컨트롤러 서버 UCS C240 M4 교체 - vEPC

목차

소개

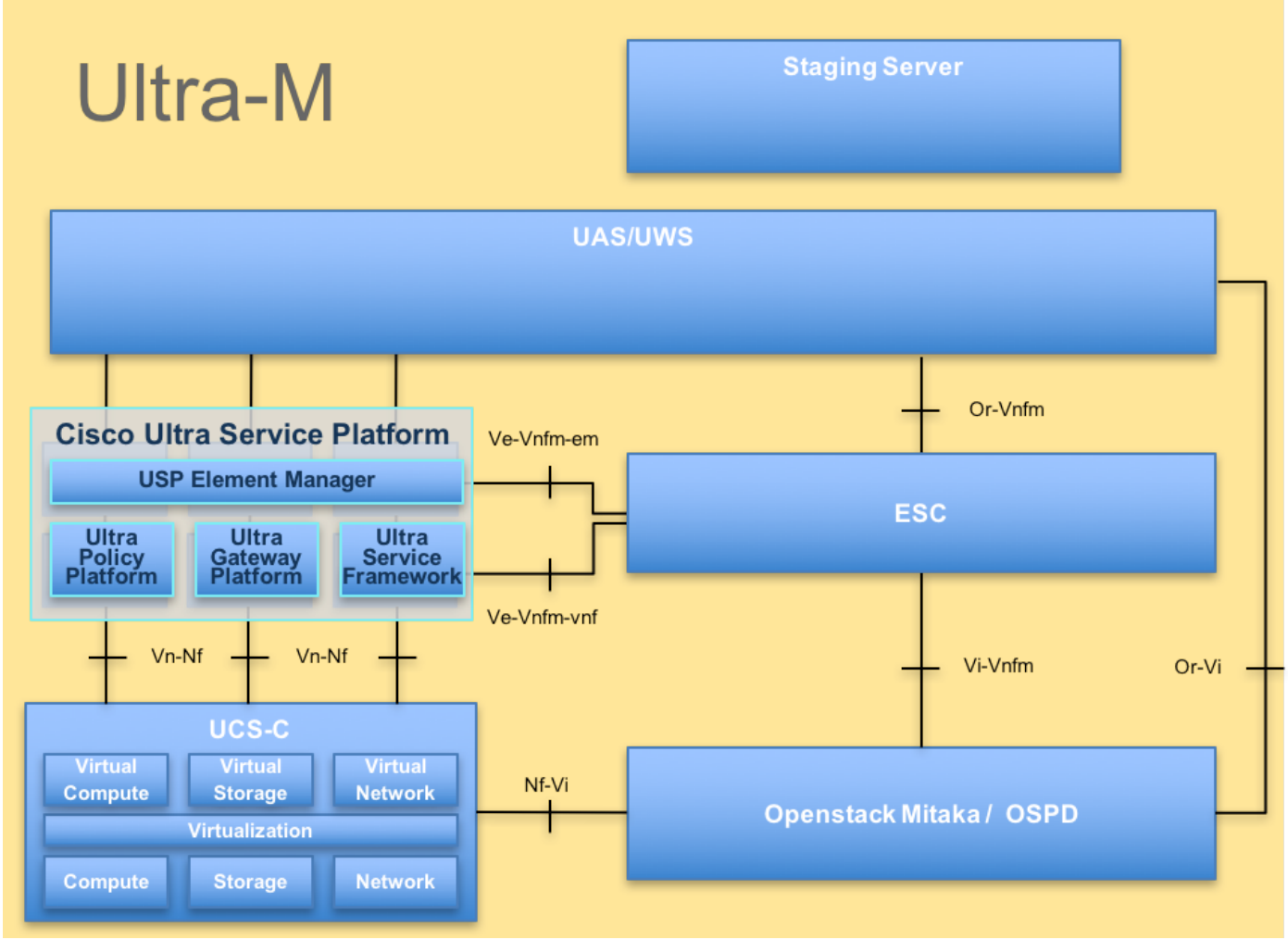
이 문서에서는 StarOS VNF(Virtual Network Functions)를 호스팅하는 Ultra-M 설정에서 결함 있는 컨트롤러 서버를 교체하는 데 필요한 단계에 대해 설명합니다.

배경 정보

Ultra-M은 VNF의 구축을 간소화하기 위해 설계된, 사전 패키징되고 검증된 가상화된 모바일 패킷 코어 솔루션입니다. OpenStack은 Ultra-M용 VIM(Virtualized Infrastructure Manager)이며 다음 노드 유형으로 구성됩니다.

- 컴퓨팅
- 개체 스토리지 디스크 - 컴퓨팅(OSD - 컴퓨팅)
- 컨트롤러
- OpenStack 플랫폼 - 디렉터(OSPD)

이 그림에는 Ultra-M의 고급 아키텍처와 관련 구성 요소가 나와 있습니다.



UltraM 아키텍처

이 문서는 Cisco Ultra-M 플랫폼에 익숙한 Cisco 직원을 대상으로 하며 컨트롤러 서버 교체 시 OpenStack 및 StarOS VNF 레벨에서 수행해야 하는 단계에 대해 자세히 설명합니다.

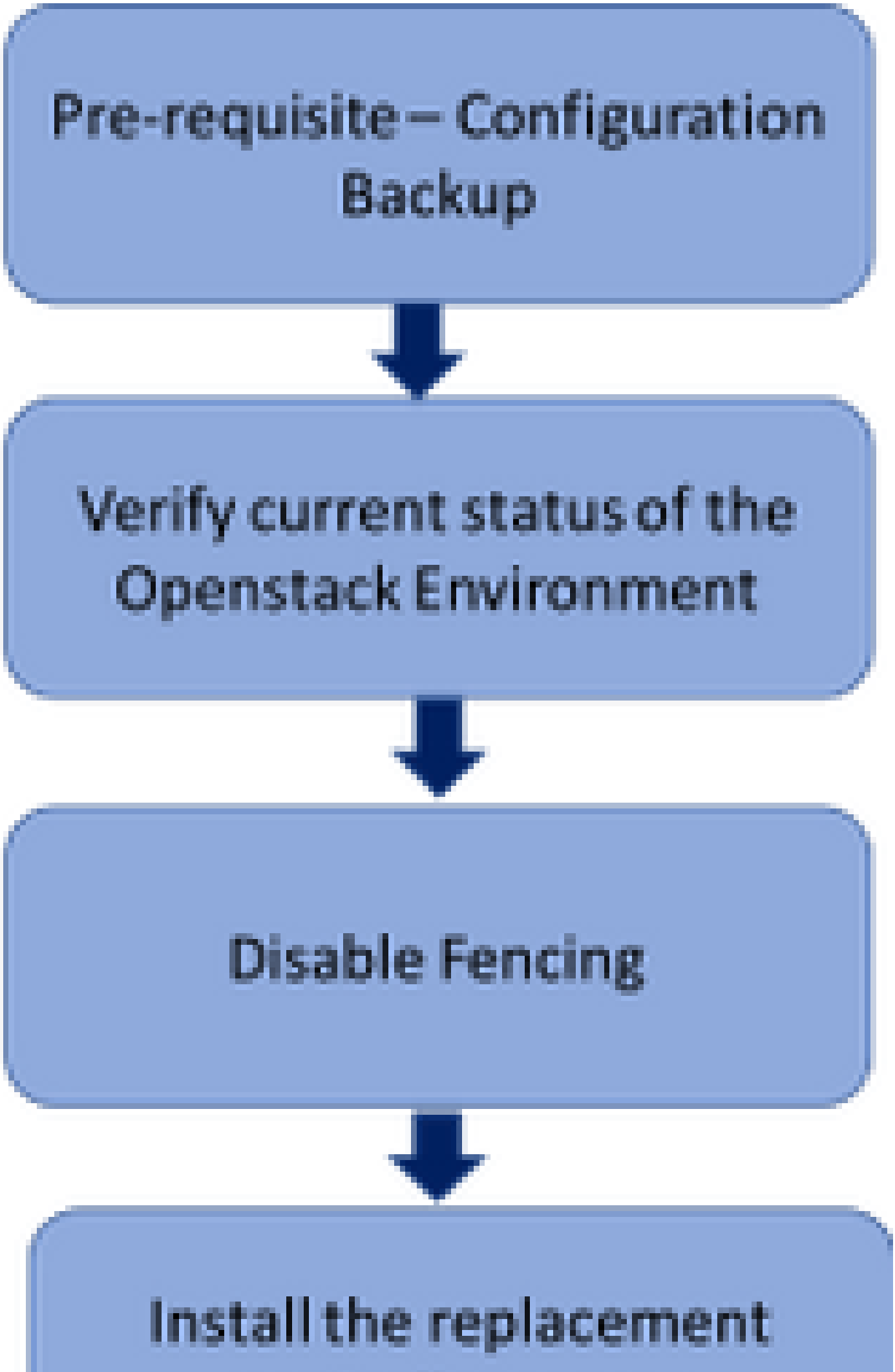
✎ 참고: 이 문서의 절차를 정의하기 위해 Ultra M 5.1.x 릴리스가 고려됩니다.

