仮想インフラストラクチャ マネージャ

仮想インフラストラクチャ マネージャ

UI パス：**[仮想管理（Virtual Management）]>[仮想インフラストラクチャ マネージャ（Virtual Infrastructure Manager）]**

（注） Cisco Nexus Dashboard ファブリックコントローラの仮想マシンのネットワーク可視化機能が有効になっていることを確認します。

1. **[設定（Settings）]>[機能管理（Feature Management）]** を選択し、次のチェックボックスをオンにします。
   - Kubernetes ビジュアライザ
   - VMM ビジュアライザ
   - OpenStack ビジュアライザ
2. [Apply] をクリックします。

次の表では、[仮想インフラストラクチャ マネージャ（Virtual Infrastructure Manager）] ウィンドウに表示されるフィールドについて説明します。

フィールド	説明
[サーバ（Server）]	サーバー IP アドレスを指定します。

フィールド	説明
タイプ	次のいずれかのインスタンスのタイプを指定します。 • vCenter • Kubernetesクラスタ • OpenStack クラスタ
管理対象（Managed）	管理対象または管理対象外のクラスタのステータスを指定します。
ステータス	追加されたクラスタの状態を指定します。
ユーザー（User）	クラスタを作成したユーザーを指定します。
最終更新時刻	クラスタの最終更新時刻を指定します。

（注） **[更新（Refresh）]**アイコンをクリックして、仮想インフラストラクチャ マネージャ テーブルを更新します。

次の表では、[アクション（Actions）] メニューのドロップダウン リストで、[仮想インフラストラクチャ マネージャ（Virtual Infrastracture Manager）] に表示されるアクション項目について説明します。

アクション項目	説明
インスタンスの追加	**[アクション（Actions）]** ドロップダウンリストから **[インスタンスの追加（Add Instance）]** を選択します。詳細については、「インスタンスの追加」を参照してください。 （注） ルート上で同じ IP アドレスを設定していることを確認します。「ルート IP アドレスの設定」を参照してください。
インスタンスの編集	編集するインスタンスを選択します。**[アクション（Actions）]** ドロップダウンリストから **[インスタンスの編集（Edit Instance）]** を選択します。必要な変更を行って、**[保存（Save）]** をクリックします。**[キャンセル（Cancel）]** をクリックして、変更を破棄します。
インスタンスの削除	削除する1つ以上の必要なインスタンスを選択します。**[アクション（Actions）]** ドロップダウンリストから、**[削除（Delete）]** を選択します。[確認（Confirm）] をクリックしてインスタンスを削除します。**[キャンセル（Cancel）]** をクリックしてこの削除を破棄します。

アクション項目	説明
インスタンスの再検出	再検出する1つ以上の必要なインスタンスを選択します。**[アクション（Actions）]** ドロップダウンリストから、**[インスタンスの再検出（Rediscover Instance(s)）]** を選択します。確認メッセージが表示されます。

