コントローラサーバUCS C240 M4のPCRF交換

内容

概要

このドキュメントでは、CPS仮想ネットワーク機能(VNF)をホストするUltra-Mセットアップで障害のあるコントローラサーバを交換するために必要な手順について説明します。

前提条件

バックアップ

リカバリの場合は、次の手順を使用してOSPDデータベース(DB)のバックアップを取ることを推奨します。

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

状態の予備確認

交換手順に進む前に、OpenStack環境とサービスの現在のステータスを確認し、正常であることを確認することが重要です。コントローラの交換プロセス時の複雑さを回避するのに役立ちます。

ステップ1: OpenStackのステータスとノードリストを確認します。

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

ステップ2：コントローラのPacemakerステータスを確認します。

アクティブなコントローラのいずれかにログインし、パケットメーカーのステータスを確認します。すべてのサービスが使用可能なコントローラで実行され、障害が発生したコントローラで停止している必要があります。

```
[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
```

この例では、Controller-2はオフラインです。したがって、交換されます。Controller-0とController-1は動作しており、クラスタサービスを実行しています。

ステップ3：アクティブなコントローラのMariaDBステータスを確認します。

```
[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=\"SHOW STATUS LIKE 'wsrep_local_state_comment'\" ; sudo mysql --
exec=\"SHOW STATUS LIKE 'wsrep_cluster_size'\""; done
*** 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
```

アクティブなコントローラごとに次の行があることを確認します。

wsrep_local_state_comment:同期済み

wsrep_cluster_size:0

ステップ4：アクティブなコントローラのRabbitmqステータスを確認します。障害が発生したコントローラは、実行するノードのリストに表示されません。

```
[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',[]}]}]
```

ステップ5：すべてのアンダークラウドサービスがOSP-Dノードからロード済み、アクティブ、および実行中のステータスであるかどうかを確認します。

```
[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 Utility
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
service
openstack-ceilometer-notification.service  loaded active running OpenStack ceilometer
notification agent
openstack-glance-api.service               loaded active running OpenStack Image Service (code-
named Glance) API server
openstack-glance-registry.service          loaded active running OpenStack Image Service (code-
named Glance) Registry server
openstack-heat-api-cfn.service             loaded active running Openstack Heat CFN-compatible
API Service
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 Inspector
openstack-ironic-inspector.service         loaded active running Hardware introspection service
for OpenStack Ironic
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) - Account Reaper
openstack-swift-account.service            loaded active running OpenStack Object Storage
(swift) - Account Server
openstack-swift-container-updater.service  loaded active running OpenStack Object Storage
(swift) - Container Updater
openstack-swift-container.service          loaded active running OpenStack Object Storage
(swift) - Container Server
openstack-swift-object-updater.service     loaded active running OpenStack Object Storage
(swift) - Object Updater
openstack-swift-object.service             loaded active running OpenStack Object Storage
(swift) - Object Server
openstack-swift-proxy.service              loaded active running OpenStack Object Storage
(swift) - Proxy Server
openstack-zaqar.service                    loaded active running OpenStack Message Queuing
Service (code-named Zaqar) Server
openstack-zaqar@1.service                  loaded active running OpenStack Message Queuing
Service (code-named Zaqar) Server Instance 1
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'.
```

コントローラクラスタでのフェンシングの無効化

```
[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
```

新しいコントローラノードのインストール

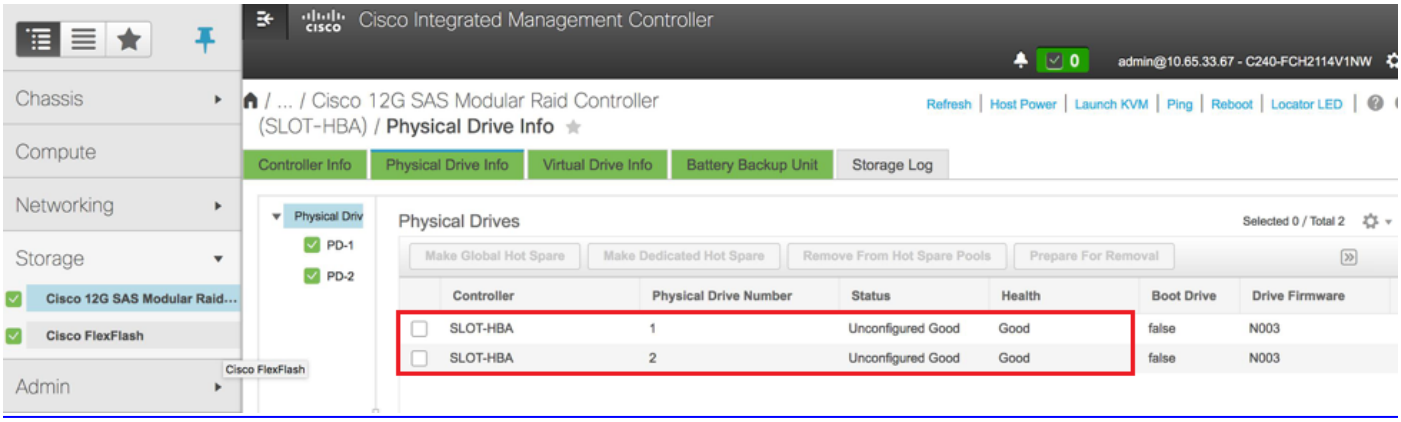
ステップ1：新しいUCS C240 M4サーバをインストールするための手順と初期セットアップ手順については、『Cisco UCS C240 M4サーバインストールおよびサービスガイド』を参照してください