약어

VNF	가상 네트워크 기능
CF	제어 기능
SF	서비스 기능
Esc 키	Elastic Service Controller
자루걸레	절차 방법
OSD	개체 스토리지 디스크
HDD	하드 디스크 드라이브
SSD	SSD(Solid State Drive)
빔	가상 인프라 관리자
VM	가상 머신

엠	요소 관리자
UAS	Ultra Automation 서비스
UUID	보편적으로 고유한 식별자

MoP의 워크플로


Pre-requisite – Configuration Backup
Verify current status of the Openstack Environment
Disable Fencing
Install the replacement

```
[root@director ~]# tar --xattrs -czf undercloud-backup-`date +%F`.tar.gz /root/undercloud-all-databases
/etc/my.cnf.d/server.cnf /var/lib/glance/images /srv/node /home/stack
tar: Removing leading `/' from member names
```

예비 상태 확인

교체 절차를 진행하기 전에 OpenStack 환경 및 서비스의 현재 상태를 확인하고 정상적인지 확인하는 것이 중요합니다. 컨트롤러 교체 과정에서 발생하는 복잡성을 방지할 수 있습니다.

- OpenStack 및 노드 목록의 상태를 확인합니다.

```
[stack@director ~]$ source stackrc
[stack@director ~]$ openstack stack list --nested
[stack@director ~]$ ironic node-list
[stack@director ~]$ nova list
```

- 컨트롤러에서 Pacemaker 상태를 확인합니다.

활성 컨트롤러 중 하나에 로그인하여 페이스메이커 상태를 확인합니다. 모든 서비스는 사용 가능한 컨트롤러에서 실행되고 장애가 발생한 컨트롤러에서 중지되어야 합니다.

```
[stack@pod1-controller-0 ~]# pcs status

<snip>
Online: [ pod1-controller-0 pod1-controller-1 ]
OFFLINE: [ pod1-controller-2 ]
Full list of resources:
 ip-11.120.0.109 (ocf::heartbeat:IPaddr2): Started pod1-controller-0
 ip-172.25.22.109 (ocf::heartbeat:IPaddr2): Started pod1-controller-1
 ip-192.200.0.107 (ocf::heartbeat:IPaddr2): Started pod1-controller-0

 Clone Set: haproxy-clone [haproxy]
 Started: [ pod1-controller-0 pod1-controller-1 ]
 Stopped: [ pod1-controller-2 ]

 Master/Slave Set: galera-master [galera]
 Masters: [ pod1-controller-0 pod1-controller-1 ]
 Stopped: [ pod1-controller-2 ]
 ip-11.120.0.110 (ocf::heartbeat:IPaddr2): Started pod1-controller-0
 ip-11.119.0.110 (ocf::heartbeat:IPaddr2): Started pod1-controller-1

 Clone Set: rabbitmq-clone [rabbitmq]
 Started: [ pod1-controller-0 pod1-controller-1 ]
 Stopped: [ pod1-controller-2 ]

 Master/Slave Set: redis-master [redis]
 Masters: [ pod1-controller-0 ]
 Slaves: [ pod1-controller-1 ]
 Stopped: [ pod1-controller-2 ]
```

```
 ip-11.118.0.104 (ocf::heartbeat:IPaddr2): Started pod1-controller-1
 openstack-cinder-volume (systemd:openstack-cinder-volume): Started pod1-controller-0

 my-ipmilan-for-controller-6 (stonith:fence_ipmilan): Started pod1-controller-1
 my-ipmilan-for-controller-4 (stonith:fence_ipmilan): Started pod1-controller-0
 my-ipmilan-for-controller-7 (stonith:fence_ipmilan): Started pod1-controller-0

Failed Actions:
Daemon Status:

 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
```

이 예에서 Controller-2는 오프라인 상태입니다. 따라서 교체됩니다. Controller-0 및 Controller-1이 작동하며 클러스터 서비스를 실행하고 있습니다.

- 활성 컨트롤러에서 MariaDB 상태를 확인합니다.

<#root>

```
[stack@director] nova list | grep control
| 4361358a-922f-49b5-89d4-247a50722f6d | pod1-controller-0 | ACTIVE | - | Running | ctlplane=

192.200.0.102

 |
| d0f57f27-93a8-414f-b4d8-957de0d785fc | pod1-controller-1 | ACTIVE | - | Running | ctlplane=

192.200.0.110

 |

[stack@director ~]$

for i in 192.200.0.102 192.200.0.110 ; do echo "*** $i ***" ; ssh heat-admin@$i "sudo mysql --exec=\"SHO


*** 192.200.0.152 ***
Variable_name     Value

wsrep_local_state_comment  Synced


Variable_name     Value

wsrep_cluster_size         2


*** 192.200.0.154 ***
Variable_name     Value
wsrep_local_state_comment  Synced
Variable_name     Value
wsrep_cluster_size         2
```

각 활성 컨트롤러에 대해 다음 행이 있는지 확인합니다.

wsrep_local_state_comment: 동기화됨

wsrep_cluster_size: 2

- 활성 컨트롤러에서 Rabbitmq 상태를 확인합니다. 장애가 발생한 컨트롤러는 실행 중인 노드 목록에 나타나지 않아야 합니다.

<#root>

```
[heat-admin@pod1-controller-0 ~] sudo rabbitmqctl cluster_status
Cluster status of node 'rabbit@pod1-controller-0' ...
[{nodes,[{disc,['rabbit@pod1-controller-0','rabbit@pod1-controller-1',
               'rabbit@pod1-controller-2']}]},

{running_nodes,['rabbit@pod1-controller-1',



               'rabbit@pod1-controller-0']},


 {cluster_name,<<"rabbit@pod1-controller-2.localdomain">>},
 {partitions,[]},
 {alarms,[{'rabbit@pod1-controller-1',[]},
          {'rabbit@pod1-controller-0',[]}]}]


[heat-admin@pod1-controller-1 ~] sudo rabbitmqctl cluster_status
Cluster status of node 'rabbit@pod1-controller-1' ...
[{nodes,[{disc,['rabbit@pod1-controller-0','rabbit@pod1-controller-1',
               'rabbit@pod1-controller-2']}]},

 {running_nodes,['rabbit@pod1-controller-0',



               'rabbit@pod1-controller-1']},


 {cluster_name,<<"rabbit@pod1-controller-2.localdomain">>},
 {partitions,[]},
 {alarms,[{'rabbit@pod1-controller-0',[]},
          {'rabbit@pod1-controller-1',[]}]}]
```

- OSP-D 노드에서 모든 언더클라우드 서비스가 로드됨, 활성 및 실행 중인지 확인합니다.

```
[stack@director ~]$ systemctl list-units "openstack*" "neutron*" "openvswitch*"

UNIT                                    LOAD   ACTIVE SUB     DESCRIPTION

neutron-dhcp-agent.service              loaded active running OpenStack Neutron DHCP Agent
neutron-openvswitch-agent.service       loaded active running OpenStack Neutron Open vSwitch Agent
neutron-ovs-cleanup.service             loaded active exited  OpenStack Neutron Open vSwitch Cleanup
neutron-server.service                  loaded active running OpenStack Neutron Server
openstack-aodh-evaluator.service        loaded active running OpenStack Alarm evaluator service
openstack-aodh-listener.service         loaded active running OpenStack Alarm listener service
openstack-aodh-notifier.service         loaded active running OpenStack Alarm notifier service
```

```
openstack-ceilometer-central.service       loaded active running OpenStack ceilometer central agent
openstack-ceilometer-collector.service     loaded active running OpenStack ceilometer collection servic
openstack-ceilometer-notification.service  loaded active running OpenStack ceilometer notification agen
openstack-glance-api.service               loaded active running OpenStack Image Service (code-named Gl
openstack-glance-registry.service          loaded active running OpenStack Image Service (code-named Gl
openstack-heat-api-cfn.service             loaded active running Openstack Heat CFN-compatible API Serv
openstack-heat-api.service                 loaded active running OpenStack Heat API Service
openstack-heat-engine.service              loaded active running Openstack Heat Engine Service
openstack-ironic-api.service               loaded active running OpenStack Ironic API service
openstack-ironic-conductor.service         loaded active running OpenStack Ironic Conductor service
openstack-ironic-inspector-dnsmasq.service loaded active running PXE boot dnsmasq service for Ironic In
openstack-ironic-inspector.service         loaded active running Hardware introspection service for Ope
openstack-mistral-api.service              loaded active running Mistral API Server
openstack-mistral-engine.service           loaded active running Mistral Engine Server
openstack-mistral-executor.service         loaded active running Mistral Executor Server
openstack-nova-api.service                 loaded active running OpenStack Nova API Server
openstack-nova-cert.service                loaded active running OpenStack Nova Cert Server
openstack-nova-compute.service             loaded active running OpenStack Nova Compute Server
openstack-nova-conductor.service           loaded active running OpenStack Nova Conductor Server
openstack-nova-scheduler.service           loaded active running OpenStack Nova Scheduler Server
openstack-swift-account-reaper.service     loaded active running OpenStack Object Storage (swift) - Acc
openstack-swift-account.service            loaded active running OpenStack Object Storage (swift) - Acc
openstack-swift-container-updater.service  loaded active running OpenStack Object Storage (swift) - Con
openstack-swift-container.service          loaded active running OpenStack Object Storage (swift) - Con
openstack-swift-object-updater.service     loaded active running OpenStack Object Storage (swift) - Obj
openstack-swift-object.service             loaded active running OpenStack Object Storage (swift) - Obj
openstack-swift-proxy.service              loaded active running OpenStack Object Storage (swift) - Pro
openstack-zaqar.service                    loaded active running OpenStack Message Queuing Service (cod
openstack-zaqar@1.service                  loaded active running OpenStack Message Queuing Service (cod
openvswitch.service                        loaded active exited  Open vSwitch

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

37 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

컨트롤러 클러스터에서 펜싱 비활성화

<#root>

