インストール、アップグレード、リブートのトラブルシューティング

この章は、次の項で構成されています。

Orchestrator VM の CPU サイクル予約の増加

Cisco ACI Multi-Site Orchestrator VM には、一定量の専用 CPU サイクルが必要です。新しい展開では CPU サイクルの予約が自動的に適用されますが、リリース 2.1(1) より前のリリースから Orchestrator をアップグレードする場合は、各 Orchestrator VM の設定を手動で更新する必要があります。

適切な CPU サイクル予約を構成すると、次のようなランダムに見える多くの問題を解決または防止できます。

- ロードに 1 回以上の再試行が必要な Orchestrator GUI アイテム。
- 1 つまたは複数のノードが [不明（Unknown）] ステータスに変化し、後でそれ自体が [準備完了（Ready）] に解決されます。

```
# docker node ls
ID                            HOSTNAME    STATUS      AVAILABILITY    MANAGER STATUS
  ENGINE VERSION
```

```
t8wl1zoke0vpxdl9fysqu9otb     node1       Ready       Active          Reachable
   18.03.0-ce
kyriihdfhy1k1tlggan6e1ahs *   node2       Unknown     Active          Reachable
   18.03.0-ce
yburwactxd86dorindmx8b4y1     node3       Ready       Active          Leader
   18.03.0-ce
```

- 次のログ エントリの例では、Orchestrator ログ（`/var/log/messages` にあります）で一時的なハートビートの欠落が発生します。

```
node2 dockerd: [...] level=error msg="agent: session failed" backoff=100ms
    error="rpc error: code = Canceled desc = context canceled" module=node/agent
[...]
node2 dockerd: [...] level=error msg="heartbeat to manager [...] failed"
    error="rpc error: code = Canceled desc = context canceled" [...]
```

CPU サイクルの予約設定を更新するには、Orchestrator VM ごとに次の手順を繰り返します。

ステップ 1 vSphere クライアントにログインします。

ステップ 2 Orchestrator VM が配置されている ESX ホストに移動します。

ステップ 3 VM をシャットダウンします。

ステップ 4 VM を右クリックし、**[設定の編集（Edit Settings）]** を選択します。

ステップ 5 **[仮想ハードウェア（Virtual Hardware）]** タブで、**CPU** カテゴリを展開します。

ステップ 6 **[予約（Reservation）]** フィールドに、`10 GHz` と入力します。

ステップ 7 [OK] をクリックして変更を保存します。

ステップ 8 VM の電源を入れ、すべてのノードが正常な状態で Orchestrator クラスタが安定するのを待ちます。

Orchestrator ノードの NTP の有効化

Orchestrator ノードにクロック同期が構成されていないと、認証トークンの有効期限切れによるランダムな GUI セッション ログオフなどの問題が発生する可能性があります。

通常、Multi-Site Orchestrator のインストール中に、Orchestrator ノードの Network Time Protocol（NTP）サーバの詳細を指定します。ただし、何らかの理由で NTP 設定を指定していない場合は、次の手順を使用して設定できます。

ステップ 1 Orchestrator VM に直接ログインします。

ステップ 2 スクリプト ディレクトリに変更します。

```
# cd /opt/cisco/msc/scripts
```

ステップ 3 ノードに NTP 設定を構成します。

次のコマンド：

- '**-tz** *<time-zone>*' では、今いるタイムゾーンを指定します

- '**-ne** ' は NTP を有効にします
- '**-ns** *<ntp-server>*' は NTP サーバを指定します

```
# ./svm-msc-tz-ntp  -tz <time-zone> -ne -ns <ntp-server>
```

次に例を示します。

```
# ./svm-msc-tz-ntp -tz US/Pacific -ne -ns ntp.esl.cisco.com
svm-msc-tz-ntp: Start
svm-msc-tz-ntp: Executing timedatectl set-timezone US/Pacific
svm-msc-tz-ntp: Executing sed -i 's|^server|\# server|' /etc/ntp.conf
svm-msc-tz-ntp: Executing timedatectl set-ntp true
svm-msc-tz-ntp: Sleeping 10 seconds
svm-msc-tz-ntp: Checking NTP status
svm-msc-tz-ntp: Executing ntpstat;ntpq -p
unsynchronised
   polling server every 64 s
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
mtv5-ai27-dcm10 .GNSS.           1 u    -   64    1    1.581   -0.002   0.030
```