ステップ2:CIMC IPを使用してサーバにログインします。

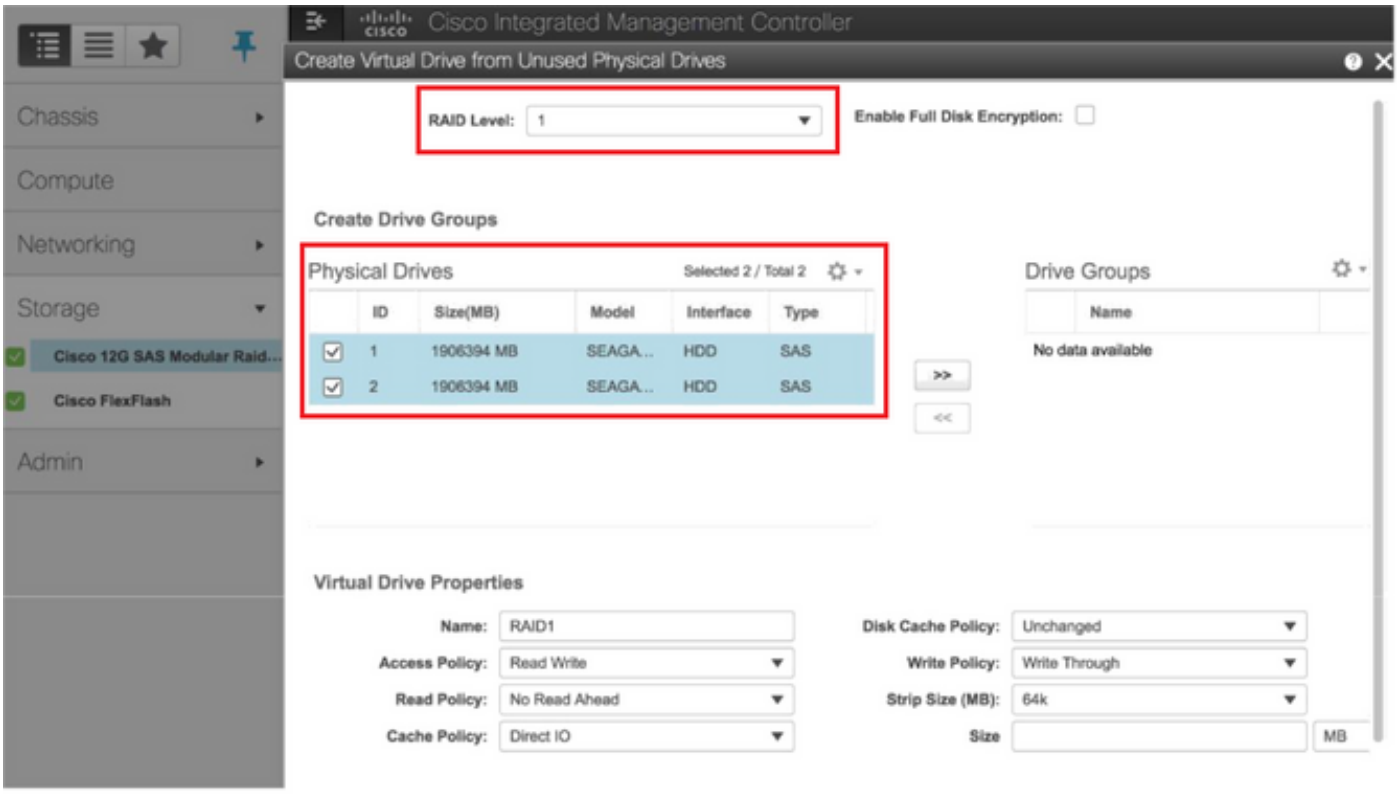
ステップ3:ファームウェアが以前に使用した推奨バージョンと異なる場合は、BIOSアップグレードを実行します。BIOSアップグレードの手順は次のとおりです。

