제합니다.

```
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset noout
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset norecover
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset norebalance
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset nobackfill
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset nodown
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset pause
```

검증

1. 클러스터 상태를 확인합니다. 다음 예제에서 **podman** 명령은 컨트롤러 노드의 Ceph MON 컨테이너 내에서 상태 점검을 실행합니다.

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

상태가 **HEALTH_OK**인지 확인합니다.

추가 리소스

- "시스템을 종료하고 전체 ceph 클러스터를 가져오는 절차는 무엇입니까?"

3.4.4. 컴퓨팅 노드 시작

Red Hat OpenStack Platform 환경 시작 과정의 일환으로 각 컴퓨팅 노드의 전원을 켜고 노드에서 서비스를 확인합니다.

사전 요구 사항

- 전원이 꺼진 컴퓨팅 노드

절차

1. 각 컴퓨팅 노드의 전원을 켭니다.

검증

1. 각 Compute 노드에 **root** 사용자로 로그인합니다.
2. 컴퓨팅 노드에서 서비스를 확인합니다.

```
$ systemctl -t service
```

3.4.5. 오버클라우드 컴퓨팅 노드에서 인스턴스 HA 서비스 시작

Red Hat OpenStack Platform 환경 시작 과정의 일환으로 컴퓨팅 노드에서 모든 인스턴스 HA 서비스를 시작합니다.

사전 요구 사항

- 컴퓨팅 노드를 실행하는 오버클라우드

- 컴퓨팅 노드에서 인스턴스 HA가 활성화됨

절차

1. Pacemaker를 실행하는 오버클라우드 노드에 **root** 사용자로 로그인합니다.
2. 컴퓨팅 노드의 STONITH 장치를 활성화합니다.
   a. 컴퓨팅 노드 STONITH 장치를 식별합니다.

   ```
   # pcs stonith status
   ```

   b. 컴퓨팅 노드의 STONITH 오류를 지웁니다.

   ```
   # pcs stonith confirm <COMPUTE_NODE>
   ```

   이 명령은 노드를 정리 STONITH 상태로 반환합니다.

   c. 컴퓨팅 노드 STONITH 장치를 활성화합니다.

   ```
   # pcs stonith enable <STONITH_DEVICE>
   ```

   d. STONITH를 사용하여 각 컴퓨팅 노드에 대해 다음 단계를 수행합니다.
3. 각 컴퓨팅 노드에서 Pacemaker 원격 리소스를 활성화합니다.
   a. 컴퓨팅 노드에서 Pacemaker 원격 리소스를 식별합니다.

   ```
   # pcs resource status
   ```

   이러한 리소스는 **ocf::pacemaker:remote** 에이전트를 사용하며 일반적으로 **overcloud-novacomputeiha-0** 과 같은 컴퓨팅 노드 호스트 형식으로 이름이 지정됩니다.

   b. 각 Pacemaker 원격 리소스를 활성화합니다. 다음 예제에서는 **overcloud-novacomputeiha-0** 에 대한 리소스를 활성화하는 방법을 보여줍니다.

   ```
   # pcs resource enable overcloud-novacomputeiha-0
   ```

   c. Pacemaker 원격 관리를 사용하여 각 컴퓨팅 노드에 대해 다음 단계를 수행합니다.
4. Pacemaker 서비스가 시작될 때까지 기다린 후 서비스가 시작되었는지 확인합니다.

   ```
   # pcs status
   ```
5. 시작 프로세스 중에 Pacemaker 리소스를 시작하지 못하는 경우 리소스의 상태 및 실패 수를 재설정합니다.

   ```
   # pcs resource cleanup
   ```

참고

일부 서비스를 시작하는 데 더 많은 시간이 필요할 수 있습니다(예: **fence_compute** 및 **fence_kdump**).

3.4.6. 오버클라우드 컴퓨팅 노드에서 인스턴스 시작

Red Hat OpenStack Platform 환경을 시작하는 과정의 일환으로 컴퓨팅 노드에서 인스턴스를 시작합니다.

사전 요구 사항

- 활성 노드가 있는 활성 오버클라우드

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. 오버클라우드의 인증 정보 파일을 가져옵니다.

```
$ source ~/overcloudrc
```

3. 오버클라우드에서 실행 중인 인스턴스를 확인합니다.

```
$ openstack server list --all-projects
```

4. 오버클라우드에서 인스턴스를 시작합니다.

```
$ openstack server start <INSTANCE>
```

4장. 인스턴스 HA를 사용하여 컴퓨팅 노드 및 컨트롤러 노드에서 유지보수 수행

컴퓨팅 노드 또는 인스턴스 HA가 있는 컨트롤러 노드에서 유지보수를 수행하려면 **대기** 모드로 설정하고 노드에서 Pacemaker 리소스를 비활성화하여 노드를 중지합니다. 유지보수 작업을 완료한 후 노드를 시작하고 Pacemaker 리소스가 정상인지 확인합니다.

사전 요구 사항

- 인스턴스 HA가 활성화된 실행 중인 오버클라우드

절차

1. 컨트롤러 노드에 로그인하고 컴퓨팅 또는 컨트롤러 노드를 중지합니다.

```
# pcs node standby <node UUID>
```

중요

중지하려는 노드에서 다른 노드에 로그인해야 합니다.

2. 노드에서 Pacemaker 리소스를 비활성화합니다.

```
# pcs resource disable <ocf::pacemaker:remote on the node>
```

3. 노드에서 유지보수 작업을 수행합니다.

4. IPMI 연결을 복원하고 노드를 시작합니다. 진행하기 전에 노드가 준비될 때까지 기다립니다.

5. 노드에서 Pacemaker 리소스를 활성화하고 노드를 시작합니다.

```
# pcs resource enable <ocf::pacemaker:remote on the node>
# pcs node unstandby <node UUID>
```

6. 노드를 유지보수 모드로 설정하면 오버클라우드의 인증 정보 파일을 가져오고 유지보수 모드에서 노드를 설정 해제합니다.

```
# source stackrc
# openstack baremetal node maintenance unset <baremetal node UUID>
```

검증

1. Pacemaker 리소스가 활성 상태이고 정상인지 확인합니다.

```
# pcs status
```

2. 시작 프로세스 중에 Pacemaker 리소스를 시작하지 못하는 경우 **pcs resource cleanup** 명령을 실행하여 리소스의 상태 및 실패 수를 재설정합니다.

3. 노드를 중지하기 전에 컴퓨팅 노드에서 인스턴스를 비우는 경우 인스턴스가 다른 노드로 마이그레이션되었는지 확인합니다.

```
# openstack server list --long
# nova migration-list
```