Nexus 9000v の展開

この章は、次の項で構成されています。

Nexus 9000v ハイパーバイザ サポート

Nexus 9000v プラットフォーム ファミリの両方のプラットフォームは、サポートされているハイパーバイザで仮想マシンとして実行するように設計されています。基盤となるハイパーバイザの制限により、プラットフォーム機能の一部が制限される場合があります。このセクションでは、サポートのレベルと関連する制限について説明します。

KVM/QEMU 属性

次の表に、KVM/QEMU ハイパーバイザでサポートされる属性を示します。

属性	サポート
QEMU バージョン	4.2.0

属性	サポート
BIOS	OVMF バージョン 16、https://www.kraxel.org/repos/jenkins/edk2/ この URL は、最新の OVMF RPM パッケージ ファイルを含むインデックス ページにアクセスします。ファイルの例は次のとおりです。 `edk2.git-ovmf-x64-0-20200515.1388.g9099dcbd61.noarch.rpm` RPM ユーティリティを使用してパッケージ ファイルをダウンロードして抽出します。パッケージにはいくつかのファイルが含まれています。`OVMF-pure-efi.fd` を見つけて、BIOS ファイルとして使用します。必要に応じて、名前を `bios.bin` に変更できます。
Linux バージョン	Ubuntu 20.0.4
プラットフォーム（Platform）	Nexus 9300v の展開 Nexus 9500v の展開
ラインカード	Nexus 9300v: 1 枚のラインカード Nexus 9500v: 最高 16 枚のラインカード
ライン カード インターフェイス	Nexus 9300v: 最大 64 枚のライン カード インターフェイス Nexus 9500v: 最大 400 枚のライン カード インターフェイス

ESXI 属性

次の表に、ESXI ハイパーバイザでサポートされる属性を示します。

属性	サポート
version	8.0。
プラットフォーム（Platform）	Nexus 9300v の展開 Nexus 9500v の展開
ライン カード	Nexus 9300v: 1 枚のラインカード Nexus 9500v: 最高 16 枚のラインカード

属性	サポート
ラインカード インターフェイス	Nexus 9300v: 最大 9 つのラインカードインターフェイス Nexus 9500v: 最大 9 つのラインカードインターフェイス

VirtualBox 属性

次の表に、VirtualBox ハイパーバイザでサポートされている属性を示します。

属性	サポート
version	7.0
プラットフォーム（Platform）	Nexus 9300v の展開
ライン カード	Nexus 9300v: 1 枚のラインカード
ラインカード インターフェイス	Nexus 9300v: 最大 4 個のライン カード インターフェイス

KVM/QEMU に対する Nexus 9000v 展開ワークフロー

このセクションでは、KVM/QEMU ハイパーバイザに Nexus 9000v プラットフォームを展開するために必要な手順について説明します。次の 3 種類の展開を使用できます。

- 共通展開
- プラットフォーム特有の展開
- インターコネクトの展開

共通展開ワークフロー

KVM/QEMU ハイパーバイザを介して Cisco Nexus 9000v プラットフォームを展開できます。次の表に、KVM/QEMU での Cisco Nexus 9000v 展開でサポートされるパラメータを示します。

パラメータ	例	説明
/path_to/qemu	/usr/bin/qemu-system-x86_64	QEMU の実行可能ファイルへのパス。(さまざまなバージョンの QEMU ソフトウェアを http://wiki.qemu.org/download からダウンロードします。)

パラメータ	例	説明
-nographic	-nographic	Cisco Nexus 9000v プラットフォームはVGAをサポートしていないため、推奨されます。
-bios file	-bios bios.bin	必須。Cisco Nexus 9000v プラットフォームは EFI ブートを使用し、動作するには互換性のあるBIOSイメージが必要です。 ディスク操作のパフォーマンスを向上させるには、SATA コントローラで最新の OVMF BIOS ファイルを使用することをお勧めします。SATA コントローラでは QEMU 2.6 を推奨します。詳細については、http://www.linux-kvm.org/page/OVMF を参照してください。
-smp	-smp 4	Cisco Nexus 9000v プラットフォームは、1 ～ 4 個の vCPU をサポートします（2 ～ 4 個をお勧めします）。
-m memory	-m 10240	メモリ（MB）。
-serial telnet:host:port,server,nowait	-serial telnet:localhost:8888,server,nowait または -serial telnet:server_ip:8888,server,nowait	少なくとも 1 つを指定します。