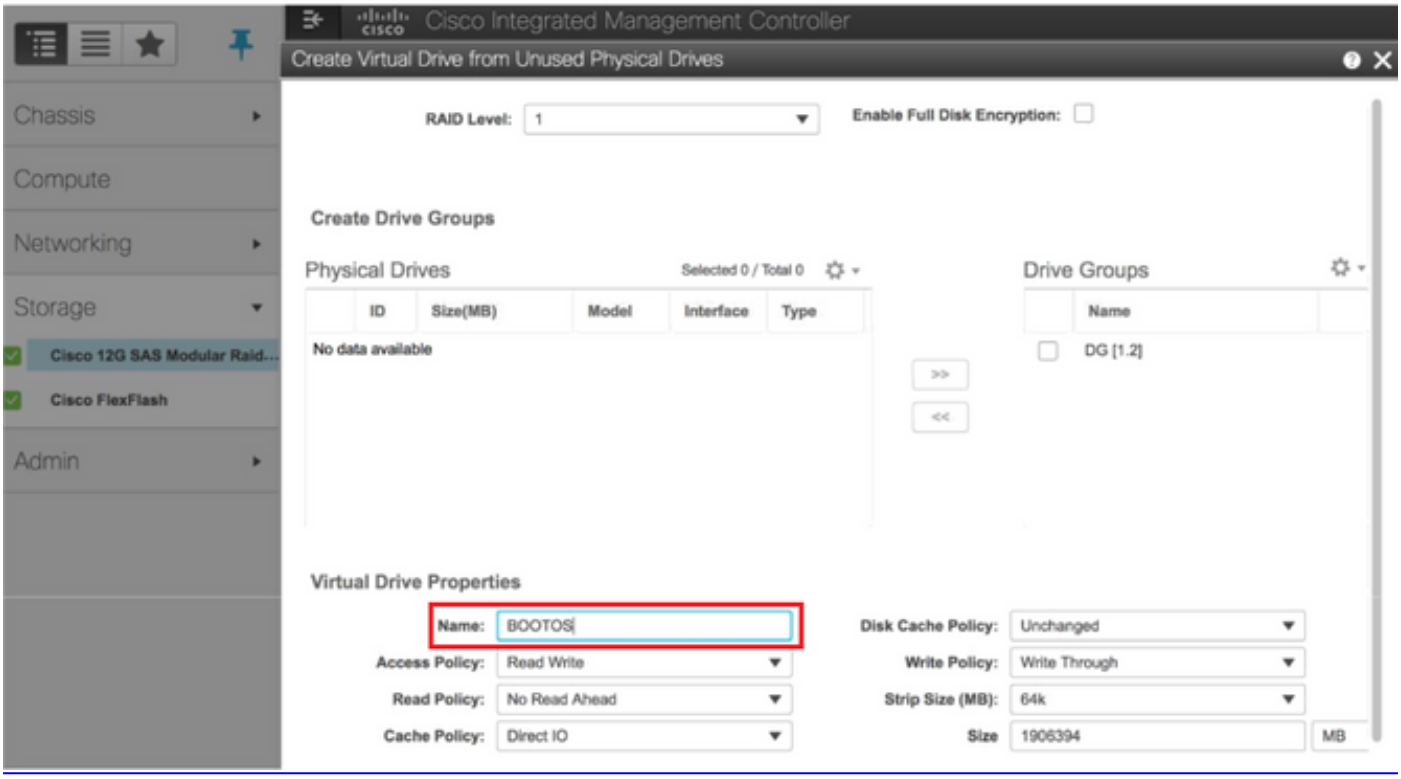
Cisco UCS Cシリーズラックマウントサーバ BIOSアップグレードガイド

ステップ4:物理ドライブのステータスを確認します。構成されていない正常な**状態である必要があります。[ストレージ] > [Cisco 12G SAS Modular Raid Controller (SLOT-HBA)] > [物理ドライブ情報]に移動します。**

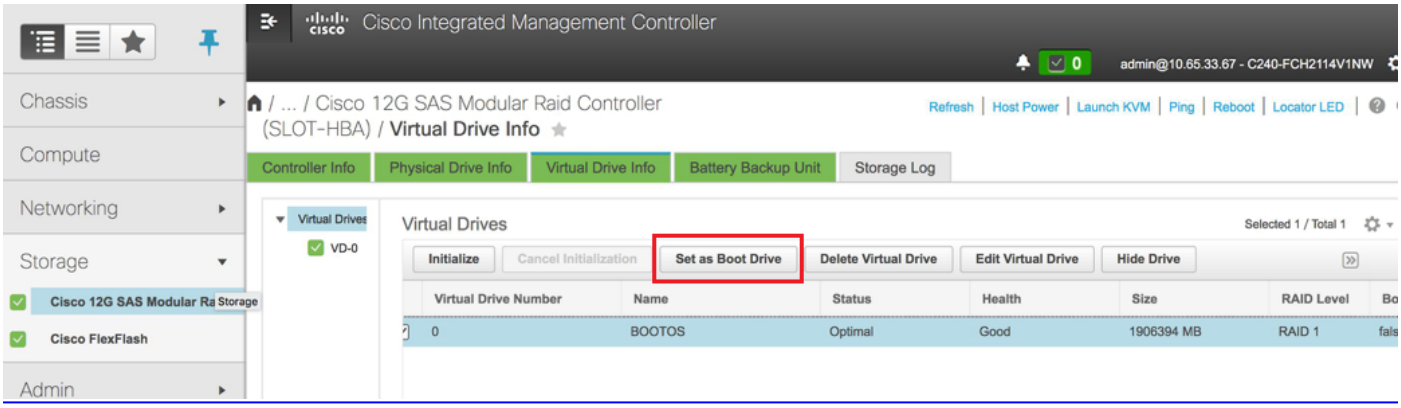


ステップ5:RAIDレベル1の物理ドライブから仮想ドライブを作成するには、次の手順を実行します。図に示すように、[Storage] > [Cisco 12G SAS Modular Raid Controller (SLOT-HBA)] > [Controller Info] > [Create Virtual Drive from Unused Physical Drives]に移動します。