詳細については、次を参照してください。

Cisco UCS B シリーズ ブレードサーバーのサポート

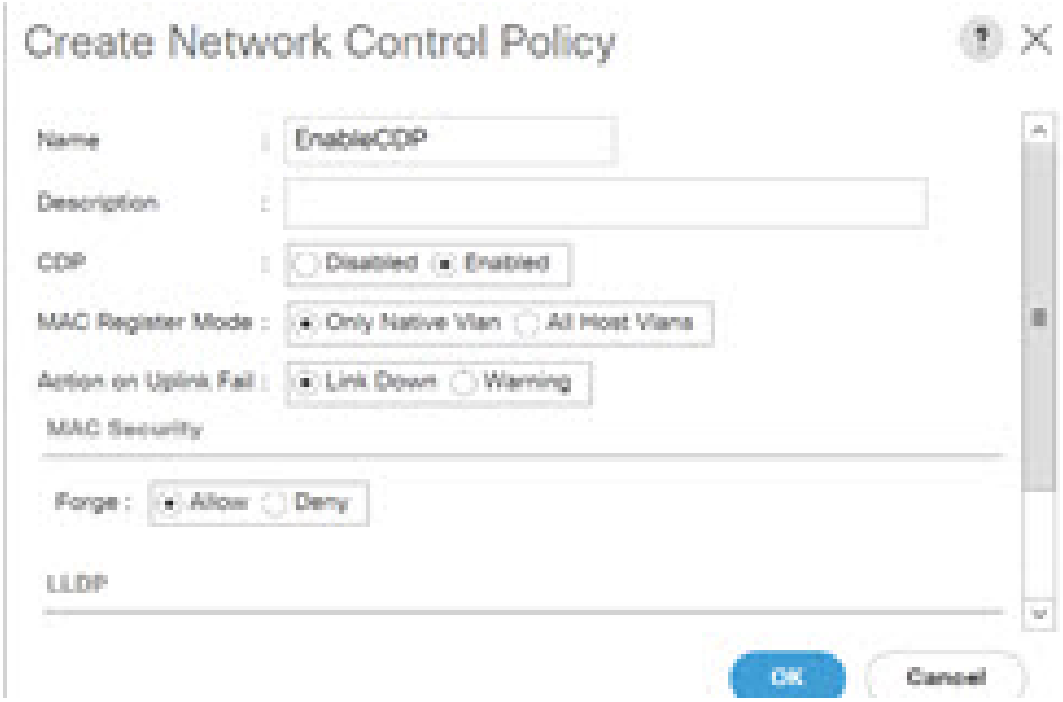
NDFC は、ファブリックインターコネクトの背後にある UCS タイプ B （シャーシ UCS）で実行されているホストをサポートします。この機能を使用するには、Cisco UCSM で vNIC の CDP を有効にする必要があります。

（注） デフォルトでは、CDP は Cisco UCSM で無効になっています。

参考のために、VMM-A と VMM-B の 2 つの VMM について考えてみましょう。Cisco UCS UCS B シリーズ ブレードサーバーの検出後、トポロジに青色の VMM-A と VMM-B がファブリック インターコネクト ノードであることが表示されます。トポロジの例を下図に示します。

UCSM で CDP を有効にするには、次の手順を使用して新しいネットワーク制御ポリシーを作成する必要があります。

1. USCM で、**[LAN]** を選択し、ポリシーを展開します。
2. **[ネットワーク制御ポリシー（Network Control Policies）]** を右クリックして、新しいポリシーを作成します。
3. [名前（Name）] フィールド、にポリシーの名前を **EnableCDP** と入力します。
4. CDP の**有効な**オプションを選択します。



5. **[OK]** をクリックしてポリシーを作成します。

新しいポリシーを ESX NIC に適用するには、次の手順を実行します。

- 更新された vNIC テンプレートを使用している場合は、ESXi vNIC の各 vNIC テンプレートを選択し、[ネットワーク制御ポリシー] ドロップダウンリストから EnableCDP ポリシーを適用します。
- vNIC テンプレートを使用していない場合は、更新されたサービス プロファイル テンプレートを使用します。各サービス プロファイル テンプレートに EnableCDP ポリシーを適用します。
- 1 回限りのサービスプロファイルを使用している場合（つまり、各サーバーが独自のサービスプロファイルを使用している場合）、すべてのサービスプロファイルに移動し、すべての vNIC で EnableCDP ポリシーを有効にする必要があります。

Cisco UCSM の詳細については、『Cisco UCSM ネットワーク管理ガイド』を参照してください。

ルート IP アドレスの設定

IP アドレスを vCenter に追加する前に、Cisco Nexus ダッシュボードで同じ IP アドレスを設定する必要があります。

Cisco Nexus ダッシュボードでルートを設定するには、次の手順を実行します。

手順

ステップ 1 **[インフラストラクチャ（Infrastructure）]** > **[クラスタ設定（Cluster Configuration）]** を選択します。

ステップ 2 **[全般（General）]** タブの **[ルート（Routes）]** カードで、**[編集（Edit）]** アイコンをクリックします。

[ルート（Routes）] ウィンドウが表示されます。

ステップ 3 IP アドレスを設定するには、**[管理ネットワーク ルートの追加（Add Management Network Routes）]** をクリックし、必要な IP アドレスを入力して、**[チェック（check）]** アイコンをクリックします。