パラメータ	例	説明
-net ... -net ... または -netdev ... -device ...	-net socket,vlan=x,name=nl_s0,listen=localhost:12000 -net nic, vlan=x, model=e1000, macaddr=aaaa.bbbb.cccc -netdev socket,listen=localhost:12000,id=eth_s_f -device e1000,addr=s.f,netdev=eth_s_f, mac=aaaa.bbbb.cccc, multifunction=on,romfile= または -netdev tap,ifname=tap_s_f,script=no, downscript=no,id=eth_s_f -device e1000,addr=s.f,netdev=eth_s_f, mac=aaaa.bbbb.ccc, multifunction=on,romfile=	net/net または netdev/device のペアは、仮想ネットワーク インターフェイス カード（vNIC）をネットワーク化するためのものです。 _s_fは、PCIスロット番号と機能番号を表します。QEMU2.0以降では、少なくとも20個のPCIスロットと4つの機能をプラグインでき、合計で約80個の vNIC に対応できます。スロットの範囲は 3 ～ 19、関数番号の範囲は 0 ～ 3 です。 mac= オプションは、各 vNIC MAC アドレスの MAC アドレスを VM インターフェイスに渡します。最初の -netdev は、VM の mgmt0 インターフェイスに自動的にマップされます。2 番目の -netdev は e1/1 インターフェースにマップされ、e1/64 の 65 番目まで同様にマップされます。MACアドレスがネットワーク デバイスごとに一意であることを確認します。
-enable-kvm	-enable-kvm	このフラグは、Cisco Nexus 9000v に必要です。
-drive ... -device ... (SATA コントローラの場合)	-device ahci, id=ahci0,bus=pci.0 -drive file=img.qcow2, if=none,id=drive-sata-disk0, format=qcow2 -device ide-hd,bus=ahci0.0、drive=drive-sata-disk0、id=drive-sata-disk0	SATA コントローラに使用するフォーマット。QEMU 2.6.0 で SATA コントローラを使用することをお勧めします。これは、このコントローラがIDEコントローラよりも優れたパフォーマンスを提供するためです。ただし、SATA コントローラをサポートしていない QEMU の早期バージョンがある場合は、IDE コントローラを使用できます。

パラメータ	例	説明
-drive ... media=cdrom	-drive file=cfg.iso,media=cdrom	Cisco Nexus 9000v プラットフォームの起動後に適用されるスイッチ構成ファイルを含む CD-ROM ディスク。 1. テキスト ファイルに名前を付けます（nxos_config.txt）。 2. Linux の mkisofs -o cfg.iso -l --iso-level 2 nxos_config.txt コマンドを使用して、cfg.iso を作成します。

プラットフォーム特有のワークフロー

Cisco Nexus 9500v プラットフォームは、シーケンシャル モードと MAC エンコード モードの 2 つの異なるモードで実行されます。Nexus 9300v および Nexus 9500v のシーケンシャル モードの展開手順は、KVM/QEMU ハイパーバイザでまったく同じです。この場合、両方のプラットフォームの最大インターフェイスは 401 インターフェイスです (1 つの管理ポートまたは 400 のデータ ポート)。

Nexus 9500v は、複数のラインカードでインターフェイス トラフィックをエミュレートします。仮想スイッチは、最大合計 400 のインターフェイスに対して KVM/QEMU 上の単一の VM を使用します。Nexus 9500v の MAC エンコード スキーマに基づいて、KVM/QEMU CLI コマンドが呼び出されたときに、エンコードされたスロットとポート番号を使用して各ネットワーク アダプターの MAC アドレスを指定します。

プラットフォームのインターコネクト

Nexus 9000v プラットフォーム インスタンスまたはその他の仮想プラットフォーム間のインターコネクトは、Linux ブリッジとタップに基づいています。CLI コマンドを呼び出す前に、以下が利用可能であることを確認してください（構成例が提供されています）。

以下の構成例では、ブリッジとタップ インターフェイスを、それぞれ 1 つの管理インターフェイスと 1 つのデータ インターフェイスを持つ 2 つの N9Kv スイッチとともに作成できます。管理インターフェイス「interface mgmt0」は、ブリッジ「mgmt_bridge」を使用して管理ネットワークに接続されています。両方のスイッチからのデータ ポート インターフェイス「interface Eth1/1」は、ブリッジ「interconnect_br」を使用して背中合わせに接続されます。