ステップ 4 NTP 構成を確認します。

次のコマンドを使用して、NTP が有効になっていることを確認できます。

```
# ntpstat;ntpq -p
unsynchronised
   polling server every 64 s
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*mtv5-ai27-dcm10 .GNSS.           1 u   14   64    1    3.522   -0.140   0.128
```

正しい日付と時刻が設定されていることも確認できます。

```
# date
Mon Jul  8 14:19:26 PDT 2019
```

ステップ 5 各ノードでこの手順を繰り返します。

DNS の更新

このセクションでは、Multi-Site Orchestrator クラスタの DNS サーバ アドレスを更新する方法について説明します。この手順は、VMware ESX の MSO OVA 展開にのみ適用され、アプリケーション サービス エンジンまたは Nexus ダッシュボードの展開には適用されないことに注意してください。

ステップ 1 `root` ユーザーとしてクラスタ ノードの 1 つに SSH で接続します。

ステップ 2 DNS 構成を更新します。

`nmcli` コマンドを使用して、DNS サーバの IP アドレスを更新します。

```
# nmcli connection modify eth0 ipv4.dns "<dns-server-ip>"
```

複数の DNS サーバの IP を指定する場合は、スペースで区切ったリストを使用します。

```
# nmcli connection modify eth0 ipv4.dns "<dns-server-ip-1> <dns-server-ip-2>"
```

ステップ 3 更新したネットワーク インターフェイスを再起動します。

変更を適用するには、eth0 インターフェイスを再起動する必要があります。

```
# nmcli connection down eth0 && nmcli connection up eth0
```

ステップ 4 ノードをリブートします。

ステップ 5 他の 2 つのノードについても前の手順を繰り返します。

一時的にダウンした場合クラスタの単一ノードを再起動する

このセクションでは、クラスタの1つのノードが一時的にダウンした場合に再起動する方法について説明します。

ダウンしたノードを再起動します。追加の手順は必要なく、クラスタは自動的に回復します。

一時的にダウンしているクラスタの 2 つのノードを再起動する

このセクションでは、一時的にダウンしたクラスタの2つのノードを再起動する方法について説明します。

ステップ 1 現時点では、Docker スウォームに 3 つのマネージャー ノードのクォーラムがないため、マルチサイト は利用できません。リカバリを試みる前に、MongoDB をバックアップすることをお勧めします。

詳細については、MongoDB のバックアップ Cisco ACI マルチサイト （4 ページ）を参照してください。

ステップ 2 ダウンしていた 2 つのノードを再起動します。追加で必要な手順はありません。クラスタが自己回復します。

MongoDB のバックアップ Cisco ACI マルチサイト

このセクションで説明するように、CiscoはCisco ACIマルチサイトOrchestratorのアップグレードまたはダウングレードの前にMongoDBをバックアップすることを推奨します。

（注） データベースをバックアップできるのは、Orchestrator クラスタが稼働している場合のみです。これには、少なくとも 2 つのノードが稼働している必要があります。

ステップ 1 Cisco ACI マルチサイト Orchestrator 仮想マシン（VM）にログインします。

ステップ 2 Cisco ACI マルチサイト Orchestrator バックアップ スクリプトを実行します。

```
# ~/msc_scripts/msc_db_backup.sh
```

msc_backup_<*date+%Y%m%d%H%M*>.archive ファイルが作成されます。

ステップ 3 msc_backup_<*date+%Y%m%d%H%M*>.archive ファイルを安全な場所にコピーします。

Cisco ACI マルチサイト 向け MongoDB の復元

このセクションでは、Cisco ACI マルチサイト 向け MongoDB を復元する方法について説明します。

ステップ 1 マルチサイト 仮想マシン（VM）にログインします。

ステップ 2 msc_backup_<*date+%Y%m%d%H%M*>.archive ファイルを VM にコピーします。

ステップ 3 マルチサイト DB 復元スクリプトを実行します。

```
# ~/msc_scripts/msc_db_restore.sh
```

ステップ 4 Python スクリプトを実行して、スキーマを再度プッシュします。

```
# msc_push_schemas.py
```

カスタム証明書のトラブルシューティング

ここでは、マルチサイト Orchestrator でカスタム SSL 証明書を使用する場合の一般的な問題を解決する方法について説明します。

Orchestrator GUI をロードできません

カスタム証明書をインストールしてアクティブ化した後に Orchestrator GUI ページをロードできない場合は、各 Orchestrator ノードに証明書が正しくコピーされていない可能性があります。この問題を解決するには、デフォルトの証明書を回復してから、新しい証明書のインストール手順を再度繰り返します。