ステップ 4 [保存（Save）] をクリックします。

ルート設定は、次の 2 つのシナリオによって管理されます。

1. アプリケーションサーバーである vCenter の場合、通常は管理ネットワーク経由で到達可能です。
2. vCenter によって管理される ESXi サーバーと、K8s インスタンスや OpenStack インスタンスをホストするベアメタルサーバーは、ファブリックネットワークに直接接続されます。したがって、それらはデータネットワークを介して到達可能です。

vCenter の可視化の追加

[仮想的な管理（Virtual Management）] > **[仮想インフラストラクチャ マネージャ（Virtual Infrastructure Manager）]** に表示される **[アクション（Actions）]** メニューのドロップダウンリストで、さまざまなアクションを実行できます。

手順

ステップ 1 **[アクション（Actions）]** **[インスタンスの追加（Add Instance）]** を選択します。

[インスタンスの追加（Add Instance）] ウィンドウが表示されます。

ステップ 2 [タイプの選択（Select Type）]ドロップダウン リストから **[vCenter]** を選択します。

必要な IP アドレスまたはドメイン名とパスワードをそれぞれのフィールドに入力します。

ステップ 3 [Add] をクリックします。

追加された vCenter クラスタは、**[仮想インフラストラクチャ マネージャ（Virtual Infrastructure Manager）]** ウィンドウで表示できます。

ステップ 4 インスタンスを編集するには、必要な vCenter を選択して、**[アクション（Actions）]>[インスタンスの編集（Edit Instance）]** を選択して、**[保存（Save）]** をクリックします。

選択済みの vCenter クラスタのパスワードをアップデートし、ステータスを「管理対象」または「管理対象外」に変更できます。

（注） 管理対象外ステータスの vCenter クラスタの場合、ダッシュボードでトポロジと vCenter クラスタの詳細を表示できません。

ステップ 5 1 つ以上の vCenter クラスタを削除するには、必要な vCenter を選択し、**[アクション（Actions）] > [インスタンスの削除（Delete Instance(s)）]** を選択して、**[変更の確認（Confirm changes）]** をクリックします。

（注） クラスタを削除すると、すべてのデータが削除されます。クラスタは、トポロジビューからも削除されます。

ステップ 6 1 つ以上の vCenter クラスタを再検出するには、必要な vCenter を選択して、**[アクション（Actions）] > [インスタンスの再検出（Rediscover Instance(s)）]** を選択します。

確認メッセージが表示されます。

Kubernetesクラスタ

（注） Cisco Nexusダッシュボード ファブリック コントローラ の K8s クラスタのネットワーク可視化機能が有効になっていることを確認します。

[設定（Settings）] > **[機能管理（Feature Management）]**を選択し、**[Kubernetes ビジュアライザ（Kubernetes Visualizer）]** チェックボックスを選択して、**[適用（Apply）]** をクリックします。

追加された Kubernetes Visualizer の詳細をダッシュボードで表示できます。[ダッシュボード（Dashboard）] > [Kubernetes ポッド（Kubernetes Pods）] に移動します >

NDFC で LLDP を有効にするには、**[設定（Settings）]** > **[サーバー（Server）]** > **[設定（Settings）]** > **[検出（Discovery）]**を選択します。**[LLDP を使用したネイバーリンクディスカバリを有効または無効にします（enable / disable neighbor link discovery using LLDP）]** チェックボックスを選択します。

（注） LLDP は、ベアメタル Kubernetes クラスタにのみ適用されます。

- クラスタノードが接続されているすべてのファブリックスイッチで LLDP 機能が有効になっていることを確認します。（スイッチはスパインまたはリーフスイッチの場合があります）。
- Kubernetes クラスタで、すべてのベアメタルノードで LLDP および SNMP サービスが有効になっていることを確認します。
- Cisco UCS が Intel NIC を使用している場合、FW-LLDP が原因で LLDP ネイバーシップの確立に失敗します。