（注） 必要な最小 QEMU バージョンは、Cisco NX-OS リリース 9.3(3) 以降の 4.2.0 です。

- ブリッジ (ESXi ハイパーバイザーの vSwitch に類似) が作成され、「UP」状態に設定されます。

ブリッジを作成して UP 状態にする Linux コマンド:

sudo brctl addbr *mgmt_bridge*

sudo brctl addbr *interconnect_br*

sudo ifconfig *mgmt_bridge* **up**

sudo ifconfig *interconnect_br* **up**

- タップ インターフェイスは、Nexus 9000v が使用しているインターフェイスの数に基づいて作成されます。

  タップ インターフェイスを作成する Linux コマンド:

  sudo openvpn --mktun –dev *tap_sw1_mgmt*

  sudo openvpn --mktun –dev *tap_sw2_mgmt*

  sudo openvpn --mktun –dev *tap_sw1_eth1_1*

  sudo openvpn --mktun –dev *tap_sw2_eth1_1*

- ブリッジはタップ インターフェイスに接続されます。

  ブリッジをタップ インターフェイスに接続する Linux コマンド:

  sudo brctl addif *mgmt_bridge tap_sw1_mgmt*

  sudo brctl addif *mgmt_bridge tap_sw2_mgmt*

  sudo brctl addif *interconnect_br tap_sw1_eth1_1*

  sudo brctl addif *interconnect_br tap_sw2_eth1_1*

- すべてのタップ インターフェイスは「UP」状態である必要があります。

  タップ インターフェイスを UP 状態にする Linux コマンド:

  sudo ifconfig *tap_sw1_mgmt* **up**

  sudo ifconfig *tap_sw2_mgmt* **up**

  sudo ifconfig *tap_sw1_eth1_1* **up**

  sudo ifconfig *tap_sw2_eth1_1* **up**

- すべてのタップ インターフェイスがブリッジに接続されていることを確認します

  タップ インターフェイスがブリッジに接続されていることを確認する Linux コマンド:

  brctl show

```
bridge name        bridge id           STP enabled        interfaces
interconnect_br    8000.1ade2e11ec42   no                 tap_sw1_eth1_1
                                                          tap_sw2_eth1_1

mgmt_bridge        8000.0a52a9089354   no                 tap_sw1_mgmt
                                                          tap_sw2_mgmt
```

2 つの Nexus 9000v プラットフォームを起動し、それぞれ 1 つのインターフェイスを背中合わせに接続するには、次のコマンドを例として使用できます。接続は、ソケットベースまたはブリッジベースの接続にすることができます。この例では、ブリッジを使用して、管理インター

フェイスのインスタンスと 1 つのデータ ポートを接続します。同様に、コマンド ライン オプションでネット デバイスを追加することで、同じ方法でより多くの Nexus 9000v データ ポートを接続できます。この例では、両方の Nexus 9000v インスタンスの 2 つのインターフェイス (インターフェイス mgmt0 とインターフェイス eth1/1) がそれぞれマッピングされています。

Nexus 9000v の最初のインスタンスの場合:

```
sudo qemu-system-x86_64 -smp 2  -m 8196 -enable-kvm -bios bios.bin
-device i82801b11-bridge,id=dmi-pci-bridge
-device pci-bridge,id=bridge-1,chassis_nr=1,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-2,chassis_nr=2,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-3,chassis_nr=3,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-4,chassis_nr=4,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-5,chassis_nr=5,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-6,chassis_nr=6,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-7,chassis_nr=7,bus=dmi-pci-bridge
-netdev tap,ifname=tap_sw1_mgmt,script=no,downscript=no,id=eth1_1_0
-device e1000,bus=bridge-1,addr=1.0,netdev=eth1_1_0,mac=00:b0:b0:01:aa:bb,multifunction=on,
romfile=
-netdev tap,ifname=tap_sw1_eth1_1,script=no,downscript=no,id=eth1_1_1
-device e1000,bus=bridge-1,addr=1.1,netdev=eth1_1_1,mac=00:b0:b0:01:01:01,multifunction=on,
romfile=
-device ahci,id=ahci0 -drive
file=test1.qcow2,if=none,id=drive-sata-disk0,id=drive-sata-disk0,
format=qcow2
-device ide-hd,bus=ahci0.0,drive=drive-sata-disk0,id=drive-sata-disk0
-serial telnet:localhost:9000,server,nowait -M q35 -daemonize
```

Nexus 9000v の 2 番目のインスタンスの場合:

```
sudo qemu-system-x86_64 -smp 2  -m 8196 -enable-kvm -bios bios.bin
-device i82801b11-bridge,id=dmi-pci-bridge
-device pci-bridge,id=bridge-1,chassis_nr=1,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-2,chassis_nr=2,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-3,chassis_nr=3,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-4,chassis_nr=4,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-5,chassis_nr=5,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-6,chassis_nr=6,bus=dmi-pci-bridge
-device pci-bridge,id=bridge-7,chassis_nr=7,bus=dmi-pci-bridge
-netdev tap,ifname=tap_sw2_mgmt,script=no,downscript=no,id=eth1_1_0
-device e1000,bus=bridge-1,addr=1.0,netdev=eth1_1_0,mac=00:b0:b0:02:aa:bb,multifunction=on,
romfile=
-netdev tap,ifname=tap_sw2_eth1_1,script=no,downscript=no,id=eth1_1_1
-device e1000,bus=bridge-1,addr=1.1,netdev=eth1_1_1,mac=00:b0:b0:02:01:01,multifunction=on,
romfile=
-device ahci,id=ahci0 -drive
file=test2.qcow2,if=none,id=drive-sata-disk0,id=drive-sata-disk0,
format=qcow2
-device ide-hd,bus=ahci0.0,drive=drive-sata-disk0,id=drive-sata-disk0
-serial telnet:localhost:9100,server,nowait -M q35 -daemonize
```