デフォルトの Orchestrator 証明書を回復するには、次のようにします。

ステップ 1 各 Orchestrator ノードに直接ログインします。

ステップ 2 証明書ディレクトリに移動します。

```
# cd /data/msc/secrets
```

ステップ 3 msc.key および msc.crt ファイルを、msc.key_backup および msc.crt_backup ファイルに個別に置き換えます。

msc.key および msc.crt ファイルを、msc.key_backup および msc.crt_backup ファイルに個別に置き換えます。

ステップ 4 Orchestrator GUI サービスを再起動します。

```
# docker service update msc_ui --force
```

ステップ 5 前のセクションで説明したように、新しい証明書を再インストールしてアクティブにします。

クラスタへの新しい Orchestrator ノードの追加

マルチサイト Orchestrator クラスタに新しいノードを追加する場合は、次のようにします。

ステップ 1 Orchestrator GUI にログインします。

ステップ 2 前のセクションで説明したように、使用しているキーを再度アクティブにします。

デフォルトのキーリングの有効期限が切れた後に新しいキーリングをインストールできない

デフォルトのキーリングの有効期限が切れた後に新しいキーリングをインストールできない場合は、カスタムキーリングがクラスタノードにインストールされていない可能性があります。

この問題を解決するには、以下の手順を使用して、古いデフォルトのキーリングを削除し、新しいキーリングを作成します。

ステップ 1 クラスタのすべてのノードで次のコマンドを実行します。

```
cd /data/msc/secrets
rm -rf /data/msc/secrets/msc.key
rm -rf /data/msc/secrets/msc.crt
rm -rf /data/msc/secrets/msc.key_backup
rm -rf /data/msc/secrets/msc.crt_backup
!
!
openssl req -newkey rsa:2048 -nodes -keyout /data/msc/secrets/msc.key -x509 -days 365 -out
```

```
/data/msc/secrets/msc.crt -subj '/CN=MSC'
cp /data/msc/secrets/msc.key /data/msc/secrets/msc.key_backup
cp /data/msc/secrets/msc.crt /data/msc/secrets/msc.crt_backup
cd /data/msc/secrets
chmod 777 msc.key
chmod 777 msc.key_backup
chmod 777 msc.crt
chmod 777 msc.crt_backup
```

ステップ 2 次のコマンドを実行して、msc_ui サービスの更新を強制します。

```
# docker service update msc_ui --force
```

ステップ 3 更新が完了したら、msc_ui のすべての複製が正常かどうかを確認します。

```
[root@node1 ~]# docker service ls
...
rqs0607lgixg     msc_ui     global     3/3    msc-ui:3.1.1i
*:443->443/tcp
```

ステップ 4 任意のブラウザを使用して、任意の MSO ノードにログインします。証明書の詳細を受け入れるときにブラウザがループでスタックするか、白い画面が表示される場合は、GUI が再び正常に表示されるまでページを 1 ～ 2 回更新します。

ステップ 5 ユーザー名とパスワードを使用してログインし、「カスタム キーリングのアクティブ化」セクションで説明されている手順に従ってキーリングをアクティブ化します。

クラスタの単一ノードを新しいノードに置き換える

このセクションでは、クラスタの単一ノードを新しいノードに置き換える方法について説明します。

このシナリオでは、ノード 1 がダウンし、ノード 1 を新しいノードに置き換える必要があります。

ステップ 1 既存のノードで、ダウンしているノード (node1) の ID を取得します。次のコマンドを実行します。

```
root@node2 ~]# docker node ls
ID                           HOSTNAME  STATUS  AVAILABILITY   MANAGER STATUS
11624powztg5tl9nlfoubydtp *  node2     Ready   Active         Leader
fsrca74nl7byt5jcv93ndebco    node3     Ready   Active         Reachable
wnfs9oc687vuusbzd3o7idllw    node1     Down    Active         Unreachable
```

ステップ 2 node1 を降格し、次のコマンドを実行する必要があります。

```
[root@node2 ~]# docker node demote <node ID>
Manager <node ID> demoted in the swarm.
```