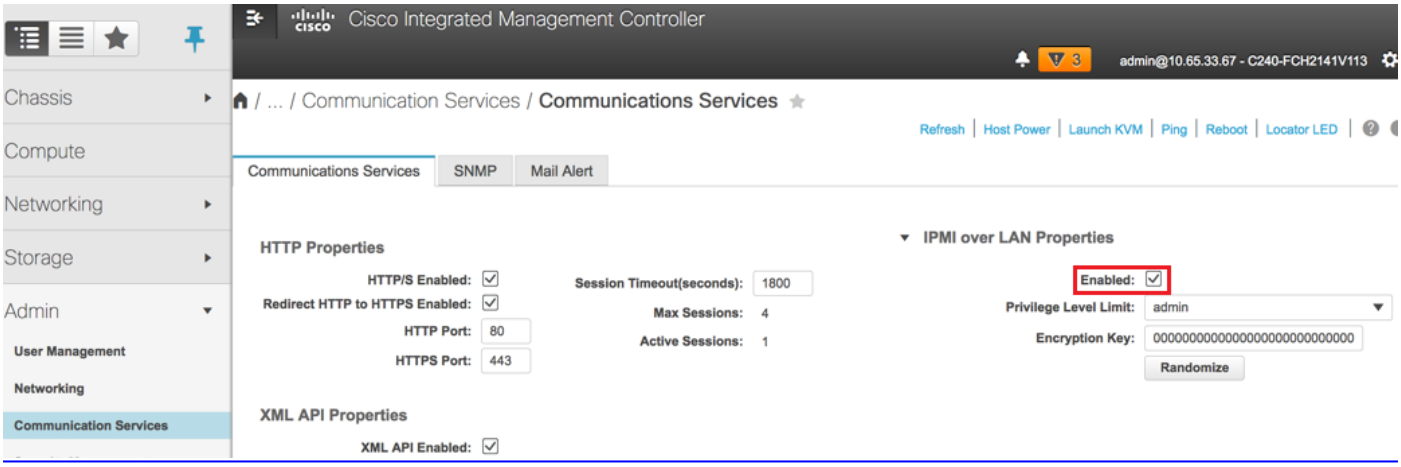




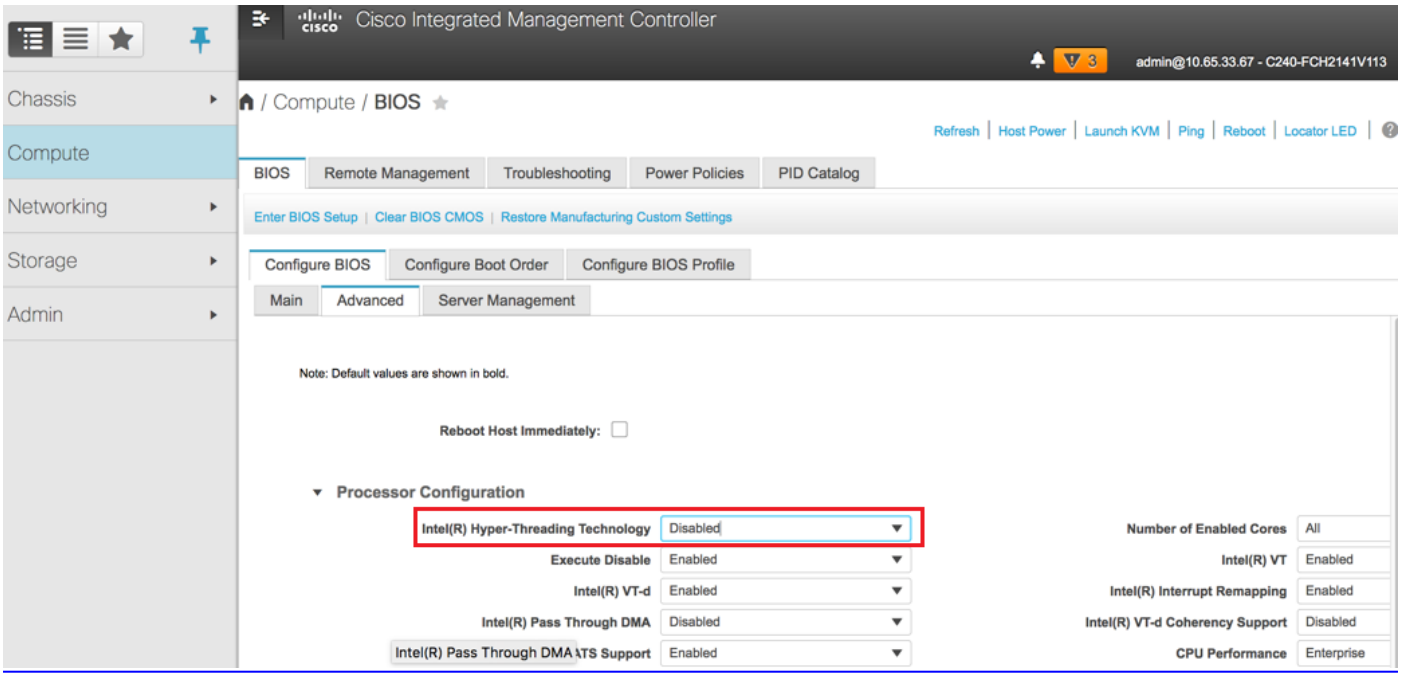
- VDを選択し、[Set as Boot Drive:



ステップ6:IPMI over LANを有効にするには、[Admin] > [Communication Services] > [Communication Services]に移動します。



ステップ7：ハイパースレッディングを無効にするには、図に示すように、[Compute] > [BIOS] > [Configure BIOS] > [Advanced] > [Processor Configuration]に移動します。



注：この図は次のとおりです。このセクションで説明する設定手順はファームウェアバージョン3.0(3e)を参照しており、他のバージョンで作業する場合は若干の違いがあります。

オーバークラウドでのコントローラノードの交換

このセクションでは、障害のあるコントローラをオーバークラウドの新しいコントローラに交換するために必要な手順について説明します。このため、スタックの起動に使用されたdeploy.shスクリプトが再利用されます。展開時に、ControllerNodesPostDeploymentフェーズで、Puppetモジュールの一部の制限により、更新が失敗します。配置スクリプトを再起動する前に、手動による介入が必要です。

障害が発生したコントローラノードの削除の準備

ステップ1：障害が発生したコントローラのインデックスを特定します。インデックスは、OpenStackサーバリスト出力のコントローラ名の数字のサフィックスです。この例では、インデックスは2です。

```
[stack@director ~]$ nova list | grep controller
| 5813a47e-af27-4fb9-8560-75decd3347b4 | pod1-controller-0  | ACTIVE | -          | Running
| ctlplane=192.200.0.152 |
| 457f023f-d077-45c9-bbea-dd32017d9708 | pod1-controller-1  | ACTIVE | -          | Running
| ctlplane=192.200.0.154 |
| d13bb207-473a-4e42-a1e7-05316935ed65 | pod1-controller-2  | ACTIVE | -          | Running
| ctlplane=192.200.0.151 |
```

ステップ2：削除するノードを定義するYamlファイル～templates/remove-controller.yamlを作成します。リソースリストのエントリには、前の手順で見つかったインデックスを使用します。

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

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

parameter_defaults:
 CorosyncSettleTries: 5
```

ステップ3：オーバークラウドをインストールするために使用する配置スクリプトのコピーを作成し、以前に作成したremove-controller.yamlファイルを含めるために行を挿入します。

```
[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
```

ステップ4:ここで説明するコマンドを使用して、交換するコントローラのIDを特定し、メンテナンスモードに移動します。

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

| 5813a47e-af27-4fb9-8560-75decd3347b4 | pod1-controller-0  | ACTIVE | -          | Running
| ctlplane=192.200.0.152 |

| 457f023f-d077-45c9-bbea-dd32017d9708 | pod1-controller-1  | ACTIVE | -          | Running
| ctlplane=192.200.0.154 |

| 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   | active             | True        |
```

ステップ5:交換手順の時点でDBが実行されるようにするには、Pacemakerの制御からGaleraを削除し、アクティブなコントローラのいずれかで次のコマンドを実行します。

```
[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 via crm_resource on pod1-controller-0
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>
```

新しいコントローラノードの追加の準備

ステップ1：新しいコントローラの詳細**のみを含む**controllerRMA.jsonファイルを作成します。新しいコントローラのインデックス番号が以前に使用されていないことを確認します。通常、次に大きいコントローラ番号に増分します。

例：最も前のバージョンはController-2だったので、Controller-3を作成します。

注：json形式に注意してください。

```
[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>"
       }
   ]
}
```

ステップ2：前のステップで作成したjsonファイルを使用して、新しいノードをインポートします。

```
[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
```

ステップ3：ノードを管理状態に設定します。

```
[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 |
```

ステップ4：イントロスペクションを実行します。

```
[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   | available          | False       |
```

ステップ5：使用可能なノードに新しいコントローラプロパティをマークします。controllerRMA.jsonファイルで使用されるように、新しいコントローラに指定されたコントローラIDを使用することを確認します。

```
[stack@director ~]$ openstack baremetal node set --property capabilities='node:controller-
3,profile:control,boot_option:local' 048ccb59-89df-4f40-82f5-3d90d37ac7dd
```

ステップ6：導入スクリプトには、layout.yamlというカスタムテンプレートがあり、これらのうちは、さまざまなインターフェイスのコントローラに割り当てられているIPアドレスを指定します。新しいスタックでは、Controller-0、Controller-1、およびController-2に3つのアドレスが定義されています。新しいコントローラを追加する場合は、各サブネットに次のIPアドレスを順に追加してください。

```
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
```