qemu-system-x86_64 以上の KVM コマンドは、Linux のデプロイ方法によっては同等です。呼び出しが成功すると、「telnet localhost 9000」または「telnet localhost 9100」を介して、シリアル コンソールの両方のインスタンスにそれぞれアクセスできるはずです。

LLDP および LACP マルチキャスト固有のパケットのトラフィックを Linux ブリッジ経由で渡すには、各インスタンスに接続するすべてのブリッジで次の値を設定します。

- VM 間の LLDP および LACP 通信を設定します。

  echo 0x4004 > /sys/class/net/br_test/bridge/group_fwd_mask

• Linux ブリッジを介したマルチキャスト パケット フローを許可します。

echo 0 > /sys/devices/virtual/net/br_test/bridge/multicast_snooping

ESXi の Nexus 9000v 展開ワークフロー

このセクションでは、ESXi ハイパーバイザに Nexus 9000v プラットフォームを展開するために必要な手順について説明します。次の 3 種類の展開を使用できます。

• 共通展開

• プラットフォーム特有の展開

• インターコネクトの展開

共通展開ワークフロー

始める前に

次の手順では、分散 OVA を使用して、ESXi ハイパーバイザで Cisco Nexus 9300v または 9500v プラットフォームをプロビジョニングします。

次の状態を確認してください。

• ESXi 8.0 ハイパーバイザをインストールしている

• サーバーと vCenter の両方で実行する ESXi 8.0 の有効なライセンスがあります。

• 配布された OVA ファイルがデスクトップにダウンロードされていること。

手順

ステップ 1 ESXi vCenter にログインします。

ステップ 2 バージョン 8.0 を右クリックして **[OVF テンプレートの展開 (Deploy OVF Template)]** を選択します。

（注）
表示される後続の画面でセルフガイドの指示を実行します。

ステップ 3 **[名前が必要（Need name）]** 画面で、**[ローカル ファイル（Local file）]** を選択し、**[参照（Browse）]** をクリックします。デスクトップからダウンロードした配布 OVA ファイルを選択します。

ステップ 4 **[need name]** 画面で、データセンター (またはフォルダを選択し、VM 名を入力します。

ステップ 5 **[名前が必要（Need name）]** 画面で、仮想マシンを展開する ESXi サーバーを選択し、検証後に **[完了（Finish）]** をクリックします。

ステップ 6 **[名前が必要（Need name）]** 画面で、詳細を確認し、**[次へ（Next）]** をクリックします。

ステップ 7 **[Configure]** 画面で、**[次へ]** をクリックします。

ステップ 8 **[Select Storage]** 画面で、データ ストアを選択し、**[次へ]**をクリックします。

ステップ 9 **[ネットワークの選択（Select Networks）]** 画面で、次の値が選択されていることを確認します。

- 送信元ネットワーク名：mgmt 0
- 宛先ネットワーク：ラボ管理 LAN vSwitch

ラボ管理 LAN vSwitch として他の vNIC 宛先を選択しないでください。そうしないと、Cisco Nexus 9000v データ ポートが物理スイッチと競合するため、管理接続の問題が発生します。

ステップ 10 **[Ready to Complete]** 画面で、**[Finish]** をクリックし、プロセスが完了するまで待ちます。

ステップ 11 **[仮想ハードウェア]** タブで、**[シリアル ポート 1]** を選択します。シリアル ポート タイプについては、**[ネットワークの使用]** パネルを選択し、次のオプションを選択します。

- 方向 - サーバー
- ポート URL - telnet://0.0.0.0:1000。1000 はこのサーバーの一意のポート番号です。

（注）
Nexus 9000v は、E1000 ネットワーク アダプタのみをサポートします。ネットワーク アダプターを追加するときは、アダプターの種類が E1000 であることを確認します。

ステップ 12 **[VM Options]** タブで、**[Boot Options]** パネルを選択し、**[EFI]** を選択します。

ステップ 13 **[VM Options]** タブで、**[Advance]** パネルを選択し、**[Edit Configuration]** 画面で、**[Add Configuration Params]** オプションを使用して次の値を追加します。

