KVM 環境でのインストール

Red Hat Enterprise Linux（RHEL）は、Red Hat が製造するエンタープライズ仮想化製品です。カーネルベース仮想マシン（KVM）をベースとする RHEL は、仮想化拡張機能を搭載した x86 ハードウェア上の Linux 向けの、オープンソース完全仮想化ソリューションです。

Cisco IOS XE 17.12.1 リリース以降、Cisco Catalyst 8000V は、RHEL 8.4 KVM ハイパーバイザ上の Intel x550 NIC を搭載した Intel Atom® C3000 プロセッサ（Denverton）CPU ベースのサーバーでもサポートされます。異なるバージョンのハイパーバイザ オペレーティング システムを使用する他の x86 CPU で Cisco Catalyst 8000V を実行できますが、サポートはこれらのリストされているバージョンでのみ使用できます。

Cisco Catalyst 8000V 仮想ルータを Red Hat KVM 仮想化の仮想マシンとしてインストールできます。インストール手順では、最初に VM を手動で作成します。その後に、.iso ファイルまたは qcow2 ファイルを使用したインストールが続きます。次を使用して、KVM 環境に Cisco Catalyst 8000V をインストールできます。

- **GUI ツール**：KVM サーバーに virt-manager RPM パッケージをダウンロードしてインストールします。virt-manager は、仮想マシンを管理するためのデスクトップ ユーザー インターフェイスです。GUI を使用したインストールは、推奨されるインストール方法です。
- **コマンド ライン インターフェイス**：このインストール方法では、コマンド ライン インターフェイスを使用して Cisco Catalyst 8000V VM をインストールします。

（注） KVM 環境での OVA テンプレートの展開はサポートされていません。

Cisco Catalyst 8000V は、KVM 実装で Virtio vNIC タイプをサポートします。KVM は最大 26 の vNIC をサポートします。

KVM のインストール要件

Cisco IOS XE 17.4.x リリース以降を使用する Cisco Catalyst 8000V の KVM 要件は次のとおりです。

表 *1 : KVM* バージョン（*Red Hat Enterprise Linux* をベースとした *Linux KVM*）

Cisco IOS XE リリース	**KVM** バージョン
Cisco IOS XE 17.15.1 リリース	Red Hat Enterprise Linux 9.2 および 8.4 をベースとした Linux KVM を推奨。
Cisco IOS XE 17.14.x リリース Cisco IOS XE 17.13.x リリース Cisco IOS XE 17.12.x リリース Cisco IOS XE 17.11.x リリース Cisco IOS XE 17.10.x リリース Cisco IOS XE 17.9.x リリース Cisco IOS XE 17.8.x リリース Cisco IOS XE 17.7.x リリース	Red Hat Enterprise Linux 7.7 および 8.4 をベースとした Linux KVM を推奨。
Cisco IOS XE 17.4.x リリース Cisco IOS XE 17.5.x リリース Cisco IOS XE 17.6.x リリース	Red Hat Enterprise Linux 7.5 および 7.7 をベースとした Linux KVM を推奨。

表 *2 : KVM* バージョン（*SUSE Linux® Enterprise Server*）

Cisco IOS XE リリース	**KVM** バージョン
Cisco IOS XE 17.14.x リリース Cisco IOS XE 17.13.x リリース Cisco IOS XE 17.12.x リリース Cisco IOS XE 17.11.x リリース Cisco IOS XE 17.10.x リリース Cisco IOS XE 17.9.x リリース Cisco IOS XE 17.6.3 リリース	SUSE Linux Enterprise Server バージョン 15 SP3 をサポート
Cisco IOS XE 17.15.1 リリース	SUSE Linux Enterprise Server バージョン 15 SP5 をサポート

表 *3:* サポートされている *VNIC*

vNIC	サポートされているリリース
Virtio	Cisco IOS XE リリース 17.4.1 以降
ixgbevf	Cisco IOS XE リリース 17.4.1 以降
i40evf	Cisco IOS XE リリース 17.4.1 から Cisco IOS XE 17.8.x リリース
iavf	Cisco IOS XE リリース 17.9.1 以降
ConnectX-5VF	Cisco IOS XE リリース 17.9.1 以降
Ixgbe	Cisco IOS XE リリース 17.10.1 以降