  回避策：Intel® イーサネットコントローラ（800 および 700 シリーズなど）に基づく選択されたデバイスでは、ファームウェアで実行される LLDP エージェントを無効にします。LLDP を無効にするには、次のコマンドを使用します。

  echo 'lldp stop' > /sys/kernel/debug/i40e/<bus.dev.fn>/command

  特定のインターフェイスの bus.dev.fn を見つけるには、次のコマンドを実行し、インターフェイスに関連付けられた ID を選択します。ID は、以下のサンプル出力で強調表示されています。

```
[ucs1-lnx1]# dmesg | grep enp6s0 [ 12.609679] IPv6: ADDRCONF(NETDEV_UP): enp6s0:
link is not ready [ 12.612287] enic 0000:06:00.0 enp6s0: Link UP [ 12.612646] IPv6:
ADDRCONF(NETDEV_UP): enp6s0: link is not ready [ 12.612665] IPv6:
ADDRCONF(NETDEV_CHANGE): enp6s0: link becomes ready[ucs1-lnx1]#
```

（注） LLDP機能は、ベアメタルクラスタノードが接続されているファブリックスイッチで有効になっています。また、ボーダーゲートウェイスイッチに接続することもできます。

クラスタが検出された後にKubernetesクラスタが接続されているファブリックが検出された場合、トポロジを正しく表示するためにクラスタを再検出する必要があります。

LLDP の設定後にベアメタルベースの Kubernetes クラスタが検出された場合、トポロジを正しく表示するためにベアメタルクラスタを再検出する必要があります。

特定のインターフェイスの bus.dev.fn を見つけるには、次のコマンドを実行し、インターフェイスに関連付けられた ID を選択します。ID は、以下のサンプル出力で強調表示されています。

（注） VM ベースの Kubernetes クラスタを検出または視覚化する場合、最初に、検出される Kubernetes クラスタをホストする VM を管理している vCenter クラスタをオンボードする必要があります。これがないと、Kubernetes クラスタの検出が失敗します。

ルート IP アドレスの設定

Kubernetes クラスタに IP アドレスを追加する前に、Cisco Nexus Dashboard で同じ IP アドレスを設定する必要があります。

Cisco Nexus ダッシュボードでルートを設定するには、次の手順を実行します。

手順

ステップ 1 **[インフラストラクチャ（Infrastructure）]** > **[クラスタ設定（Cluster Configuration）]** を選択します。

ステップ 2 **[全般（General）]** タブの **[ルート（Routes）]** カードで、**[編集（Edit）]** アイコンをクリックします。

[ルート（Routes）] ウィンドウが表示されます。

ステップ 3 IP アドレスを設定するには、**[管理ネットワーク ルートの追加（Add Management Network Routes）]** をクリックし、必要な IP アドレスを入力して、**[チェック（check）]** アイコンをクリックします。

ステップ 4 [保存（Save）] をクリックします。

Kubernetes クラスタの追加

[仮想的な管理（Virtual Management）] > **[仮想インフラストラクチャ マネージャ（Virtual Infrastructure Manager）]**に表示される **[アクション（Actions）]** メニューのドロップダウンリストで、さまざまなアクションを実行できます。

（注） ルート上で同じ IP アドレスを設定していることを確認します。「ルート IP アドレスの設定」を参照してください。

手順

ステップ 1 **[アクション（Actions）]** > **[インスタンスの追加（Add Instance）]**を選択します。

[インスタンスの追加（Add Instance）] ウィンドウが表示されます。

ステップ 2 [タイプの選択（Select Type）] ドロップダウンリストから [Kubernetes クラスタ] を選択します。

ステップ 3 適切なフィールドに **[クラスタ IP アドレス（Cluster IP address）]**、**[ユーザー名（Username）]** を入力します。

ステップ 4 **[CSR の取得（Fetch CSR）]** をクリックして、Kubernetes ビジュアライザ アプリケーションから証明書署名要求（CSR）を取得します。

（注） このオプションは、有効なクラスタ IP アドレスとユーザー名を入力するまで無効になっています。

SSL 証明書を取得していない場合にのみ、**[CSR の取得（Fetch CSR）]** を使用してください。有効な証明書がすでにある場合は、CSR を取得する必要はありません。