ステップ7：以前に作成したdeploy-removecontroller.shを実行し、古いノードを削除して新しいノードを追加します。

> **注：**この手順は、ControllerNodesDeployment_Step1で失敗することが予想されます。この時点で、手動による介入が必要です。

```
[stack@b10-ospd ~]$ ./deploy-addController.sh
START with options: [u'overcloud', u'deploy', u'--templates', u'-r', u'/home/stack/custom-
templates/custom-roles.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/puppet-pacemaker.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/network-isolation.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/storage-environment.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/neutron-sriov.yaml', u'-e', u'/home/stack/custom-templates/network.yaml',
u'-e', u'/home/stack/custom-templates/ceph.yaml', u'-e', u'/home/stack/custom-
templates/compute.yaml', u'-e', u'/home/stack/custom-templates/layout-removeController.yaml',
u'-e', u'/home/stack/custom-templates/rabbitmq.yaml', u'--stack', u'newtonoc', u'--debug', u'--
log-file', u'overcloudDeploy_11_15_17__07_46_35.log', u'--neutron-flat-networks',
u'phys_pcie1_0,phys_pcie1_1,phys_pcie4_0,phys_pcie4_1', u'--neutron-network-vlan-ranges',
u'datacentre:101:200', u'--neutron-disable-tunneling', u'--verbose', u'--timeout', u'180']
 :
DeploymentError: Heat Stack update failed
END return value: 1

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

導入の進行状況やステータスは、次のコマンドで監視できます。

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

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

| ID                                   | Stack
Name
                                                                  | Stack Status    | Creation
Time        | Updated Time         | Parent                                 |

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

| c1e338f2-877e-4817-93b4-9a3f0c0b3d37 | pod1-AllNodesDeploySteps-5psegydpwxij-
ComputeDeployment_Step1-swnuzjixac43
                                                                      | UPDATE_FAILED   |
2017-10-08T14:06:07Z | 2017-11-16T18:09:43Z | e90f00ef-2499-4ec3-90b4-d7def6e97c47 |

| 1db4fef4-45d3-4125-bd96-2cc3297a69ff | pod1-AllNodesDeploySteps-5psegydpwxij-
ControllerDeployment_Step1-
hmn3hpruubcn
   | UPDATE_FAILED   | 2017-10-08T14:03:05Z | 2017-11-16T18:12:12Z | e90f00ef-2499-4ec3-90b4-
d7def6e97c47 |

| e90f00ef-2499-4ec3-90b4-d7def6e97c47 | pod1-AllNodesDeploySteps-
5psegydpwxij
                                          | UPDATE_FAILED   | 2017-10-08T13:59:25Z | 2017-11-
16T18:09:25Z | 6c4b604a-55a4-4a19-9141-28c844816c0d |

| 6c4b604a-55a4-4a19-9141-28c844816c0d |
pod1
                                                                  | UPDATE_FAILED   | 2017-10-
08T12:37:11Z | 2017-11-16T17:35:35Z | None                                   |

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

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

手動介入

ステップ1:OSP-Dサーバ**で**OpenStack server listコマンドを実行して、使用可能なコントローラをリストします。新しく追加されたコントローラがリストに表示されます。

```
[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 |
overcloud-full |
| 5813a47e-af27-4fb9-8560-75decd3347b4 | pod1-controller-0  | ACTIVE | ctlplane=192.200.0.152 |
overcloud-full |
```

ステップ2：新しく追加されたコントローラではなく、アクティブなコントローラのいずれかに接続し、/etc/corosync/corosycn.confファイルを参照**します。**各コントロ**ーラに**nodeidを割り当てる**nodelist**を探します。失敗したノードのエントリを検索し、そのノードIDをメモ**します。**

```
[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
   }
}
```

ステップ3：アクティブコントローラのそれぞれにログインします。障害が発生したノードを削除し、サービスを再起動します。この場合は、**pod1-controller-2**を削除し**ます。新しく追加されたコントロー**ラに対してこの操作を実行しないでください。

```
[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
```

ステップ4:クラスタから障害が発生したノードを削除するには、アクティブなコントローラのいずれかからこのコマンドを実行します。

```
[root@pod1-controller-0 ~]# sudo crm_node -R pod1-controller-2 --force
```

ステップ5：障害が発生したノードをrabbitmqクラスタから削除するには、アクティブなコントローラの1つからこのコマンドを実行します。

```
[root@pod1-controller-0 ~]# sudo rabbitmqctl forget_cluster_node rabbit@pod1-controller-2
Removing node 'rabbit@newtonoc-controller-2' from cluster ...
```

ステップ6:MongoDBから失敗したノードを削除します。そのためには、アクティブなMongoノードを見つける必要があります。netstatを使用して、ホストのIPアドレスを検索します。

```
[root@pod1-controller-0 ~]# sudo netstat -tulnp | grep 27017
tcp        0      0 11.120.0.10:27017       0.0.0.0:*               LISTEN
219577/mongod
```

ステップ7：ノードにログインし、前のコマンドのIPアドレスとポート番号を使用してマスターであるかどうかを確認します。

```
[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
}
```

ノードがマスターでない場合は、他のアクティブコントローラにログインし、同じ手順を実行します。

ステップ8：マスターから、rs.status()コマンドを使用して使用可能なノードをリストします。古い/応答しないノードを見つけ、mongoノード名を特定します。