```
[root@pod1-controller-0 ~]# sudo pcs property set stonith-enabled=false
[root@pod1-controller-0 ~]# pcs property show

Cluster Properties:
 cluster-infrastructure: corosync
 cluster-name: tripleo_cluster
 dc-version: 1.1.15-11.el7_3.4-e174ec8
 have-watchdog: false
 last-lrm-refresh: 1510809585
 maintenance-mode: false
 redis_REPL_INFO: pod1-controller-0
```

stonith-enabled: false

```
Node Attributes:
 pod1-controller-0: rmq-node-attr-last-known-rabbitmq=rabbit@pod1-controller-0
 pod1-controller-1: rmq-node-attr-last-known-rabbitmq=rabbit@pod1-controller-1
 pod1-controller-2: rmq-node-attr-last-known-rabbitmq=rabbit@pod1-controller-2
```

새 컨트롤러 노드 설치

- 새 UCS C240 M4 서버를 설치하는 단계 및 초기 설정 단계는 다음에서 참조할 수 있습니다.

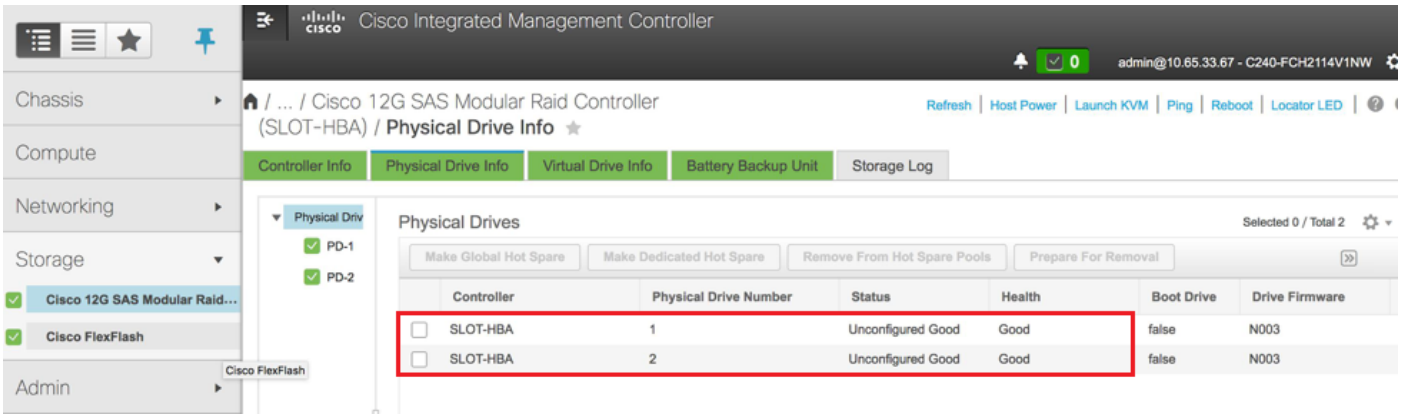
[Cisco UCS C240 M4 서버 설치 및 서비스 가이드]

- CIMC IP를 사용하여 서버에 로그인합니다
- 펌웨어가 이전에 사용한 권장 버전에 따라 다르면 BIOS 업그레이드를 수행합니다. BIOS 업그레이드 단계는 다음과 같습니다.

[Cisco UCS C-Series 랙 마운트 서버 BIOS 업그레이드 가이드]

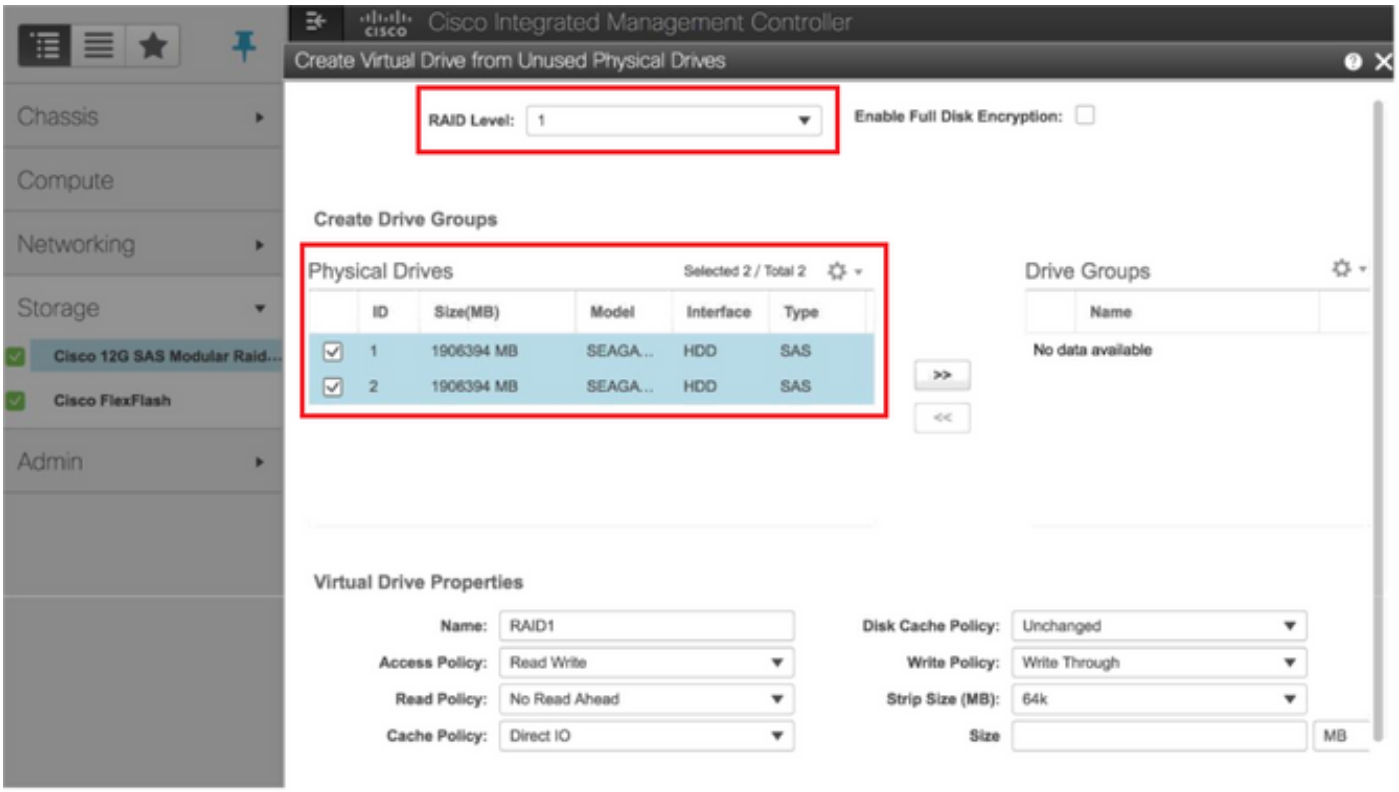
- 물리적 드라이브의 상태를 확인합니다. "Unconfigured Good(구성되지 않음, 정상)"이어야 합니다.