[CSR のダウンロード（Download CSR）] をクリックします。証明書の詳細は、ディレクトリ内の **<username>.csr** に保存されます。CSR の内容をファイル **kubereader.csr** に貼り付けます。ここで、kubereader は、Kubernetes に接続する API クライアントのユーザー名です。

CSR ファイル名は命名規則 <<*username*> > に従う必要があります。

（注） 証明書は Kubernetes クラスタで生成されるため、証明書を生成するには Kubernetes 管理者権限が必要です。

付属文書 （13 ページ） を参照して証明書 **genk8clientcert.sh** を生成します。

ステップ 5 Kubernetes クラスタコントローラノードにログインします。

証明書を生成するには、管理者権限が必要です。

ステップ 6 genk8clientcert.sh と kubereader.csr を NDFC サーバーの場所から Kubernetes クラスタコントローラノードにコピーします。

「vnc カットアンドペースト」操作を実行して、すべての文字が正しくコピーされるようにします。

ステップ 7 **genk8sclientcert.sh** スクリプトを使用して、ユーザー名の CSR を生成します。

(k8s-root)# **./genk8sclientcert.sh kubereader 10.x.x.x** ここで、

- kubereader は、Kubernetes に接続する API クライアントのユーザー名です。（手順 3 で定義）。
- 10.x.x.x は NDFC サーバーの IP アドレスです。

同じ場所に 2 つの新しい証明書が生成されます。

- k8s_cluster_ca.crt
- *username_dcnm-IP.crt*

  例：kubereader_10.xxxcrt （ここで、kubereader はユーザー名で、10.x.x.x は NDFC IP アドレスです）

```
dcnm(root)# cat k8s_cluster_ca.crt
```

ステップ 8 cat コマンドを使用して、これら 2 つのファイルから証明書を抽出します。

```
dcnm(root)# cat kubereader_10.x.x.x.crt
dcnm(root)# cat k8s_cluster_ca.crt
```

Cisco NDFC に Kubernetes クラスタを追加するユーザーに、これらの 2 つの証明書を提供します。

ステップ 9 kubereader_10.x.x.x.crt の内容を **[クライアント証明書（Client Certificate）]**フィールドにコピーします。

（注） 「vnc カットアンドペースト」操作を実行して、すべての文字が正しくコピーされるようにします。

ステップ 10 k8s_cluster_ca.crt の内容を **[クライアント証明書（Client Certificate）]** クライアント証明書フィールドにコピーします。

（注） 「vnc カットアンドペースト」操作を実行して、すべての文字が正しくコピーされるようにします。

ステップ 11 [Add] をクリックします。

追加された Kubernetes クラスタは、**[仮想インフラストラクチャ マネージャ（Virtual Infrastructure Manager）]** ウィンドウで表示できます。

（注） ダッシュボードとトポロジウィンドウで、追加された Kubernetes クラスタの詳細を表示できます。**[ダッシュボード（Dashboard）]** > **[Kubernetes ポッド（Kubernetes Pods）]**に移動します。

ステップ 12 Kubernetes クラスタを編集するには、必要なクラスタを選択し、**[アクション（Actions）]** > **[インスタンスの編集（Edit Instance）]**を選択し、[編集（Edit）] をクリックして値を適切に変更します。クラスタとクライアントの証明書を更新できます。Kubernetes クラスタの管理ステー

タスを更新することもできます。管理ステータスの更新を選択した場合、証明書は必要ありません。

（注） 非管理ステータスのkubernetesクラスタの場合、ダッシュボードでトポロジとKubernetesクラスタの詳細を表示できません。

ステップ 13 **[保存（Save）]** をクリックして変更内容を保存するか、または **[キャンセル（Cancel）]** をクリックして変更内容を取り消します。

ステップ 14 1 つ以上の Kubernetes クラスタを削除するには、必要なクラスタを選択し、**[アクション（Actions）] > [インスタンスの削除（Delete Instance(s)）]**の順に選択してクラスタを削除します。