- 名前 - efi.serialconsole.enabled
- 値 - TRUE

[OK] をクリックして、VGA とシリアル コンソール モードの両方で起動プロセスを表示します。

（注）
Nexus 9000v プラットフォームでは、スイッチ プロンプトにアクセスするためにシリアル コンソールをプロビジョニングする必要があります (ただし、最初の grub ブート メッセージの一部は VGA コンソールに表示されます)。シリアル コンソールが VM に正しくプロビジョニングされていることを確認します。「efi.serialconsole.enabled=TRUE」がプロビジョニングされている場合、VGA またはシリアル コンソールから「**Nexus9000v のイメージ署名検証が実行されていません**」が表示された後、ブートアップが成功するとカーネル ブート メッセージが表示されます。

ステップ 14 仮想マシンの電源をオンにします。

プラットフォーム特有のワークフロー

Cisco Nexus 9500v は、シーケンシャル モードと MAC エンコード モードの 2 つの異なるモードで動作します。Nexus 9300v および Nexus 9500v のシーケンシャル モードの展開手順は、ESXi ハイパーバイザでまったく同じです。両方のプラットフォーム タイプのインターフェイスの最

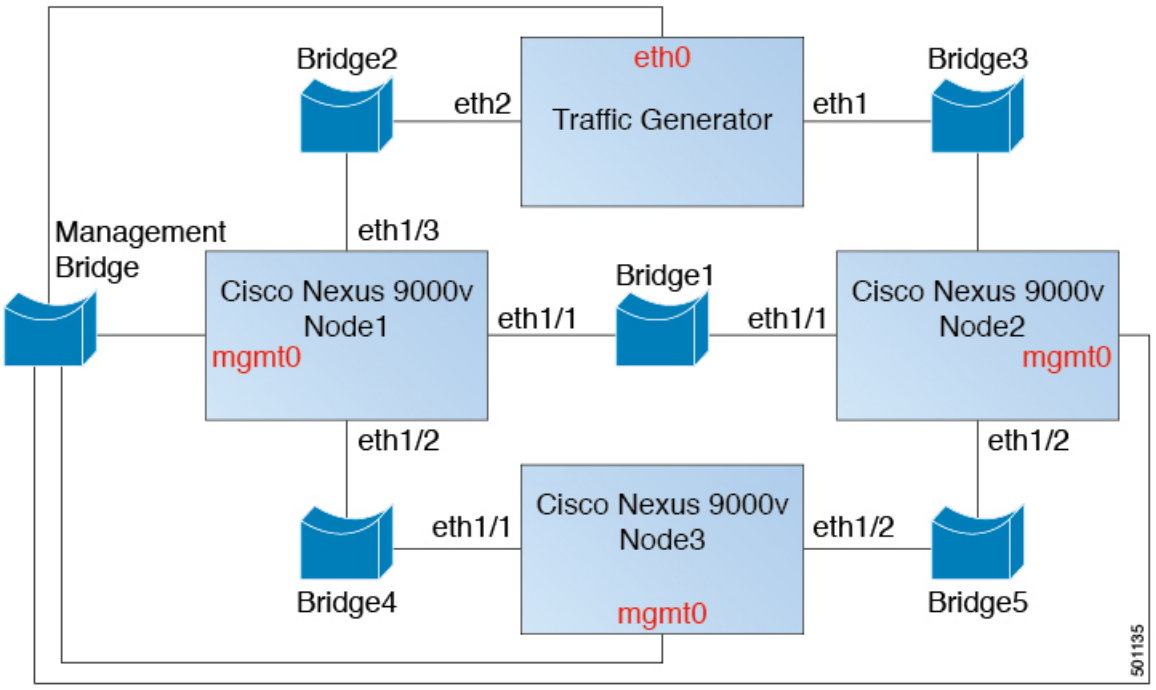
大数は 10（1 つの管理ポートと 9 つのデータ ポート）です。これはハイパーバイザの制限です。

Nexus 9500v は、インターフェイスの総数が 10 に制限されている場合でも、ESXi ハイパーバイザ上の単一の VM で複数回線カードインターフェイス トラフィックをエミュレートします。Nexus 9500v の MAC エンコード スキーマの使用を選択した場合は、エミュレートされているスロットとポートに一致するように各ネットワーク アダプタの MAC アドレスを変更します。

プラットフォームのインターコネクト

Nexus 9300v と Nexus 9500v、またはその他の仮想プラットフォーム間のネットワークは、ESXi ハイパーバイザのブリッジとしての vSwitch に基づいています。さまざまな顧客のユース ケースをシミュレートするように設計された任意のトポロジを使用できます。