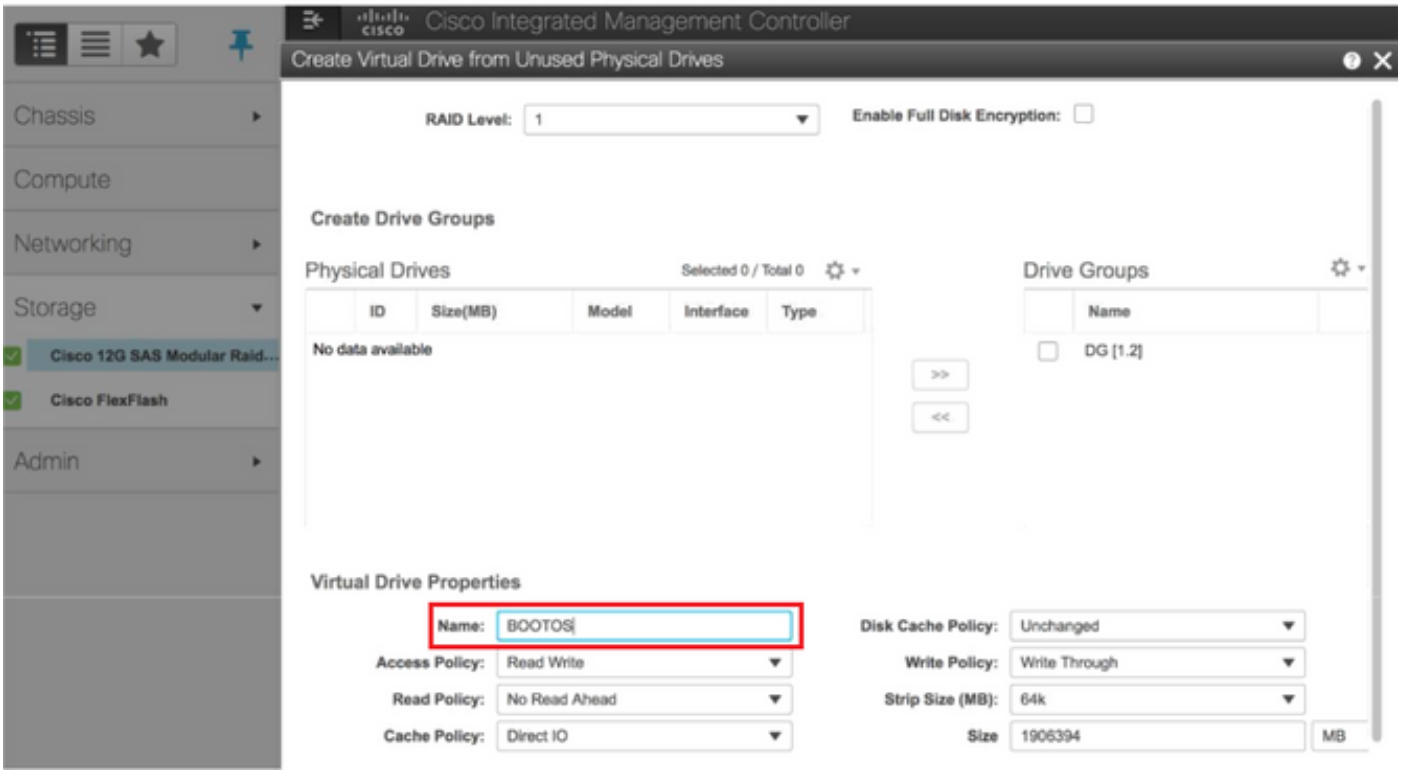
스토리지 > Cisco 12G SAS 모듈형 Raid 컨트롤러(SLOT-HBA) > 물리적 드라이브 정보



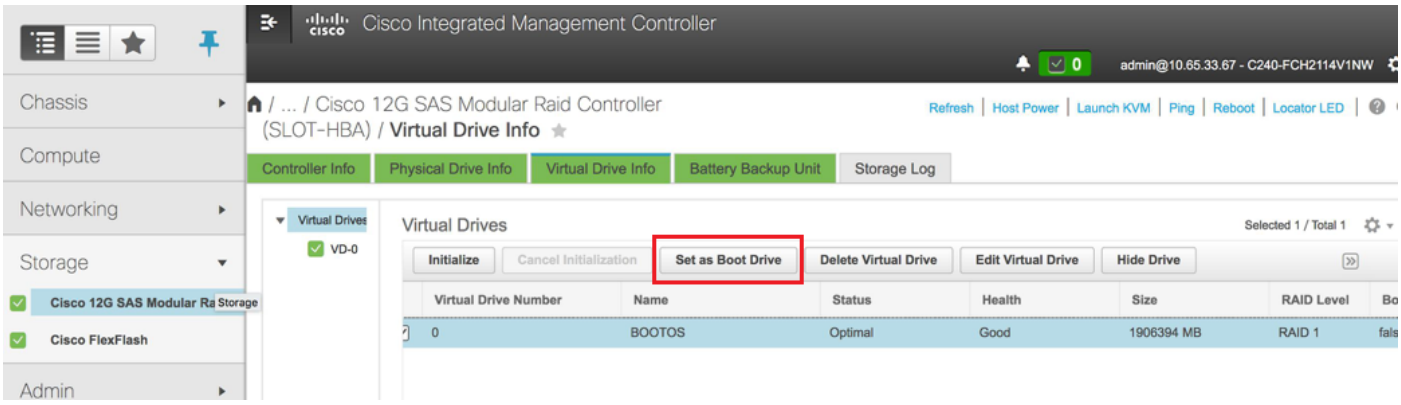
- RAID 레벨 1의 물리적 드라이브에서 가상 드라이브를 생성합니다.

스토리지 > Cisco 12G SAS 모듈형 Raid 컨트롤러(SLOT-HBA) > 컨트롤러 정보 > 사용되지 않은 물리적 드라이브에서 가상 드라이브 생성



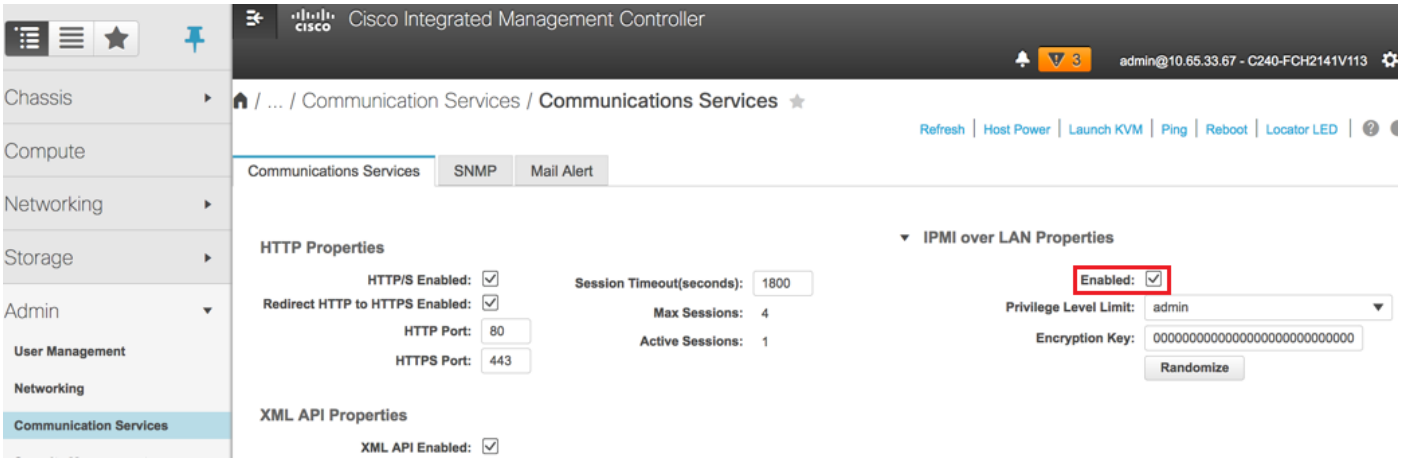


- VD를 선택하고 Set as Boot Drive를 구성합니다.



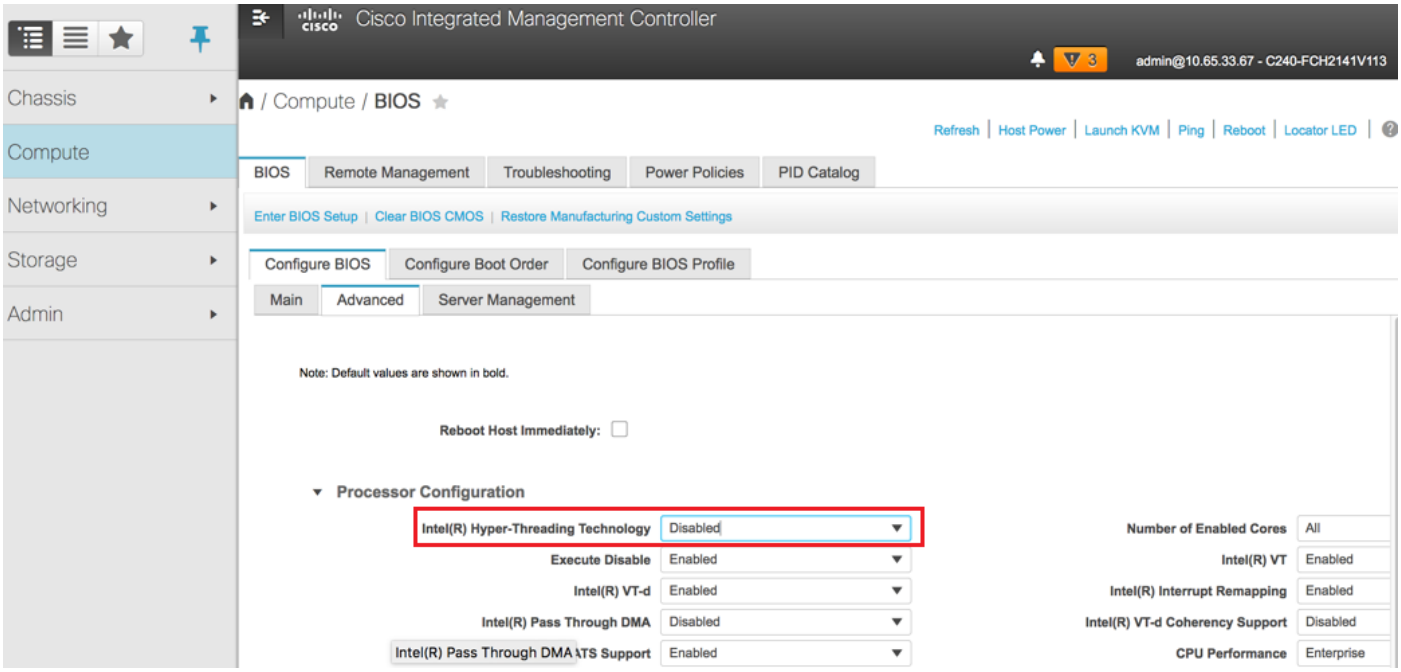
- IPMI over LAN 활성화:

Admin(관리) > Communication Services(통신 서비스) > Communication Services(통신 서비스)



- 하이퍼스레딩을 비활성화합니다.

Compute(컴퓨팅) > BIOS > Configure BIOS > Advanced(고급) > Processor Configuration(프로세서 컨피그레이션)



참고: 여기에 표시된 이미지 및 이 섹션에서 설명한 컨피그레이션 단계는 펌웨어 버전 3.0(3e)을 참조하며, 다른 버전에서 작업하는 경우 약간의 차이가 있을 수 있습니다.

오버클라우드의 컨트롤러 노드 교체

이 섹션에서는 결함이 있는 컨트롤러를 오버클라우드의 새 컨트롤러로 교체하는 데 필요한 단계를 다룹니다. 이 경우 스택을 불러오는 데 사용된 deploy.sh 스크립트가 다시 사용됩니다. 배포 시 ControllerNodesPostDeployment 단계에서 Puppet 모듈의 일부 제한으로 인해 업데이트가 실패합니다. 구축 스크립트를 재시작하기 전에 수동 개입이 필요합니다.

실패한 컨트롤러 노드 제거 준비

- 실패한 컨트롤러의 인덱스를 식별합니다. 인덱스는 OpenStack 서버 목록 출력의 컨트롤러 이름에 대한 숫자 접미사입니다. 이 예에서 인덱스는 2입니다.

<#root>