（注） クラスタを削除すると、すべてのデータが削除されます。クラスタは、トポロジビューからも削除されます。

ステップ 15 **[確認（Confirm）]** をクリックしてクラスタを削除します。

ステップ 16 1 つ以上の Kubernetes クラスタを再検出するには、必要な Kubernetes クラスタを選択し、**[アクション（Actions）] > [インスタンスの再検出（Rediscover Instance(s)）]**の順に選択します。

確認メッセージが表示されます。

OpenStack クラスタ

（注） これは、「Nexus ダッシュボード ファブリック コントローラ、リリース 12.0.2a」のプレビュー機能です。ラボセットアップでのみ、ベータ版としてマークされたこの機能を使用することをお勧めします。実稼働環境でこれらのフィーチャを使用しないでください。

（注）

- Cisco Nexus ダッシュ—ボード ファブリック コントローラの Openstack クラスタのネットワーク可視化機能が有効になっていることを確認します。**[設定（Settings）] > [機能管理（Feature Management）]** を選択し、**[Openstack ビジュアライザ（Openstack Visualizer）]** チェックボックスをオンにして、**[適用（Apply）]** をクリックします。
- openstack クラスタを追加するには、vCenter クラスタまたは Kubernetes クラスタ機能を有効にする必要があります。

- NDFC で LLDP を有効にするには、[Web UI] を選択し、**[設定（Settings ）] > [サーバー設定（Server Settings）] > [検出（Discovery）]** を選択します。**[LLDP を使用したネイバーリ**

ンクディスカバリを有効または無効にします（enable / disable neighbor link discovery using LLDP）] チェックボックスを選択します。

- OpenStack クラスタで、すべてのベアメタルノードで LLDP サービスが有効になっていることを確認します。LLDP機能は、ベアメタルクラスタノードが接続されているファブリックスイッチで有効になっています。また、ボーダーゲートウェイスイッチに接続することもできます。
- Intel® イーサネットコントローラに基づく、選択されたデバイス（例：800 および 700 シリーズ）については、ファームウェアで実行される Link Layer Discovery Protocol（LLDP）エージェントを無効にします。同じことを行うには、次のコマンドを使用します。

```
# echo 'lldp stop'>/sys/kernel/debug/i40e/bus.dev.fn/command
```

- 特定のインターフェイスの *bus.dev.fn* を見つけるには、次のコマンドを実行し、インターフェイスに関連付けられた ID を選択します。ID は、以下の出力で強調表示されています。

```
# dmesg | grep eth0
[ 8.269557] enic 0000:6a:00.0 eno5: renamed from eth0
[ 8.436639] i40e 0000:18:00.0 eth0: NIC Link is Up, 40 Gbps Full Duplex, Flow Control:
 None
[ 10.968240] i40e 0000:18:00.0 ens1f0: renamed from eth0
[ 11.498491] ixgbe 0000:01:00.1 eno2: renamed from eth0
```

ルート IP アドレスの設定

Openstack ビジュアライザに IP アドレスを追加する前に、Cisco Nexus ダッシュボードで同じ IP アドレスを設定する必要があります。

Cisco Nexus ダッシュボードでルートを設定するには、次の手順を実行します。

手順

ステップ 1 **[インフラストラクチャ（Infrastructure）]** > **[クラスタ設定（Cluster Configuration）]** を選択します。

ステップ 2 **[全般（General）]** タブの **[ルート（Routes）]** カードで、**[編集（Edit）]** アイコンをクリックします。

[ルート（Routes）] ウィンドウが表示されます。

ステップ 3 IP アドレスを設定するには、**[管理ネットワーク ルートの追加（Add Management Network Routes）]** をクリックし、必要な IP アドレスを入力して、**[チェック（check）]** アイコンをクリックします。

ステップ 4 [保存（Save）] をクリックします。

OpenStack クラスタでの AMQP エンドポイントの設定

- RabbitMQ 通知（oslo.messaging）バス設定は、OpenStack クラスタで完了する必要があります。

  OpenStack Nova サービスで以下の設定変更を行います。パラメータ値を次のように置き換えます。Nova 構成ファイルは次のパスにあります。

  `/etc/nova/nova.conf`

```
[notifications]
notify_on_state_change=vm_and_task_state
default_level=INFO
notification_format=both

[oslo_messaging_notifications]
driver = messagingv2
transport_url=rabbit://guest:guest@X.X.X.X:5672/
topics=notifications
retry=-1
```