<*node ID*> は、手順 1 でノード ID を受け取った場所です。

ステップ 3 新しいノードを追加する前にダウンしている node1 を削除し、次のコマンドを実行します。

```
[root@node2 ~]# docker node rm <node ID>
```

ステップ 4 既存のノードで、/opt/cisco/msc/builds/<*build_number*>/prodhaディレクトリに変更します。

例：

```
# cd /opt/cisco/msc/builds/<build_number>/prodha
```

ステップ 5 トークンをメモします。既存のノードで、次のコマンドを実行します。

```
[root@node1 prodha]# docker swarm join-token manager
docker swarm join --token
SWMTKN-1-4yaodn4nj8nek0qghh4dfzn6zm9o9p29rjisdikhjpvwu8bgmw-0ig2g62e0fe62cq2hbexk6xgv \
1.1.1.1:2376
```

ステップ 6 新しいリーダーの IP アドレスを書き留めます。既存のノードで、次のコマンドを入力します。

例：

```
[root@node1 prodha]# docker node ls
ID                           HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS
pjicie1wlcgkoef1x9s0td7ac    node1      Down     Active         Reachable
qy6peh6wtsbsaf9cpyh2wr5f6    node2      Ready    Active         Leader
tfhhvzt7qx9lxkqalbxfwknsq    node3      Ready    Active         Reachable
```

ステップ 7 リーダー ノード（node2）で、IP アドレスをメモします。

```
# ifconfig
inet 10.23.230.152 netmask 255.255.255.0 broadcast 192.168.99.255
```

ステップ 8 新しい 3 番目のノードを準備します。新しいノードの正しいホスト名を設定します。

例：

```
# hostnamectl set-hostname <node name>
```

ステップ 9 /opt/cisco/msc/builds/<ビルド番号>/prodha ディレクトリに変更します。

例：

```
# cd /opt/cisco/msc/builds/<build_number>/prodha
```

ステップ 10 新しいノードをスウォームに参加させます:

例：

```
[root@node1 prodha]# ./msc_cfg_join.py <token> <address of leader>
```

<*token*> は、手順 5 でトークン情報を受け取った場所です。

<*address of leader*> は、手順 7 でリーダーの IP アドレスを受け取った場所です。

ステップ 11 いずれかのノードで、/opt/cisco/msc/builds/<*build_number*>/prodha ディレクトリに変更します。

例：

```
# cd /opt/cisco/msc/builds/<build_number>/prodha
```

ステップ 12 いずれかのノードで、次のコマンドを実行します。

```
[root@node1 prodha]# ./msc_deploy.py
```

この時点で、すべてのサービスが稼働し、データベースが複製されているはずです。

クラスタの 2 つの既存のノードを新しいノードに置き換える

このセクションでは、クラスタの2つの既存のノードを新しいノードに置き換える方法について説明します。ここでの手順は、クラスタに Docker スウォームを使用する ESX VMware VM での Orchestrator の展開に関するものです。

この時点では、Docker スウォームに 3 つのマネージャ ノードのクォーラムがないため、マルチサイト Orchestrator は使用できません。リカバリを試みる前に、DB をバックアップすることをお勧めします。詳細については、MongoDB のバックアップ Cisco ACI マルチサイト （4 ページ）を参照してください。

ステップ 1 2 つの新しいノードを起動し、新しいノードごとに適切なノード名を設定します。

新しいノードを起動したら、次のコマンドを使用してそのノード名を割り当てることができます。

```
# hostnamectl set-hostname <node-name>
```

3 つのノード名はすべてクラスタ内で一意である必要があることに注意してください。

ステップ 2 以前はスウォームの一部であった唯一のライブ ノードで、ダウンしている他のノードを削除します。

a) スウォームの一部である唯一のライブ ノードに SSH で接続します。

b) すべてのノードのステータスを表示します。

```
# docker node ls
ID                           HOSTNAME   STATUS   AVAILABILITY  MANAGER     STATUS
g3mebdulaed2n0cyywjrtum31     node2      Down     Active        Reachable
ucgd7mm2e2divnw9kvm4in7r7     node1      Ready    Active        Leader
zjt4dsodu3bff3ipn0dg5h3po *   node3      Down     Active        Reachable
```