（注） i40evf ドライバを備えた vNIC を使用する場合、物理 VLAN の最大数は 512 に制限され、すべての（仮想機能）VF で共有されます。VF の VLAN の数は、信頼されていない VF のホスト（PF）ドライバによってさらに制限される場合があります。最新の Intel i40e PF ドライバは、信頼されていない VF を最大 8 つの VLAN/サブインターフェイスに制限します。

VM インスタンスあたりのサポートされる最大 vNIC 数：26

- vCPU。次の vCPU 設定がサポートされています。
  - 1 vCPU：最低 4 GB の RAM 割り当てが必要
  - 2 vCPU：最低 4 GB の RAM 割り当てが必要
  - 4 vCPU：最低 4 GB の RAM 割り当てが必要
  - 8 vCPU：最低 8 GB の RAM 割り当てが必要
  - 16 vCPU：最低 8 GB の RAM 割り当てが必要（Cisco IOS XE 17.11.1a 以降でサポート）
- 仮想 CPU コア：1 vCPU が必要
- 仮想ハードディスクサイズ：最低 8 GB
- 仮想 CD/DVD ドライブのインストール（.iso ファイルを使用してインストールする場合、または ISO を介してデイゼロ設定を提供する場合にのみ適用）：必須

KVM インスタンスの作成

GUI ツールを使用した VM の作成

始める前に

KVM サーバーに virt-manager RPM パッケージをダウンロードしてインストールします。

シスコのソフトウェア ダウンロード ページから .qcow2 イメージまたは .iso イメージをダウンロードし、ファイルをローカルデバイスまたはネットワークデバイスにコピーします。

ステップ 1 virt-manager GUI を起動します。

ステップ 2 [Create a New Virtual Machine] をクリックします。

ステップ 3 次のいずれかを実行します。

a) .qcow2 ファイルをダウンロードした場合は、[Import Existing Disk Image] を選択します。

b) .iso ファイルをダウンロードした場合は、[Local Install Media (ISO Image or CDROM)] を選択します。

ステップ 4 Cisco Catalyst 8000V qcow2 または iso ファイルの場所を選択します。

ステップ 5 メモリと CPU のパラメータを設定します。

ステップ 6 仮想マシンのストレージを設定します。

ステップ 7 （オプション）VM を作成する前にハードウェアを追加するには、[Customize configuration before install] を選択します。[Add Hardware] ボタンが表示されます。追加ディスクやシリアル ポート インターフェイスなど、さまざまなハードウェアオプションを追加するには、このボタンをクリックします。

ステップ 8 （オプション）シリアルコンソールを追加するには、シリアルコンソールの追加 （4 ページ）に記載されている手順に従います。

ステップ 9 （オプション）VMを作成する前に設定をカスタマイズする場合は、VMを作成する前の設定のカスタマイズ （5 ページ）を参照してください。

ステップ 10 [Finish] をクリックします。

ステップ 11 Cisco Catalyst 8000V コンソールにアクセスするには、次のいずれかのアクションを実行します。

a) 仮想コンソールを使用している場合は、VM インスタンスをダブルクリックして VM コンソールにアクセスします。

b) シリアルコンソールを使用している場合は、「Booting the Cisco Catalyst 8000V and Accessing the Console」を参照してください。

シリアルコンソールの追加

このタスクを実行し、シリアルコンソールを追加して Cisco Catalyst 8000V インスタンスへのアクセスを有効にします。

ステップ 1 [Add Hardware] をクリックします。
ステップ 2 メニューから [Serial] オプションを選択します。
ステップ 3 [Device Type] ドロップダウンメニューから、[TCP net console (tcp)] を選択します。
ステップ 4 ポート番号を指定し、[Use Telnet] チェックボックスをオンにします。
ステップ 5 [Finish] をクリックします。
ステップ 6 必要なハードウェアをすべて追加したら、[Begin Installation] をクリックします。

VM を作成する前の設定のカスタマイズ

始める前に

.qcow2 または .iso イメージを使用して GUI ツールを使用した VM の作成 （4 ページ） タスクを実行します。[Customize configuration before install] オプションをオンにしてから、[Finish] をクリックします。[Add Hardware] ボタンが表示されます。