図 *1 : EXSi* を介した *Cisco Nexus 9000v* プラットフォームのインターコネクト



Vagrant 用 Nexus 9000v 展開ワークフロー

このセクションでは、Vagrant ハイパーバイザーに Nexus 9000v プラットフォームを展開するために必要な手順について説明します。次の 3 種類の展開を使用できます。

- 共通展開
- プラットフォーム特有の展開
- インターコネクトの展開

共通展開ワークフロー

Vagrant/VBox 環境に Cisco Nexus 9300v を展開することはできません。仮想 artifacts.boxファイルは、配布でのみ入手できます。

プラットフォーム特有のワークフロー

nexus9300v.9.3.3.IDI9.0.XXX.box を VirtualBox に展開します。Vagrant/Vbox の使用に関する次のカスタマイズ ガイドラインと警告を参照してください。

- Vagrant ファイルでのユーザーのカスタマイズは必要ありません。
- Windows の名前付きパイプを変更する必要はありません。Mac または Windows の両方で、デフォルトのポート 2023 を使用してシリアル コンソールにアクセスします。必要に応じて、このシリアル コンソールを **telnet localhost 2023** 経由で使用して、スイッチの起動プロセスを監視します。
- 標準のボックス プロセスは、他のアプライアンスのディストリビューションと同様に使用されます。ベース ボックス名を使用して VM を起動するだけです。
- ボックス名は、Vagrant ファイルの **config.vm.box** フィールドを使用して、「base」以外の別の名前に変更できます。
- ブートストラップ構成は、リリース イメージ ファイルから **.box** の既存の汎用構成以外に、スイッチに別の構成を適用する場合に可能です。この場合、**vb.customize pre-boot** を使用します。例：

```
vb.customize "pre-boot", [
             "storageattach", :id,
             "--storagectl", "SATA",
             "--port", "1",
             "--device", "0",
             "--type", "dvddrive",
             "--medium", "../common/nxosv_config.iso",
```

- **config.vm.base_mac** フィールドを使用して、VM インターフェイスの MAC アドレスをカスタマイズします。この変更は、**vagrant up** CLI コマンドを入力する前、および **vagrant init** CLI コマンドを入力した後に実行する必要があります。**vagrant up** CLI コマンドの入力後、または VM の作成後に MAC アドレスを変更する場合は、ボックス コマンドを使用して VM を変更します。

Vagrant での Sync フォルダのサポート

リリース 10.1(1) 以降、Nexus 9300v は、ホスト マシン上のディレクトリ/フォルダを Nexus 9300v マシンと共有できる Vagrant 同期フォルダをサポートします。Vagrant スクリプトの **vagrant up** コマンドは仮想ボックスにログインし、Vagrantfile のユーザー構成に基づいてディレクトリをマウントします。デフォルトでは、Vagrant スクリプトは *vagrant* ユーザー名を使用し、bash がログイン シェルであることを想定しています。この機能を容易にするために、事前設定された *vagrant* ユーザー名のデフォルトのログイン シェルが bash に変更されました。

ただし、デフォルトのシェル (ユーザー *vagrant* の場合) を、Nexus または Vagrantfile で明示的に設定して NX-OS CLI に変更するオプションがあります。

デフォルトでは、Vagrant はホストの現在の作業ディレクトリを Nexus 9300v の `directory/vagrant` にマウントします。ホスト上の現在のフォルダを Nexus 9300v と共有したくない場合は、Vagrantfile に次の行を含める必要があります。

```
config.vm.synced_folder ".", "/vagrant", disabled: true
```

サンプル Vagrantfile - ホスト フォルダを共有する場合、たとえば、Nexus 9300v の **/bootflash/home/vagrant** の **/home/james/my_shared_folder/** ：

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
  # The most common configuration options are documented and commented below.
  # For a complete reference, please see the online documentation at
  # https://docs.vagrantup.com.

  # Every Vagrant development environment requires a box. You can search for
  # boxes at https://vagrantcloud.com/search.

config.vm.define "n9kv1" do |n9kv1|

        n9kv1.vm.box = "10.1.1"

        n9kv1.ssh.insert_key = false
        n9kv1.vm.boot_timeout = 600

        if Vagrant.has_plugin?("vagrant-vbguest")
          config.vbguest.auto_update = false
        end

        config.vm.synced_folder ".", "/vagrant", disabled: true
       config.vm.synced_folder "/home/james/my_shared_folder"  "/bootflash/home/vagrant/"