（注）

- **transport_url** は、ポート 5672 に IP X.X.X.X を持つサーバーでホストされている RabbitMQ エンドポイントのアドレスです。適切なサーバーの IP アドレスに置き換えます。
- **guest:guest** は、エンドポイントに接続するためのユーザー名とパスワードです。

  また、モニタリング アプリケーション クライアントがポートに接続して通知データを読み取れるように、適切な「iptables」ルールを設定してポート 5672 を開きます。

- OpenStack プラグインは、OpenStack クラスタからリアルタイムの変更通知を受信して処理し、トポロジの説明情報を更新します。リアルタイムの変更通知は、VM の状態の変更（VM の追加、削除、または更新など）およびネットワークの状態の変更（VM と仮想スイッチ間のリンクのシャットダウンなど）に関連しています。
- クラスタノードの電源を入れると、トポロジビューに反映されます。対応するノードがクラスタビューに追加されます。同様に、クラスタノードの電源を切ると、トポロジビューに反映されます。対応するノードがクラスタビューから削除されます。
- OpenStack クラスタ内のノード（コントローラ、コンピューティング、またはストレージ）の追加または削除は、トポロジクラスタビューの NDFC に自動的に反映されます。

付属文書

証明書が正常に生成されると、次のメッセージが表示されます。

```
#!/usr/bin/bash
################################################################
# Title: Script to provision the client CSR and generat the #
#        the client SSL certificate.                           #
################################################################

# Create CSR resource template.
function create_csr_resource() {
    K8SUSER=$1
    DCNM=$2
    FILE=${K8SUSER}_${DCNM}_csr_res.yaml
    echo "
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: ${K8SUSER}_${DCNM}csr
spec:
  groups:
  - system:authenticated
  request: ${BASE64_CSR}
  signerName: kubernetes.io/kube-apiserver-client
  usages:
  - digital signature
  - key encipherment
  - client auth" > $FILE
}

# Create CLUSTER ROLE resource template
function create_cluster_role() {
    K8SUSER=$1
    DCNM=$2
    FILE=${K8SUSER}_${DCNM}_cluster_role_res.yaml
    echo "
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clustrole_${K8SUSER}_${DCNM}
rules:
- apiGroups: [\"\"]
  resources: ["nodes", "namespaces", "pods", "services"]
  verbs: ["get", "list", "watch"]" > $FILE
}

# Create CLUSTER ROLE BINDING template
function create_cluster_role_binding() {
    K8SUSER=$1
    DCNM=$2
    FILE=${K8SUSER}_${DCNM}_cluster_rolebinding_res.yaml
    echo "
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clustrolebind_${K8SUSER}_${DCNM}
roleRef:
  kind: ClusterRole
  name: clustrole_${K8SUSER}_${DCNM}
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: ${K8SUSER}
  apiGroup: rbac.authorization.k8s.io" > $FILE
}

function valid_ip() {
```

```
    local  ip=$1
    local  stat=1

    if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        OIFS=$IFS
        IFS='.'
        ip=($ip)
        IFS=$OIFS
        [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 \
            && ${ip[2]} -le 255 && ${ip[3]} -le 255 ]]
        stat=$?
    fi
    return $stat
}

# Start of the script
if [ "$#" -ne 2 ]; then
    echo "Please provide the username and IP of the DCNM"
    echo
    exit 1
else

    # Check if user have required K8s privileges
    LINUX_USER=$(whoami)
    K8S_CONF_PATH=""
    echo
    echo "Hello ${LINUX_USER}! I am going to help you generate K8s cluster CA and K8s
client certificate."

    if [ ${LINUX_USER} == "root" ] ; then
        # You are root
        if [ ! -d "/root/.kube" ] ; then
            echo
            echo "Directory /root/.kube does not exists."
            echo "User ${LINUX_USER} does not have required K8s privileges"
            echo "Please make sure you are logged into K8s cluster's master node"
            echo
            exit 1
        else
            K8S_CONF_PATH=/${LINUX_USER}/.kube/config
        fi
    else
        # You are not root
        if [ ! -d "/home/${LINUX_USER}/.kube" ] ; then
            echo