```
[stack@director ~]$ nova list | grep controller
| 5813a47e-af27-4fb9-8560-75decd3347b4 | pod1-controller-0  | ACTIVE | -          | Running     | ctlpl
| 457f023f-d077-45c9-bbea-dd32017d9708 | pod1-controller-1  | ACTIVE | -          | Running     | ctlpl
| d13bb207-473a-4e42-a1e7-05316935ed65 |
```

pod1-controller-2

```
 | ACTIVE | -          | Running     | ctlplane=192.200.0.151 |
```

- 삭제할 노드를 정의하는 Yaml 파일~templates/remove-controller.yaml을 만듭니다. 자원 목록의 항목에 대해 이전 단계에서 찾은 인덱스를 사용합니다.

```
[stack@director ~]$ cat templates/remove-controller.yaml

parameters:
  ControllerRemovalPolicies:
    [{'resource_list': ['2']}]

parameter_defaults:
  CorosyncSettleTries: 5
```

- 오버클라우드를 설치하기 위해 사용되는 배포 스크립트의 복사본을 만들고 이전에 만든 remove-controller.yaml 파일을 포함하기 위해 행을 삽입합니다.

```
[stack@director ~]$ cp deploy.sh deploy-removeController.sh
[stack@director ~]$ cat deploy-removeController.sh
time openstack overcloud deploy --templates \
-r ~/custom-templates/custom-roles.yaml \
```

```
-e /home/stack/templates/remove-controller.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/puppet-pacemaker.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/storage-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/neutron-sriov.yaml \
-e ~/custom-templates/network.yaml \
-e ~/custom-templates/ceph.yaml \
-e ~/custom-templates/compute.yaml \
-e ~/custom-templates/layout-removeController.yaml \
-e ~/custom-templates/rabbitmq.yaml \
--stack pod1 \
--debug \
--log-file overcloudDeploy_$(date +%m_%d_%y__%H_%M_%S).log \
--neutron-flat-networks phys_pcie1_0,phys_pcie1_1,phys_pcie4_0,phys_pcie4_1 \
--neutron-network-vlan-ranges datacentre:101:200 \
--neutron-disable-tunneling \
--verbose --timeout 180
```

- 여기에 설명된 명령을 사용하여 교체할 컨트롤러의 ID를 식별하고 유지 보수 모드로 이동합니다.

<#root>

```
[stack@director ~]$ nova list | grep controller

| 5813a47e-af27-4fb9-8560-75decd3347b4 | pod1-controller-0  | ACTIVE | -          | Running     | ctlpl

| 457f023f-d077-45c9-bbea-dd32017d9708 | pod1-controller-1  | ACTIVE | -          | Running     | ctlpl

|

d13bb207-473a-4e42-a1e7-05316935ed65

 | pod1-controller-2  | ACTIVE | -          | Running     | ctlplane=192.200.0.151 |


[stack@director ~]$ openstack baremetal node list | grep

d13bb207-473a-4e42-a1e7-05316935ed65



|

e7c32170-c7d1-4023-b356-e98564a9b85b

 | None | d13bb207-473a-4e42-a1e7-05316935ed65 | power off   | active             | False       |


[stack@b10-ospd ~]$ openstack baremetal node maintenance set

e7c32170-c7d1-4023-b356-e98564a9b85b



[stack@director~]$ openstack baremetal node list | grep True

| e7c32170-c7d1-4023-b356-e98564a9b85b | None | d13bb207-473a-4e42-a1e7-05316935ed65 | power off   | ac

True
```

|

- 교체 절차 시 DB가 실행되도록 하려면 페이스메이커 컨트롤에서 Galera를 제거하고 활성 컨트롤러 중 하나에서 이 명령을 실행합니다.

<#root>

```
[root@pod1-controller-0 ~]# sudo pcs resource unmanage galera
[root@pod1-controller-0 ~]# sudo pcs status

Cluster name: tripleo_cluster
Stack: corosync
Current DC: pod1-controller-0 (version 1.1.15-11.el7_3.4-e174ec8) - partition with quorum
Last updated: Thu Nov 16 16:51:18 2017                    Last change: Thu Nov 16 16:51:12 2017 by root
3 nodes and 22 resources configured
Online: [ pod1-controller-0 pod1-controller-1 ]
OFFLINE: [ pod1-controller-2 ]

Full list of resources:

 ip-11.120.0.109          (ocf::heartbeat:IPaddr2):        Started pod1-controller-0
 ip-172.25.22.109        (ocf::heartbeat:IPaddr2):        Started pod1-controller-1
 ip-192.200.0.107        (ocf::heartbeat:IPaddr2):        Started pod1-controller-0

 Clone Set: haproxy-clone [haproxy]
     Started: [ pod1-controller-0 pod1-controller-1 ]
     Stopped: [ pod1-controller-2 ]


Master/Slave Set: galera-master [galera] (unmanaged)


     galera         (ocf::heartbeat:galera):          Master pod1-controller-0 (unmanaged)
     galera         (ocf::heartbeat:galera):          Master pod1-controller-1 (unmanaged)

     Stopped: [ pod1-controller-2 ]
 ip-11.120.0.110          (ocf::heartbeat:IPaddr2):        Started pod1-controller-0
 ip-11.119.0.110          (ocf::heartbeat:IPaddr2):        Started pod1-controller-1

<snip>
```

새 컨트롤러 노드 추가 준비

- 새 컨트롤러 세부 정보만으로 controllerRMA.json 파일을 만듭니다. 새 컨트롤러의 인덱스 번호가 이전에 사용되지 않았는지 확인합니다. 일반적으로 다음으로 높은 컨트롤러 번호로 증가합니다.

예: 가장 높은 우선 순위는 Controller-2였으므로 Controller-3을 생성하십시오.

참고: json 형식에 유의하십시오.

```
[stack@director ~]$ cat controllerRMA.json
{
    "nodes": [
        {
            "mac": [
                <MAC_ADDRESS>
            ],
            "capabilities": "node:controller-3,boot_option:local",
            "cpu": "24",
            "memory": "256000",
            "disk": "3000",
            "arch": "x86_64",
            "pm_type": "pxe_ipmitool",
            "pm_user": "admin",
            "pm_password": "<PASSWORD>",
            "pm_addr": "<CIMC_IP>"
        }
    ]
}
```

- 이전 단계에서 생성한 json 파일을 사용하여 새 노드를 가져옵니다.

<#root>

```
[stack@director ~]$ openstack baremetal import --json controllerRMA.json

Started Mistral Workflow. Execution ID: 67989c8b-1225-48fe-ba52-3a45f366e7a0

Successfully registered node UUID

048ccb59-89df-4f40-82f5-3d90d37ac7dd


Started Mistral Workflow. Execution ID: c6711b5f-fa97-4c86-8de5-b6bc7013b398

Successfully set all nodes to available.

[stack@director ~]$ openstack baremetal node list | grep available

|

048ccb59-89df-4f40-82f5-3d90d37ac7dd

 | None | None                                 | power off   | available          | False
```

- 상태를 관리할 노드를 설정합니다.

<#root>

```
[stack@director ~]$ openstack baremetal node manage 048ccb59-89df-4f40-82f5-3d90d37ac7dd
[stack@director ~]$ openstack baremetal node list | grep off
| 048ccb59-89df-4f40-82f5-3d90d37ac7dd | None | None | power off |

manageable
```

```
 | False |
```

- 자체 검사 실행:

<#root>

```
[stack@director ~]$
```

openstack overcloud node introspect

048ccb59-89df-4f40-82f5-3d90d37ac7dd --provide

```
Started Mistral Workflow. Execution ID: f73fb275-c90e-45cc-952b-bfc25b9b5727
Waiting for introspection to finish...
Successfully introspected all nodes.
Introspection completed.
Started Mistral Workflow. Execution ID: a892b456-eb15-4c06-b37e-5bc3f6c37c65
Successfully set all nodes to available

[stack@director ~]$
```

openstack baremetal node list | grep available

```
| 048ccb59-89df-4f40-82f5-3d90d37ac7dd | None | None                                    | power off   | av
```

- 사용 가능한 노드를 새 컨트롤러 속성으로 표시합니다. controllerRMA.json 파일에 사용된 대로 새 컨트롤러에 지정된 컨트롤러 ID를 사용하십시오.

<#root>