        config.vm.box_check_update = false

 end
```

以下に、Nexus 9300v プラットフォーム固有の展開例を示します。

```
vagrant box add 10.1.1 nexus9300v.10.1.1.box

$ vagrant init 10.1.1
$ vagrant up

Bringing machine 'n9kv1' up with 'virtualbox' provider...
==> n9kv1: Importing base box '10.1.1'...
==> n9kv1: Matching MAC address for NAT networking...
==> n9kv1: Setting the name of the VM: vagrant_n9kv1_1605848223701_17342
==> n9kv1: Clearing any previously set network interfaces...
==> n9kv1: Preparing network interfaces based on configuration...
    n9kv1: Adapter 1: nat
==> n9kv1: Forwarding ports...
    n9kv1: 22 (guest) => 2222 (host) (adapter 1)
==> n9kv1: Booting VM...
==> n9kv1: Waiting for machine to boot. This may take a few minutes...
    n9kv1: SSH address: 127.0.0.1:2222
    n9kv1: SSH username: vagrant
    n9kv1: SSH auth method: private key
```

```
==> n9kv1: Machine booted and ready!
==> n9kv1: Checking for guest additions in VM...
    n9kv1: The guest additions on this VM do not match the installed version of
    n9kv1: VirtualBox! In most cases this is fine, but in rare cases it can
    n9kv1: prevent things such as shared folders from working properly. If you see
    n9kv1: shared folder errors, please make sure the guest additions within the
    n9kv1: virtual machine match the version of VirtualBox you have installed on
    n9kv1: your host and reload your VM.
    n9kv1:
    n9kv1: Guest Additions Version: 5.2.18 r123745
    n9kv1: VirtualBox Version: 6.1
==> n9kv1: Mounting shared folders...
    n9kv1: /bootflash/home/vagrant => /home/james/my_shared_folder


$ vagrant ssh

-bash-4.4$
```

デフォルトのシェルの NX-OS CLI への変更

NX-OS CLI にログインする必要がある場合は、次のいずれかのオプションを使用します。

- ログインするたびに bash プロンプトで **vsh** コマンドを手動で実行します。
- 以下に示すように、Nexus 9300v 仮想ボックスに事前にパッケージ化されたスクリプトを使用して、Vagrantfile から実行することができます。

```
        config.vm.synced_folder ".", "/vagrant", disabled: true
        config.vm.synced_folder "/home/james/my_shared_folder"
"/bootflash/home/vagrant/"
        config.vm.box_check_update = false

             config.vm.provision "shell", inline: "vsh -r
/var/tmp/set_vsh_as_default.cmd"
```

- ユーザー名 *vagrant* の代わりにユーザー名 *admin* でログインできます (**vagrant ssh** コマンドを使用すると、デフォルトでユーザー名 *vagrant* が使用されます)。

```
ssh -p 2222 admin@127.0.0.1
```

Nexus 9300v での Ansible の使用

Vagrant は、Ansible、Shell スクリプト、Ruby スクリプト、Puppet、Chef、Docker、Salt などのさまざまなプロビジョナーを使用して、ボックスの構成と管理をサポートする汎用オーケストレーターです。

Vagrant ファイルには、1 つ (または複数) のプロビジョナーのセクションとその構成が含まれている場合があります。Ansible の例をここに示します。

```
n9kv1.vm.provision "ansible" do |ansible|
         ansible.playbook = "n9kv1.yml"
         ansible.compatibility_mode = "2.0"
end
```

これらのプロビジョナーは、仮想ボックスが起動するたびに、または **vagrant provision** コマンドまたは **vagrant provision –provision-wth** コマンドを使用して手動でトリガーされたときに、自動的にトリガーされます。Ansible が仮想ボックスにログインして NX-OS CLI を実行するに

は、Ansible ホスト構成ファイルでログイン資格情報を提供します。Ansible はログイン後に NX-OS CLI が表示されることを想定しているため、事前構成されたユーザー名 *admin* を使用するか、新しいユーザー名を手動で作成して、Ansible ホスト構成ファイルで使用できます。

VM のシャットダウン

次を使用して VM をシャットダウンします。