```
[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
}
```

ステップ9:マスターから、**rs**.removeコマンドを使用して、障害が発生したノードを**削除します**。このコマンドを実行すると、いくつかのエラーが表示されますが、ノードが削除されたことを確認するには、もう一度ステータスを確認します。

```
[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
```

ステップ10:このコマンドを実行して、アクティブなコントローラノードのリストを更新します。新しいコントローラノードをこのリストに含めます。

```
[root@pod1-controller-0 ~]# sudo pcs resource update galera wsrep_cluster_address=gcomm://pod1-controller-0,pod1-controller-1,pod1-controller-2
```

ステップ11：既に存在するコントローラから新しいコントローラに次のファイルをコピーします。

/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
```

ステップ12：既に存在するコントローラの1つからcluster node addコマンドを実行します。

```
[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
```

ステップ13：各コントローラにログインし、ファイル/etc/corosync/corosync.confを表示します。新しいコントローラがリストされていることを確認し、そのコントローラに割り当てられたノードが、以前に使用されていないシーケンスの次の番号であることを確認します。この変更が3つすべてのコントローラで行われていることを確認します。

```
[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
}
```

たとえば、修正後の/etc/corosync/corosync.confは次のようになります。

```
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
}
```

ステップ14：アクティブコントローラでcorosyncを再起動します。新しいコントローラでcorosyncを開始しない。

```
[root@pod1-controller-0 ~]# sudo pcs cluster reload corosync
[root@pod1-controller-1 ~]# sudo pcs cluster reload corosync
```

ステップ15:動作中のいずれかのコントローラから新しいコントローラノードを起動します。

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

ステップ16：アクティングコントローラの1つからGaleraを再起動します。

```
[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
```

ステップ17：クラスタはメンテナンスモードです。サービスを開始するには、メンテナンスモードをディセーブルにします。

```
[root@pod1-controller-2 ~]# sudo pcs property set maintenance-mode=false --wait
```

ステップ18:Galeraで3つのコントローラがすべてマスターとしてリストされるまで、GaleraのPCのステータスを確認します。

注：大規模なセットアップでは、DBの同期に時間がかかる場合があります。

```
[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 ]
```

ステップ19：クラスタをメンテナンスモードに切り替えます。

```
[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 via cibadmin on pod1-controller-1
              *** 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
```

ステップ20:以前に実行した配置スクリプトを再実行します。今回は成功するはずです。

```
[stack@director ~]$ ./deploy-addController.sh
START with options: [u'overcloud', u'deploy', u'--templates', u'-r', u'/home/stack/custom-
templates/custom-roles.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/puppet-pacemaker.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/network-isolation.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/storage-environment.yaml', u'-e', u'/usr/share/openstack-tripleo-heat-
templates/environments/neutron-sriov.yaml', u'-e', u'/home/stack/custom-templates/network.yaml',
u'-e', u'/home/stack/custom-templates/ceph.yaml', u'-e', u'/home/stack/custom-
templates/compute.yaml', u'-e', u'/home/stack/custom-templates/layout-removeController.yaml',
u'--stack', u'newtonoc', u'--debug', u'--log-file', u'overcloudDeploy_11_14_17__13_53_12.log',
u'--neutron-flat-networks', u'phys_pcie1_0,phys_pcie1_1,phys_pcie4_0,phys_pcie4_1', u'--neutron-
network-vlan-ranges', u'datacentre:101:200', u'--neutron-disable-tunneling', u'--verbose', u'--
timeout', u'180']
options: Namespace(access_key='', access_secret='***', access_token='***',
access_token_endpoint='', access_token_type='', aodh_endpoint='', auth_type='',
auth_url='https://192.200.0.2:13000/v2.0', authorization_code='', cacert=None, cert='',
client_id='', client_secret='***', cloud='', consumer_key='', consumer_secret='***', debug=True,
default_domain='default', default_domain_id='', default_domain_name='', deferred_help=False,
discovery_endpoint='', domain_id='', domain_name='', endpoint='', identity_provider='',
identity_provider_url='', insecure=None, inspector_api_version='1', inspector_url=None,
interface='', key='', log_file=u'overcloudDeploy_11_14_17__13_53_12.log', murano_url='',
old_profile=None, openid_scope='', os_alarming_api_version='2',
os_application_catalog_api_version='1', os_baremetal_api_version='1.15', os_beta_command=False,
os_compute_api_version='', os_container_infra_api_version='1',
os_data_processing_api_version='1.1', os_data_processing_url='', os_dns_api_version='2',
os_identity_api_version='', os_image_api_version='1', os_key_manager_api_version='1',
os_metrics_api_version='1', os_network_api_version='', os_object_api_version='',
```

```
os_orchestration_api_version='1', os_project_id=None, os_project_name=None,
os_queues_api_version='2', os_tripleoclient_api_version='1', os_volume_api_version='',
os_workflow_api_version='2', passcode='', password='***', profile=None, project_domain_id='',
project_domain_name='', project_id='', project_name='admin', protocol='', redirect_uri='',
region_name='', roles='', timing=False, token='***', trust_id='', url='', user='',
user_domain_id='', user_domain_name='', user_id='', username='admin', verbose_level=3,
verify=None)
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
```