```
[stack@director ~]$ openstack baremetal node set --property capabilities='node:
```

controller-3

```
,profile:control,boot_option:local' 048ccb59-89df-4f40-82f5-3d90d37ac7dd
```

- 구축 스크립트에는 layout.yaml이라는 사용자 정의 템플릿이 있습니다.yaml은 여러 인터페이스의 컨트롤러에 할당되는 IP 주소를 지정합니다. 새 스택에는 Controller-0, Controller-1 및 Controller-2에 대해 3개의 주소가 정의되어 있습니다. 새 컨트롤러를 추가할 때 각 서브넷에 대해 다음 IP 주소를 순서대로 추가해야 합니다.

<#root>

```
ControllerIPs:
 internal_api:
 - 11.120.0.10
 - 11.120.0.11
 - 11.120.0.12
```

- 11.120.0.13

```
 tenant:
 - 11.117.0.10
 - 11.117.0.11
 - 11.117.0.12
 - 11.117.0.13

 storage:
 - 11.118.0.10
 - 11.118.0.11
 - 11.118.0.12
 - 11.118.0.13

 storage_mgmt:
 - 11.119.0.10
 - 11.119.0.11
 - 11.119.0.12
 - 11.119.0.13
```

- 이제 이전 노드를 제거하고 새 노드를 추가하기 위해 이전에 만든 deploy-removecontroller.sh를 실행합니다.

참고: 이 단계는 ControllerNodesDeployment_Step1에서 실패할 것으로 예상됩니다. 이 시점에서 수동 개입이 필요합니다.

<#root>

```
[stack@b10-ospd ~]$
./deploy-addController.sh

START with options: [u'overcloud', u'deploy', u'--templates', u'-r', u'/home/stack/custom-templates/cus
  :
DeploymentError: Heat Stack update failed
END return value: 1

real     42m1.525s
user     0m3.043s
sys      0m0.614s
```

다음 명령을 사용하여 구축의 진행/상태를 모니터링할 수 있습니다.

```
[stack@director~]$ openstack stack list --nested | grep -iv complete

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

```
| ID                                   | Stack Name

+--------------------------------------+-----------------------------------------------------------------

| c1e338f2-877e-4817-93b4-9a3f0c0b3d37 | pod1-AllNodesDeploySteps-5psegydpwxij-ComputeDeployment_Step1-

| 1db4fef4-45d3-4125-bd96-2cc3297a69ff | pod1-AllNodesDeploySteps-5psegydpwxij-ControllerDeployment_Ste

| e90f00ef-2499-4ec3-90b4-d7def6e97c47 | pod1-AllNodesDeploySteps-5psegydpwxij

| 6c4b604a-55a4-4a19-9141-28c844816c0d | pod1

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

수동 개입

- OSP-D 서버에서 OpenStack server list 명령을 실행하여 사용 가능한 컨트롤러를 나열합니다 . 새로 추가된 컨트롤러가 목록에 나타나야 합니다.

<#root>

```
[stack@director ~]$ openstack server list | grep controller
| 3e6c3db8-ba24-48d9-b0e8-1e8a2eb8b5ff |
```

pod1-controller-3

```
  | ACTIVE | ctlplane=192.200.0.103 | overcloud-full |
| 457f023f-d077-45c9-bbea-dd32017d9708 | pod1-controller-1  | ACTIVE | ctlplane=192.200.0.154 | overclo
| 5813a47e-af27-4fb9-8560-75decd3347b4 | pod1-controller-0  | ACTIVE | ctlplane=192.200.0.152 | overclo
```

- 새로 추가된 컨트롤러가 아닌 활성 컨트롤러 중 하나에 연결하고 /etc/corosync/corosycn.conf 파일을 확인합니다. 각 컨트롤러에 nodeid를 할당하는 nodelist를 찾습니다. 실패 한 노드에 대한 항목을 찾아 해당 노드 ID를 기록 합니다.

<#root>

```
[root@pod1-controller-0 ~]# cat /etc/corosync/corosync.conf
totem {
    version: 2
    secauth: off
    cluster_name: tripleo_cluster
    transport: udpu
    token: 10000
}

nodelist {
    node {
        ring0_addr: pod1-controller-0
        nodeid: 5
    }
    node {
        ring0_addr: pod1-controller-1
        nodeid: 7
```

```
    }
    node {
        ring0_addr: pod1-controller-2

nodeid: 8

    }
}
```

- 각 활성 컨트롤러에 로그인합니다. 장애가 발생한 노드를 제거하고 서비스를 다시 시작합니다. 이 경우 pod1-controller-2를 제거합니다. 새로 추가된 컨트롤러에서는 이 작업을 수행하지 마십시오.

```
[root@pod1-controller-0 ~]# sudo pcs cluster localnode remove pod1-controller-2
pod1-controller-2: successfully removed!
[root@pod1-controller-0 ~]# sudo pcs cluster reload corosync
Corosync reloaded

[root@pod1-controller-1 ~]# sudo pcs cluster localnode remove pod1-controller-2
pod1-controller-2: successfully removed!
[root@pod1-controller-1 ~]# sudo pcs cluster reload corosync
Corosync reloaded
```

- 클러스터에서 장애가 발생한 노드를 삭제하려면 활성 컨트롤러 중 하나에서 이 명령을 실행합니다.

<#root>

```
[root@pod1-controller-0 ~]# sudo

crm_node -R pod1-controller-2 --force
```

- rabbitmq 클러스터에서 장애가 발생한 노드를 삭제하려면 활성 컨트롤러 중 하나에서 이 명령을 실행합니다.

<#root>

```
[root@pod1-controller-0 ~]#

sudo rabbitmqctl forget_cluster_node rabbit@pod1-controller-2

Removing node 'rabbit@newtonoc-controller-2' from cluster ...
```

- MongoDB에서 실패 한 노드를 삭제 합니다. 이렇게 하려면 활성 Mongo 노드를 찾아야 합니다. netstat를 사용하여 호스트의 IP 주소를 찾습니다.

<#root>

```
[root@pod1-controller-0 ~]#
```

sudo netstat -tulnp | grep 27017

```
tcp        0      0 11.120.0.10:27017       0.0.0.0:*               LISTEN      219577/mongod
```

- 노드에 로그인하여 이전 명령의 IP 주소 및 포트 번호를 사용하는 마스터인지 확인합니다.

<#root>

```
[heat-admin@pod1-controller-0 ~]$
```

echo "db.isMaster()" | mongo --host

11.120.0.10:27017

```
MongoDB shell version: 2.6.11
connecting to: 11.120.0.10:27017/test
{
         "setName" : "tripleo",
         "setVersion" : 9,
```

"ismaster" : true,

```
         "secondary" : false,
         "hosts" : [
                  "11.120.0.10:27017",
                  "11.120.0.12:27017",
                  "11.120.0.11:27017"
         ],
         "primary" : "11.120.0.10:27017",
         "me" : "11.120.0.10:27017",
         "electionId" : ObjectId("5a0d2661218cb0238b582fb1"),
         "maxBsonObjectSize" : 16777216,
         "maxMessageSizeBytes" : 48000000,
         "maxWriteBatchSize" : 1000,
         "localTime" : ISODate("2017-11-16T18:36:34.473Z"),
         "maxWireVersion" : 2,
         "minWireVersion" : 0,
         "ok" : 1
}
```

노드가 마스터가 아닌 경우 다른 활성 컨트롤러에 로그인하고 동일한 단계를 수행합니다.

- 마스터에서 rs.status() 명령을 사용하여 사용 가능한 노드를 나열합니다. 이전/응답하지 않는 노드를 찾아 mongo 노드 이름을 확인합니다.

<#root>

```
[root@pod1-controller-0 ~]#
```

mongo --host 11.120.0.10

```
MongoDB shell version: 2.6.11
connecting to: 11.120.0.10:27017/test
<snip>
tripleo:PRIMARY>
```

rs.status()

```
{
        "set" : "tripleo",
        "date" : ISODate("2017-11-14T13:27:14Z"),
        "myState" : 1,
        "members" : [
                {
                        "_id" : 0,
                        "name" : "11.120.0.10:27017",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 418347,
                        "optime" : Timestamp(1510666033, 1),
                        "optimeDate" : ISODate("2017-11-14T13:27:13Z"),
                        "electionTime" : Timestamp(1510247693, 1),
                        "electionDate" : ISODate("2017-11-09T17:14:53Z"),
                        "self" : true
                },
                {
                        "_id" : 2,
                        "name" : "11.120.0.12:27017",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 418347,
                        "optime" : Timestamp(1510666033, 1),
                        "optimeDate" : ISODate("2017-11-14T13:27:13Z"),
                        "lastHeartbeat" : ISODate("2017-11-14T13:27:13Z"),
                        "lastHeartbeatRecv" : ISODate("2017-11-14T13:27:13Z"),
                        "pingMs" : 0,
                        "syncingTo" : "11.120.0.10:27017"
                },
                {
```

"_id" : 3,

```
                        "name" : "11.120.0.11:27017
                        "health" : 0,
                        "state" : 8,
```

"stateStr" : "(not reachable/healthy)",