c) **[ダウン（Down）]** ステータスのノードを削除します。

```
# docker node rm <node-id>
```

次に例を示します。

```
# docker node rm g3mebdulaed2n0cyywjrtum31
```

ステップ 3 Docker スウォームを再起動します。

唯一のライブ ノードにログインしたままで、次の手順を実行します。

a) 既存のスウォームを残します。

```
# docker swarm leave --force
```

b) Orchestrator スクリプト ディレクトリに変更します。

```
# cd /opt/cisco/msc/builds/<build-number>/prodha
```

c) 新しいスウォームを再開します。

```
# ./msc_cfg_init.py
```

このコマンドは、2 つの新しいノードを新しいクラスタに参加させるために使用する必要があるトークンと IP アドレスを返します。

ステップ 4 2 つの新しいノードをクラスタに参加させます。

新しい各ノードで、次の手順を実行します。

a) ノードに SSH 接続します。

b) Orchestrator スクリプト ディレクトリに変更します。

```
# cd /opt/cisco/msc/builds/<build-number>/prodha
```

c) ノードをクラスタに参加させます。

以下のコマンド

- 前の手順で Docker スウォームを再起動したとき、<*token*> を **./msc_cfg_init.py** コマンドから受け取ったトークンに置き換えます。
- <*ip-address*> を前の手順でも受け取った最初のノードの IP アドレスに置換します。

```
# ./msc_cfg_join.py <token> <ip-address>
```

ステップ 5 新しい構成を展開します。

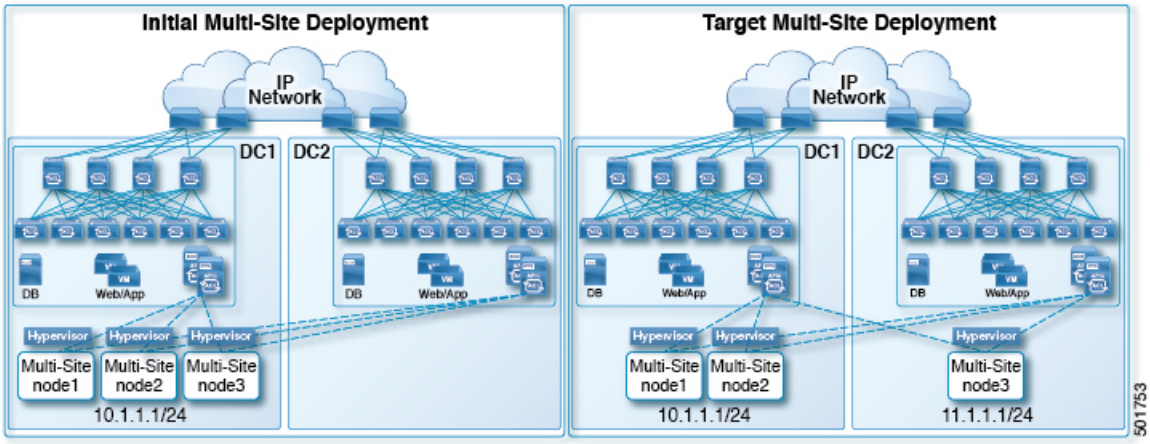
新しいくラスタのいずれかのノードから次のコマンドを実行できます。スクリプトは、同じ /opt/cisco/msc/builds/<*build-number*>/prodha スクリプト ディレクトリにあります。

```
# ./msc_deploy.py
```

別のサブネットへのノードの再配置マルチサイト

このセクションでは、1つ以上のマルチサイト ノードをあるサブネットから別のサブネットに再配置する方法について説明します。これは、マルチサイト が単一のデータ センター内に展開され、ノードを1つ以上のデータ センターに分散することが目標である場合の一般的なタスクです。移行中に冗長性を維持するために、一度に1つのノードを移動することが重要です。

図 *1 : Cisco ACI* マルチサイトの導入



以下の手順例は、管理サブネットが 10.1.1.1/24 サブネットを使用するデータセンター 1 から、管理サブネットが 11.1.1.1/24 サブネットを使用するデータセンター 2 への マルチサイト node3 の再配置を示しています。