コントローラでのオーバークラウドサービスの確認

すべてのマネージドサービスがコントローラノードで正しく実行されていることを確認します。

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

L3エージェントルータの確定

ルータをチェックして、L3エージェントが正しくホストされていることを確認します。このチェックを実行する際は、必ずovercloudrcファイルをソースにしてください。

ステップ1：ルータ名を検索します。

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

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

| id                                   | name |
external_gateway_info                                        | distributed | ha   |

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

| d814dc9d-2b2f-496f-8c25-24911e464d02 | main | {"network_id": "18c4250c-e402-428c-87d6-
a955157d50b5",            | False       | True |
```

この例では、ルータの名前はmainです。

ステップ2：障害が発生したノードと新しいノードのUUIDを見つけるために、すべてのL3エージェントをリストします。

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

| 70242f5c-43ab-4355-abd6-9277f92e4ce6 | L3 agent           | pod1-controller-0.localdomain  |
```

```
nova              | :-)   | True          | neutron-l3-agent          |
| 8d2ffbcb-b6ff-42cd-b5b8-da31d8da8a40 | L3 agent           | pod1-controller-2.localdomain  |
nova              | xxx   | True          | neutron-l3-agent          |
| a410a491-e271-4938-8a43-458084ffe15d | L3 agent           | pod1-controller-3.localdomain  |
nova              | :-)   | True          | neutron-l3-agent          |
| cb4bc1ad-ac50-42e9-ae69-8a256d375136 | L3 agent           | pod1-controller-1.localdomain  |
nova              | :-)   | True          | neutron-l3-agent          |
```

ステップ3：この例では、**pod1-controller-2.localdomainに対応するL3エージェントをルータから削除する必要があります。pod1-controller-3.localdomainに対応するエージェントをルータに追加する必要があります。**

```
[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
```

ステップ4：更新されたL3エージェントのリストを確認します。

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

+--------------------------------------+--------------------------------+----------------+---
----+----------+
| id                                   | host                           | admin_state_up |
alive | ha_state |
+--------------------------------------+--------------------------------+----------------+---
----+----------+

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

ステップ5：削除されたコントローラーノードから実行されているサービスをすべてリストし、削除します。

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

| 877314c2-3c8d-4666-a6ec-69513e83042d | Metadata agent     | pod1-controller-2.localdomain
|                   | xxx   | True          | neutron-metadata-agent    |
| 8d2ffbcb-b6ff-42cd-b5b8-da31d8da8a40 | L3 agent           | pod1-controller-2.localdomain  |
nova              | xxx   | True          | neutron-l3-agent          |
| 911c43a5-df3a-49ec-99ed-1d722821ec20 | DHCP agent         | pod1-controller-2.localdomain  |
nova              | xxx   | True          | neutron-dhcp-agent        |
| a58a3dd3-4cdc-48d4-ab34-612a6cd72768 | Open vSwitch agent | pod1-controller-2.localdomain
|                   | xxx   | True          | neutron-openvswitch-agent |

[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 ~]$
```

コンピューティングサービスの最終決定

ステップ1：削除されたノードから残っているnovaサービスリスト項目を確認し、削除します。

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

| 615 | nova-consoleauth | pod1-controller-2.localdomain  | internal          | enabled | down
| 2017-11-16T16:08:14.000000 | -               |
| 618 | nova-scheduler   | pod1-controller-2.localdomain  | internal          | enabled | down
| 2017-11-16T16:08:13.000000 | -               |
| 621 | nova-conductor   | pod1-controller-2.localdomain  | internal          | enabled | down
| 2017-11-16T16:08:14.000000 | -

[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
```

ステップ2：コンソネータプロセスがすべてのコントローラで実行されていることを確認するか、次のコマンドを使用して再起動します。pcs resource restart openstack-nova-console-auth:

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

| 601 | nova-consoleauth | pod1-controller-0.localdomain  | internal          | enabled | up
| 2017-11-16T21:00:10.000000 | -               |
| 608 | nova-consoleauth | pod1-controller-1.localdomain  | internal          | enabled | up
| 2017-11-16T21:00:13.000000 | -               |
| 622 | nova-consoleauth | pod1-controller-3.localdomain  | internal          | enabled | up
| 2017-11-16T21:00:13.000000 | -
```

コントローラノードでのフェンシングの再起動

ステップ1:すべてのコントローラでアンダークラウド192.0.0.0/8へのIPルートを確認します

```
[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
```

ステップ2：現在のストニス構成を確認してください。古いコントローラノードへの参照を削除します。

```
[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 lanplus=1
```

```
 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 lanplus=1
 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 lanplus=1
 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
```

ステップ3：新しいコントローラのストニット構成を追加します。

```
[root@pod1-controller-3 ~]sudo pcs stonith create my-ipmilan-for-controller-8 fence_ipmilan
pcmk_host_list=pod1-controller-3 ipaddr=<CIMC_IP> login=admin passwd=<PASSWORD> lanplus=1 op
monitor interval=60s
```

ステップ4：任意のコントローラからフェンシングを再起動し、ステータスを確認します。

```
[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
```