[Customize Configuration Before Install] オプションを選択した後のオプションの手順を説明するこの手順に進みます。

ステップ 1 [Add Hardware] をクリックします。
ステップ 2 [Storage] オプションを選択します。
ステップ 3 [Select Managed Or Other Existing Storage] チェックボックスをオンにします。
ステップ 4 [Browse] をクリックし、**c8000v_config.iso** の場所に移動します。この手順は、デイゼロまたはブートストラップ設定を追加する場合にのみ適用されます。
ステップ 5 [Device-type] ドロップダウンメニューから、[IDE CDROM] を選択します。
ステップ 6 [Finish] をクリックします。
ステップ 7 必要なハードウェアをすべて追加したら、[Begin Installation] をクリックします。

ブートストラップ設定を実行するには、デイゼロ設定を参照してください。

CLI を使用した VM の作成

- KVM サーバーに virt-install RPM パッケージをダウンロードしてインストールします。
- Cisco Catalyst 8000V ソフトウェア インストール イメージ パッケージから **qcow2** イメージをダウンロードし、ローカルデバイスまたはネットワークデバイスにコピーします。

ステップ 1 .qcow2 イメージの VM を作成するには、virt-install コマンドを使用してインスタンスを作成し、起動します。次の構文を使用します。

例：

```
virt-install                    \
     --connect=qemu:///system   \
     --name=my_c8kv_vm           \
     --os-type=linux            \
     --os-variant=rhel4         \
     --arch=x86_64              \
     --cpu host                 \
     --vcpus=1,sockets=1,cores=1,threads=1   \
     --hvm                      \
     --ram=4096                 \
     --import                   \
     --disk path=<path_to_c8000v_qcow2>,bus=ide,format=qcow2   \
     --network bridge=virbr0,model=virtio                        \
     --noreboot
```

ステップ 2 .iso イメージの VM を作成するには、次の手順を実行します。

a) **qemu-img** コマンドを使用し、**.qcow2** 形式で 8 G のディスクイメージを作成します。

例：

```
qemu-img create -f qcow2 c8000v_disk.qcow2 8G
```

b) **virt-install** コマンドを使用して、Cisco Catalyst 8000V インスタンスをインストールします。これには、新しい VM を作成するための適切な権限が必要です。次の例では、4G の RAM、1 つのネットワークインターフェイス、および 1 つのシリアルポートを備えた 1 つの vCPU Cisco Catalyst 8000V を作成します。

例：

```
virt-install                \
 --connect=qemu:///system   \
 --name=my_c8000v_vm          \
 --description "Test VM"    \
 --os-type=linux            \
 --os-variant=rhel4         \
 --arch=x86_64              \
 --cpu host                 \
 --vcpus=1,sockets=1,cores=1,threads=1    \
 --hvm                      \
 --ram=4096                 \
 --cdrom=<path_to_c8000v_iso>          \
 --disk path=c8000v_disk.qcow2,bus=virtio,size=8,sparse=false,cache=none,format=qcow2   \
 --network bridge=virbr0,model=virtio     \
 --noreboot
```

virt-install コマンドで新しい VM インスタンスを作成し、Cisco Catalyst 8000V は指定したディスクファイルにイメージをインストールします。

インストールが完了すると、Cisco Catalyst 8000V VM はシャットダウンされます。**virsh start** コマンドを実行することで VM を起動できます。

（注） c8000v_config.iso ディスクイメージを使用してデイゼロ設定を指定する場合は、**virt-install** コマンドにパラメータを追加します。たとえば、`--disk path=/my/path/c8000v_config.iso,device=cdrom,bus=ide` です。詳細については、デイゼロ設定を参照してください。

Red Hat Enterprise Linux - ホストモードの設定

Red Hat Enterprise Linux に固有の問題により、**virt-install** を使用して Red Hat Enterprise Linux 環境で Cisco Catalyst 8000V を起動する場合は、次のようにホストモードを設定します。