            echo "Directory /home/${LINUX_USER}/.kube does not exists."
            echo "User ${LINUX_USER} does not have required K8s privileges"
            echo "Please make sure you are logged into K8s cluster's master node"
            echo
            exit 1
        else
            K8S_CONF_PATH=/home/${LINUX_USER}/.kube/config
        fi
    fi

    # Check if K8s config file exist
    if [ ! -f ${K8S_CONF_PATH} ]; then
        echo
        echo "${K8S_CONF_PATH} file does not exist"
        echo "K8s CA certificate can not be exported"
        echo "Please make sure you are logged into K8s cluster's master node"
        echo
        exit 1
    fi
```

```
K8SUSER=$1
DCNM=$2
K8S_CA_CRT="k8s_cluster_ca.crt"

# Validate the IP address
if valid_ip $DCNM; then
    echo -e
else
    echo "${2} is not a valid IP address"
    echo
    exit 1
fi

# Validate the CSR file format
if [ ${K8SUSER: -4} == ".csr" ]; then
    K8SUSER=${K8SUSER%.csr}
fi

if [ ! -f "./${K8SUSER}.csr" ]; then
    echo
    echo "./${K8SUSER}.csr does not exist"
    echo "CSR file is required for creation of client certificate"
    echo
    exit 1
fi

echo "Generating certificate for ${K8SUSER} for DCNM ${DCNM}"
echo

# Encoding the .csr file in base64
export BASE64_CSR=$(cat ./${K8SUSER}.csr | tr -d '\n')

# Create the CSR resource in K8s cluster
create_csr_resource $K8SUSER $DCNM

# Delete if the CSR resource already exist. We need a fresh one.
kubectl delete csr ${K8SUSER}_${DCNM}csr &> /dev/null
status=$?
if test $status -eq 0
then
    echo "./${K8SUSER}_${DCNM}csr CSR resource already exist, removing it"
else
    echo "./${K8SUSER}_${DCNM}csr CSR resource does not exist, creating it"
fi

# Create the CertificateSigninRequest resource
kubectl apply -f ${K8SUSER}_${DCNM}_csr_res.yaml

# Check the status of the newly created CSR
kubectl get csr

# Approve this CSR
echo "Approving the CSR"
kubectl certificate approve ${K8SUSER}_${DCNM}csr

# Check the status of the newly created CSR
kubectl get csr

# Create role resource definition
kubectl delete clusterrole clustrole_${K8SUSER}_${DCNM} &> /dev/null
create_cluster_role $K8SUSER $DCNM
kubectl apply -f ${K8SUSER}_${DCNM}_cluster_role_res.yaml
```

```
    # Create role binding definition
    kubectl delete clusterrolebinding clustrolebind_${K8SUSER}_${DCNM} &> /dev/null
    create_cluster_role_binding $K8SUSER $DCNM
    kubectl apply -f ${K8SUSER}_${DCNM}_cluster_rolebinding_res.yaml

    # Extract the client certificate
    echo "Extracting the user SSL certificate"
    kubectl get csr ${K8SUSER}_${DCNM}csr -o jsonpath='{.status.certificate}' >
${K8SUSER}_${DCNM}.crt
    echo "" >> ${K8SUSER}_${DCNM}.crt

    # Export the K8s cluster CA cert
    if [ -f ${K8S_CONF_PATH} ]; then
        echo "Exporting K8s CA certificate"
       cat ${K8S_CONF_PATH} | grep certificate-authority-data | awk -F ' ' '{print $2}'
 > ${K8S_CA_CRT}
    fi
    echo
    echo "----------------------------------------------------------------"
    echo "Notes: "
    echo "1. The K8s CA certificate is copied into ${K8S_CA_CRT} file."
    echo "      This to be copied into \"Cluster CA\" field."
    echo "2. The client certificate is copied into ${K8SUSER}_${DCNM}.crt file."
    echo "      This to be copied into \"Client Certificate\" field."
    echo "----------------------------------------------------------------"
    echo
fi
```

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。