```
$ vagrant halt -f
==> default: Forcing shutdown of VM...
```

クリーンアップのために VM を破棄する

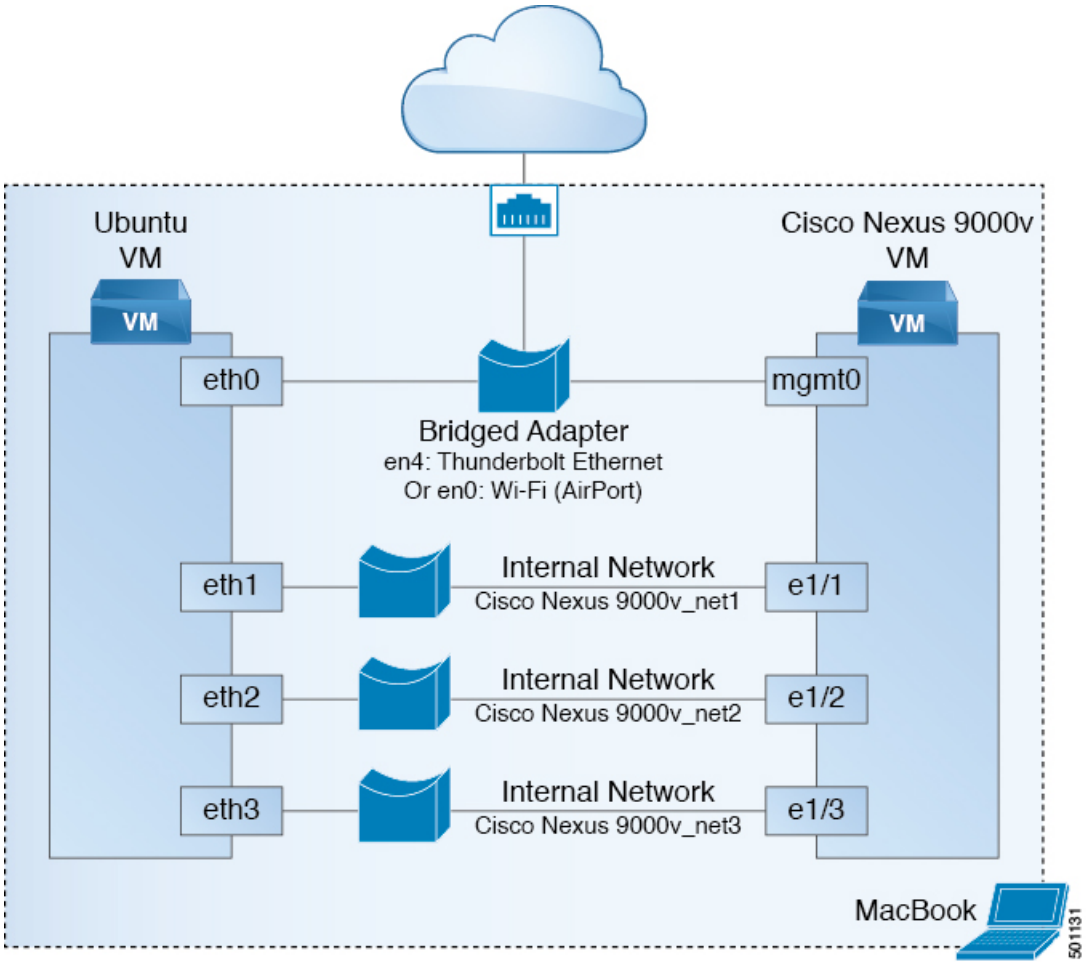
VM インスタンスを完全に削除する場合は、次を使用します。

```
$ vagrant box remove base
Removing box 'base' (v0) with provider 'virtualbox'...
                  $ vagrant destroy
                      default: Are you sure you want to destroy the 'default' VM? [y/N]
 y
==> default: Destroying VM and associated drives..
```

プラットフォームのインターコネクト

Nexus 9300v と他の仮想プラットフォーム間のネットワークは、VBox 内部ネットワークに基づいています。次の接続図を参照してください。

図 *2 : Vagrant VM* を介した *Cisco Nexus 9000v* プラットフォームのインターコネクト



イメージ アップグレードのワークフロー

このセクションでは、Cisco Nexus 9000v プラットフォームの一般的なアップグレード手順について説明します。

新しいアーティファクトからの展開

環境に応じて、適切な仮想アーティファクトを使用し、次のセクションのいずれかを参照して VM を展開します。

- KVM/QEMU に対する Nexus 9000v 展開ワークフロー （3 ページ）
- ESXi の Nexus 9000v 展開ワークフロー （9 ページ）
- Vagrant 用 Nexus 9000v 展開ワークフロー （11 ページ）

新しい NX-OS イメージからのアップグレード

Nexus 9300v のアップグレードは、Cisco Nexus 9000v リリース 9.3(1) 以降の仮想アーティファクトで作成された VM からのみ許可されます。アップグレードする前に、ブートフラッシュに 400Mb 以上の新しい NX-OS バイナリ イメージがあることを確認してください。アップグレードするには、新しいバイナリをブートフラッシュにコピーしてから、標準の NX-OS ワークフローを使用してアップグレードします (例: 'install all nxos bootflash:///<nxos.bin> ')。

これはプラットフォームの最初のリリースであるため、Nexus 9500v のアップグレードはサポートされていません。

Nexus 9300v および 9500v lite の場合、以前のバイナリ イメージから Lite バイナリ イメージへの ISSU はサポートされていません。コールドブートを使用してイメージを起動できる場合でも、最初に以前の構成を削除してから、lite バイナリをインストールします。

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。