ステップ 1 node1 で、node3 を降格します。

例：

```
[root@node1 prodha]# docker node demote node3
```

ステップ 2 node3 仮想マシン（VM）の電源を切ります。

ステップ 3 クラスタから node3 を削除します。

例：

```
[root@node1 prodha]# docker node rm node3
```

ステップ 4 新しい マルチサイト VM（node1 および node2 と同じバージョン）をデータ センターに展開します。新しい IP の詳細を構成し、ホスト名「node3」が割り当てられていることを確認します。

ステップ 5 データ センター 2 の node3 の電源を入れ、node1 と node2 への接続をテストします。

例：

```
[root@node3 prodha]# ping [node1_IP]
[root@node3 prodha]# ping [node2_IP]
```

ステップ 6 node1 で、node1 から参加トークンを取得して、node3 をクラスタに参加させます。

例：

```
[root@node1 prodha]# docker swarm join-token manager
To add a manager to this swarm, run the following command:

docker swarm join --token \
SWMTKN-1-4plaanp2uqpkjm2nidsxg9u7it0dd8hkihwjq9wvrz5heykl2n-98eo0onpacvxrrgf84juczdve \
10.1.1.1:2377

[root@node1 prodha~]#
```

ステップ 7 node3 で、手順 6 の参加トークンを使用してスウォームに参加します。

例：

```
[root@node3 prodha]# docker swarm join --token \
SWMTKN-1-4plaanp2uqpkjm2nidsxg9u7it0dd8hkihwjq9wvrz5heykl2n-98eo0onpacvxrrgf84juczdve \
10.1.1.1:2377
```

ステップ 8 任意のノードで、ノードが正常に稼働していることを確認します。各ノードについて、[STATUS] が Ready、[AVAILABILITY] が Active となっていて、[MANAGER STATUS] が 1 つのみ Leader と表示されることを除いて Reachable となっていることを確認します。

例：

```
[root@node1 ~]# docker node ls
ID                           HOSTNAME   STATUS   AVAILABILITY   MANAGER STATUS
p71zqw77kwnu8z6sr1w0uq2g0    node2      Ready    Active         Leader
q5orng9hd4f0vxneqeehixxwt    node3      Ready    Active         Reachable
ryag1u9ej33pfvrjvqgj4tjr4 *  node1      Ready    Active         Reachable
[root@node1 ~]#
```

ステップ 9 node3 のスウォーム ラベルを更新します。

例：

```
[root@node1 prodha]# docker node update node3 --label-add msc-node=msc-node3
```

ステップ 10 任意のノードで、すべての docker サービスのステータスを確認します。たとえば、1/1（1 のうち 1）または 3/3（3 のうち 3）と記載されていることを確認します。同期には最大 15 分かかる場合があります。

例：

```
[root@node1 ~]# docker service ls
ID             NAME                      MODE         REPLICAS   IMAGE
  PORTS
3kv2qtu3gjmk   msc_kongdb                replicated   1/1        msc-postgres:9.4
5fs0lg9bbbg1   msc_kong                  global       3/3        msc-kong:1.1
jrxade8o2nwn   msc_schemaservice         global       3/3        msc-schemaservice:1.2.0.206
kyq1myno38ry   msc_backupservice         global       3/3        msc-backupservice:1.2.0.206
ltx85gitz85u   msc_executionengine       replicated   1/1        msc-executionengine:1.2.0.206
n4skpiij90t1   msc_ui                    global       3/3        msc-ui:1.2.0.206
  *:80->80/tcp,*:443->443/tcp
o2h8vp3clznd   msc_mongodb1              replicated   1/1        msc-mongo:3.4
q2udphffzb7g   msc_consistencyservice    replicated   1/1        msc-consistencyservice:1.2.0.206
qr1zbd0y18u1   msc_platformservice       global       3/3        msc-platformservice:1.2.0.206
rsb7ki0zxafa   msc_mongodb2              replicated   1/1        msc-mongo:3.4
uiu25mz5h7m9   msc_userservice           global       3/3        msc-userservice:1.2.0.206
xjrp2jbws4pz   msc_auditservice          replicated   1/1        msc-auditservice:1.2.0.206
xtsdns1iy52i   msc_syncengine            replicated   1/1        msc-syncengine:1.2.0.206
ypie99rvielj   msc_mongodb3              replicated   1/1        msc-mongo:3.4
zn03gxpleu1s   msc_siteservice           global       3/3        msc-siteservice:1.2.0.206
[root@node1 ~]#
```

ステップ 11 データセンター 1 で電源を切った元の node3 VM を削除します。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。