```
                        "uptime" : 0,
                        "optime" : Timestamp(1510610580, 1),
                        "optimeDate" : ISODate("2017-11-13T22:03:00Z"),
                        "lastHeartbeat" : ISODate("2017-11-14T13:27:10Z"),
                        "lastHeartbeatRecv" : ISODate("2017-11-13T22:03:01Z"),
                        "pingMs" : 0,
                        "syncingTo" : "11.120.0.10:27017"
                }
        ],
        "ok" : 1
}
```

- 마스터에서 rs.remove 명령을 사용하여 실패한 노드를 삭제합니다. 이 명령을 실행하면 일부 오류가 표시되지만, 노드가 제거되었는지 확인하려면 상태를 한 번 더 확인합니다.

<#root>

```
[root@pod1-controller-0 ~]$
```

mongo --host 11.120.0.10

```
<snip>
tripleo:PRIMARY>
```

rs.remove(

'11.120.0.12:27017')

```
2017-11-16T18:41:04.999+0000 DBClientCursor::init call() failed
2017-11-16T18:41:05.000+0000 Error: error doing query: failed at src/mongo/shell/query.js:81
2017-11-16T18:41:05.001+0000 trying reconnect to 11.120.0.10:27017 (11.120.0.10) failed
2017-11-16T18:41:05.003+0000 reconnect 11.120.0.10:27017 (11.120.0.10) ok

tripleo:PRIMARY>
```

rs.status()

```
{
        "set" : "tripleo",
        "date" : ISODate("2017-11-16T18:44:11Z"),
        "myState" : 1,
        "members" : [
                {
                        "_id" : 3,
                        "name" : "11.120.0.11:27017",
                        "health" : 1,
                        "state" : 2,
                        "stateStr" : "SECONDARY",
                        "uptime" : 187,
                        "optime" : Timestamp(1510857848, 3),
                        "optimeDate" : ISODate("2017-11-16T18:44:08Z"),
                        "lastHeartbeat" : ISODate("2017-11-16T18:44:11Z"),
                        "lastHeartbeatRecv" : ISODate("2017-11-16T18:44:09Z"),
                        "pingMs" : 0,
                        "syncingTo" : "11.120.0.10:27017"
                },
```

```
                {
                        "_id" : 4,
                        "name" : "11.120.0.10:27017",
                        "health" : 1,
                        "state" : 1,
                        "stateStr" : "PRIMARY",
                        "uptime" : 89820,
                        "optime" : Timestamp(1510857848, 3),
                        "optimeDate" : ISODate("2017-11-16T18:44:08Z"),
                        "electionTime" : Timestamp(1510811232, 1),
                        "electionDate" : ISODate("2017-11-16T05:47:12Z"),
                        "self" : true
                }
        ],
        "ok" : 1
}
tripleo:PRIMARY> exit
bye
```

- 활성 컨트롤러 노드 목록을 업데이트하려면 이 명령을 실행합니다. 이 목록에 새 컨트롤러 노드 포함:

<#root>

```
[root@pod1-controller-0 ~]#
```

sudo pcs resource update galera wsrep_cluster_address=gcomm://pod1-controller-0,pod1-controller-1,pod1-c

- 이미 있는 컨트롤러에서 새 컨트롤러로 이러한 파일을 복사합니다.

  /etc/sysconfig/clustercheck

  /root/.my.cnf

```
On existing controller:

[root@pod1-controller-0 ~]# scp /etc/sysconfig/clustercheck stack@192.200.0.1:/tmp/.
[root@pod1-controller-0 ~]# scp /root/.my.cnf stack@192.200.0.1:/tmp/my.cnf

On new controller:

[root@pod1-controller-3 ~]# cd /etc/sysconfig

[root@pod1-controller-3 sysconfig]# scp stack@192.200.0.1:/tmp/clustercheck .

[root@pod1-controller-3 sysconfig]# cd /root

[root@pod1-controller-3 ~]# scp stack@192.200.0.1:/tmp/my.cnf .my.cnf
```

- 이미 있는 컨트롤러 중 하나에서 cluster node add 명령을 실행합니다.

```
[root@pod1-controller-1 ~]# sudo pcs cluster node add pod1-controller-3

Disabling SBD service...
pod1-controller-3: sbd disabled
pod1-controller-0: Corosync updated
pod1-controller-1: Corosync updated

Setting up corosync...
pod1-controller-3: Succeeded
Synchronizing pcsd certificates on nodes pod1-controller-3...
pod1-controller-3: Success

Restarting pcsd on the nodes in order to reload the certificates...
pod1-controller-3: Success
```

- 각 컨트롤러에 로그인하고 /etc/corosync/corosync.conf 파일을 확인합니다. 새 컨트롤러가 나열되어 있고 해당 컨트롤러에 할당된 노드 ID가 이전에 사용되지 않은 시퀀스의 다음 번호인지 확인합니다. 다음 세 컨트롤러 모두에 대해 이 변경이 수행되었는지 확인합니다.

<#root>

```
[root@pod1-controller-1 ~]# cat /etc/corosync/corosync.conf
totem {
    version: 2
    secauth: off
    cluster_name: tripleo_cluster
    transport: udpu
    token: 10000
}
nodelist {
    node {
        ring0_addr: pod1-controller-0
        nodeid: 5
    }
    node {
        ring0_addr: pod1-controller-1
        nodeid: 7
    }
    node {
        ring0_addr: pod1-controller-3

 nodeid: 6


    }
}
quorum {
    provider: corosync_votequorum
}
logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
}
```

예: 수정 후 /etc/corosync/corosync.conf

<#root>

```
totem {
    version: 2
    secauth: off
    cluster_name: tripleo_cluster
    transport: udpu
    token: 10000
}
nodelist {
    node {
        ring0_addr: pod1-controller-0
        nodeid: 5
    }
    node {
        ring0_addr: pod1-controller-1
        nodeid: 7
    }
    node {
        ring0_addr: pod1-controller-3

nodeid: 9


    }
}
quorum {
    provider: corosync_votequorum
}
logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
}
```

- 활성 컨트롤러에서 corosync를 다시 시작합니다. 새 컨트롤러에서 corosync를 시작하지 마십시오.

<#root>

```
[root@pod1-controller-0 ~]#

sudo pcs cluster reload corosync


[root@pod1-controller-1 ~]#

sudo pcs cluster reload corosync
```

- 작동 중인 컨트롤러 중 하나에서 새 컨트롤러 노드를 시작합니다.

```
[root@pod1-controller-1 ~]# sudo pcs cluster start pod1-controller-3
```

- Galera를 다시 시작 하는 컨트롤러 중 하나에서 :

```
[root@pod1-controller-1 ~]#  sudo pcs cluster start pod1-controller-3

pod1-controller-0: Starting Cluster...

[root@pod1-controller-1 ~]# sudo pcs resource cleanup galera
Cleaning up galera:0 on pod1-controller-0, removing fail-count-galera
Cleaning up galera:0 on pod1-controller-1, removing fail-count-galera
Cleaning up galera:0 on pod1-controller-3, removing fail-count-galera
  * The configuration prevents the cluster from stopping or starting 'galera-master' (unmanaged)

Waiting for 3 replies from the CRMd... OK

[root@pod1-controller-1 ~]#
[root@pod1-controller-1 ~]# sudo pcs resource manage galera
```

- 클러스터가 유지 관리 모드에 있습니다. 서비스를 시작하려면 유지 보수 모드를 비활성화하십시오.

<#root>

```
[root@pod1-controller-2 ~]#
```

sudo pcs property set maintenance-mode=false --wait

- Galera에서 3개의 컨트롤러 모두 마스터로 나열될 때까지 Galera에 대한 PC 상태를 확인합니다.

✎ 참고: 대규모 설정의 경우 DB를 동기화하는 데 시간이 걸릴 수 있습니다.

```
[root@pod1-controller-1 ~]# sudo pcs status | grep galera -A1

Master/Slave Set: galera-master [galera]
      Masters: [ pod1-controller-0 pod1-controller-1 pod1-controller-3 ]
```

- 클러스터를 유지 관리 모드로 전환합니다.

<#root>

```
[root@pod1-controller-1~]#
```

sudo pcs property set maintenance-mode=true --wait

```
[root@pod1-controller-1 ~]# pcs cluster status
Cluster Status:
 Stack: corosync
 Current DC: pod1-controller-0 (version 1.1.15-11.el7_3.4-e174ec8) - partition with quorum
 Last updated: Thu Nov 16 19:17:01 2017                  Last change: Thu Nov 16 19:16:48 2017 by root
               *** Resource management is DISABLED ***
   The cluster will not attempt to start, stop or recover services

PCSD Status:
  pod1-controller-3: Online
  pod1-controller-0: Online
  pod1-controller-1: Online
```

- 이전에 실행한 배포 스크립트를 다시 실행합니다. 이번에는 성공해야 합니다.

<#root>

```
[stack@director ~]$
```

./deploy-addController.sh

```
START with options: [u'overcloud', u'deploy', u'--templates', u'-r', u'/home/stack/custom-templates/cus
options: Namespace(access_key='', access_secret='***', access_token='***', access_token_endpoint='', ac
Auth plugin password selected

Starting new HTTPS connection (1): 192.200.0.2
"POST /v2/action_executions HTTP/1.1" 201 1696
HTTP POST https://192.200.0.2:13989/v2/action_executions 201
Overcloud Endpoint: http://172.25.22.109:5000/v2.0
Overcloud Deployed
clean_up DeployOvercloud:
END return value: 0

real     54m17.197s
user     0m3.421s
sys       0m0.670s
```

컨트롤러에서 오버클라우드 서비스 확인

- 모든 관리 서비스가 컨트롤러 노드에서 제대로 실행되는지 확인합니다.

```
[heat-admin@pod1-controller-2 ~]$ sudo pcs status
```

L3 에이전트 라우터 마무리

L3 에이전트가 제대로 호스팅되는지 확인하려면 라우터를 확인합니다. 이 검사를 수행할 때 overcloudrc 파일을 소싱해야 합니다.

- 라우터 이름 찾기:

<#root>

```
[stack@director~]$ source corerc
[stack@director ~]$ neutron router-list

+--------------------------------------+------+-----------------------------------------------------------

| id                                   | name | external_gateway_info

+--------------------------------------+------+-----------------------------------------------------------

| d814dc9d-2b2f-496f-8c25-24911e464d02 |
```

main

```
 | {"network_id": "18c4250c-e402-428c-87d6-a955157d50b5",            | False       | True |
```

이 예에서는 라우터의 이름이 main입니다.

- 실패한 노드 및 새 노드의 UUID를 찾기 위해 모든 L3 에이전트를 나열합니다.

```
[stack@director ~]$ neutron agent-list | grep "neutron-l3-agent"

| 70242f5c-43ab-4355-abd6-9277f92e4ce6 | L3 agent           | pod1-controller-0.localdomain  | nova
| 8d2ffbcb-b6ff-42cd-b5b8-da31d8da8a40 | L3 agent           | pod1-controller-2.localdomain  | nova
| a410a491-e271-4938-8a43-458084ffe15d | L3 agent           | pod1-controller-3.localdomain  | nova
| cb4bc1ad-ac50-42e9-ae69-8a256d375136 | L3 agent           | pod1-controller-1.localdomain  | nova
```

- 이 예에서는 pod1-controller-2.localdomain에 해당하는 L3 에이전트를 라우터에서 제거하고 pod1-controller-3.localdomain에 해당하는 에이전트를 라우터에 추가해야 합니다.

<#root>

```
[stack@director ~]$
```

neutron l3-agent-router-remove 8d2ffbcb-b6ff-42cd-b5b8-da31d8da8a40 main

```
Removed router main from L3 agent

[stack@director ~]$
```

```
neutron l3-agent-router-add a410a491-e271-4938-8a43-458084ffe15d main
```

```
Added router main to L3 agent
```

- 업데이트된 L3 상담원 목록 확인:

```
[stack@director ~]$ neutron l3-agent-list-hosting-router main

+--------------------------------------+--------------------------------------+----------------+-------+--
| id                                   | host                                 | admin_state_up | alive | h
+--------------------------------------+--------------------------------------+----------------+-------+--

| 70242f5c-43ab-4355-abd6-9277f92e4ce6 | pod1-controller-0.localdomain | True           | :-)   | standb
| a410a491-e271-4938-8a43-458084ffe15d | pod1-controller-3.localdomain | True           | :-)   | standb
| cb4bc1ad-ac50-42e9-ae69-8a256d375136 | pod1-controller-1.localdomain | True           | :-)   | activ
+--------------------------------------+--------------------------------------+----------------+-------+--
```

- 제거된 컨트롤러 노드에서 실행되는 모든 서비스를 나열하고 제거합니다.

```
[stack@director ~]$ neutron  agent-list | grep controller-2

| 877314c2-3c8d-4666-a6ec-69513e83042d | Metadata agent     | pod1-controller-2.localdomain  |
| 8d2ffbcb-b6ff-42cd-b5b8-da31d8da8a40 | L3 agent           | pod1-controller-2.localdomain  | nova
| 911c43a5-df3a-49ec-99ed-1d722821ec20 | DHCP agent         | pod1-controller-2.localdomain  | nova
| a58a3dd3-4cdc-48d4-ab34-612a6cd72768 | Open vSwitch agent | pod1-controller-2.localdomain  |

[stack@director ~]$ neutron agent-delete 877314c2-3c8d-4666-a6ec-69513e83042d
Deleted agent(s): 877314c2-3c8d-4666-a6ec-69513e83042d
[stack@director ~]$ neutron agent-delete 8d2ffbcb-b6ff-42cd-b5b8-da31d8da8a40
Deleted agent(s): 8d2ffbcb-b6ff-42cd-b5b8-da31d8da8a40
[stack@director ~]$ neutron agent-delete 911c43a5-df3a-49ec-99ed-1d722821ec20
Deleted agent(s): 911c43a5-df3a-49ec-99ed-1d722821ec20
[stack@director ~]$ neutron agent-delete a58a3dd3-4cdc-48d4-ab34-612a6cd72768
Deleted agent(s): a58a3dd3-4cdc-48d4-ab34-612a6cd72768

[stack@director ~]$ neutron agent-list | grep controller-2
[stack@director ~]$
```

컴퓨팅 서비스 마무리

- 제거된 노드에서 남아 있는 nova service-list 항목을 확인하고 삭제합니다.

```
[stack@director ~]$ nova service-list | grep controller-2

| 615 | nova-consoleauth | pod1-controller-2.localdomain  | internal          | enabled | down  | 2017-
| 618 | nova-scheduler   | pod1-controller-2.localdomain  | internal          | enabled | down  | 2017-
```

```
| 621 | nova-conductor   | pod1-controller-2.localdomain  | internal          | enabled | down  | 2017-

[stack@director ~]$ nova service-delete 615
[stack@director ~]$ nova service-delete 618
[stack@director ~]$ nova service-delete 621

stack@director ~]$ nova service-list | grep controller-2
```

- consoleauth 프로세스가 모든 컨트롤러에서 실행되는지 확인하거나 다음 명령을 사용하여 프로세스를 다시 시작합니다. pcs 리소스 다시 시작 openstack-nova-consoleauth:

```
[stack@director ~]$ nova service-list | grep consoleauth

| 601 | nova-consoleauth | pod1-controller-0.localdomain  | internal          | enabled | up    | 2017-
| 608 | nova-consoleauth | pod1-controller-1.localdomain  | internal          | enabled | up    | 2017-
| 622 | nova-consoleauth | pod1-controller-3.localdomain  | internal          | enabled | up    | 2017-
```

컨트롤러 노드에서 Fencing 다시 시작

- 모든 컨트롤러에서 언더클라우드 192.0.0.0/8에 대한 IP 경로를 확인합니다.

<#root>

```
[root@pod1-controller-3 ~]# ip route
default via 172.25.22.1 dev vlan101
11.117.0.0/24 dev vlan17  proto kernel  scope link  src 11.117.0.12
11.118.0.0/24 dev vlan18  proto kernel  scope link  src 11.118.0.12
11.119.0.0/24 dev vlan19  proto kernel  scope link  src 11.119.0.12
11.120.0.0/24 dev vlan20  proto kernel  scope link  src 11.120.0.12
169.254.169.254 via 192.200.0.1 dev eno1
172.25.22.0/24 dev vlan101  proto kernel  scope link  src 172.25.22.102
```

192.0.0.0/8 dev eno1 proto kernel scope link src 192.200.0.103

- 현재 stonith 컨피그레이션을 확인합니다. 이전 컨트롤러 노드에 대한 모든 참조를 제거합니다
  .

<#root>

```
[root@pod1-controller-3 ~]# sudo pcs stonith show --full
 Resource: my-ipmilan-for-controller-6 (class=stonith type=fence_ipmilan)
  Attributes: pcmk_host_list=pod1-controller-1 ipaddr=192.100.0.1 login=admin passwd=Csco@123Starent la
  Operations: monitor interval=60s (my-ipmilan-for-controller-6-monitor-interval-60s)
 Resource: my-ipmilan-for-controller-4 (class=stonith type=fence_ipmilan)
  Attributes: pcmk_host_list=pod1-controller-0 ipaddr=192.100.0.14 login=admin passwd=Csco@123Starent l
  Operations: monitor interval=60s (my-ipmilan-for-controller-4-monitor-interval-60s)
```

Resource: my-ipmilan-for-controller-7 (class=stonith type=fence_ipmilan)

```
  Attributes: pcmk_host_list=pod1-controller-2 ipaddr=192.100.0.15 login=admin passwd=Csco@123Starent l
  Operations: monitor interval=60s (my-ipmilan-for-controller-7-monitor-interval-60s)

[root@pod1-controller-3 ~]# pcs stonith delete
```

my-ipmilan-for-controller-7

```
Attempting to stop: my-ipmilan-for-controller-7...Stopped
```

- 새 컨트롤러에 대한 stonith 컨피그레이션 추가:

```
[root@pod1-controller-3 ~]sudo pcs stonith create my-ipmilan-for-controller-8 fence_ipmilan pcmk_host_l
```

- 컨트롤러에서 펜싱을 다시 시작하고 상태를 확인합니다.

```
[root@pod1-controller-1 ~]# sudo pcs property set stonith-enabled=true
[root@pod1-controller-3 ~]# pcs status

<snip>
my-ipmilan-for-controller-1 (stonith:fence_ipmilan): Started pod1-controller-3
my-ipmilan-for-controller-0 (stonith:fence_ipmilan): Started pod1-controller-3
my-ipmilan-for-controller-3 (stonith:fence_ipmilan): Started pod1-controller-3
```

사후 서버 교체 설정

이전 서버에 있던 설정을 적용하려면 아래 링크를 참조하십시오.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.