- Red Hat Enterprise Linux 6 の場合は次を使用します。

```
--cpu host
```

- Red Hat Enterprise Linux 7 の場合は次を使用します。

```
--cpu host-model
```

VM のクローン作成

問題

KVM 環境では、**virt-manager** 仮想マシンマネージャを使用して Cisco Catalyst 8000V 仮想マシンの複製を作成すると、Cisco Catalyst 8000V 仮想マシンが起動できない場合があります。この問題は、**virt-manager** によって作成され複製されたイメージのサイズが、元の Cisco Catalyst 8000V VM イメージと比較して増加したことが原因で発生します。余分なバイト（KB 範囲）が原因で起動が失敗します。

回避策

次の 3 つの回避策があります。

- **virt-clone** コマンドを使用して、Cisco Catalyst 8000V VM イメージを複製します。
- ブートアップ中に **virt-manager** によって作成された複製の Cisco Catalyst 8000V VM イメージの場合は、packages.conf の代わりに起動する GOLDEN イメージを選択します。
- [Create a new virtual machine] ウィンドウで、新しい Cisco Catalyst 8000V VM を作成する前に、[Allocate Entire Disk Now] の選択を解除します。これにより、複製された Cisco Catalyst 8000V VM イメージが起動できるようになります。ただし、この回避策はネストされた複製をサポートしていません。この方法は、最初に複製された Cisco Catalyst 8000V VM イメージでのみ使用します。

KVM 構成のパフォーマンスの向上

KVM ホストの一部の設定を変更することによって、KVM 環境内で実行されている Cisco Catalyst 8000V のパフォーマンスを向上させることができます。これらの設定は、Cisco Catalyst 8000V インスタンスの IOS XE の構成時の設定とは無関係です。

KVM 設定のパフォーマンスを向上させるために、次のことを推奨します。

- vCPU のピン止めの有効化
- エミュレータのピン止めの有効化
- NUMA チューニングの有効化。すべての vCPU が同じソケットの物理コアにピン止めされていることを確認します。
- hugepage メモリバッキングの設定
- IDE ではなく virtio を使用
- SPICE ではなくグラフィック VNC を使用
- 未使用のデバイスの USB、タブレットなどの取り外し
- memballoon の無効化

（注） これらの設定は、サーバーでインスタンス化できる VM の数に影響を与える可能性があります。

調整手順は、ホストでインスタンス化する少数の VM に対して最も影響があります。

上記に加えて、次の手順を実行します。

CPU ピンニングの有効化

KVM CPU アフィニティオプションを使用して特定のプロセッサに仮想マシンを割り当てることで、KVM 環境のパフォーマンスを向上させます。このオプションを使用する場合は、KVM ホストで CPU ピンニングを構成します。

KVM ホスト環境で、次のコマンドを使用します。

- **virsh nodeinfo**：次のコマンドを使用して、ホストトポロジを確認し、ピン止めに使用できる vCPU の数を確認します。
- **virsh capabilities**：使用可能な vCPU の数を確認します。
- **virsh vcpupin** <*vmname*> <*vcpu#*> <*host core#*>：仮想 CPU をプロセッサコアのセットにピン止めします。

  この KVM コマンドは、Cisco Catalyst 8000V インスタンス上の vCPU ごとに実行する必要があります。次に、仮想 CPU 1 をホストコア 3 にピン止めする例を示します。

  virsh vcpupin c8000v 1 3

  次の例は、vCPU が 4 個の Cisco Catalyst 8000V 構成を使用し、ホストに 8 個のコアが搭載されている場合に必要になる KVM コマンドを示しています。

  virsh vcpupin c8000v 0 2

  virsh vcpupin c8000v 1 3

virsh vcpupin c8000v 2 4

virsh vcpupin c8000v 3 5

ホストのコア番号は、0 ～ 7 のどの番号でもかまいません。詳細については、KVM のドキュメンテーションを参照してください。

（注） CPU ピン止めを構成する場合は、ホストサーバーの CPU トポロジを検討してください。複数のコアがある Cisco Catalyst 8000V インスタンスを使用している場合は、複数のソケットにまたがる CPU ピン止めを設定しないでください。

BIOS 設定

次の表に示す推奨 BIOS 設定を適用して、KVM 設定のパフォーマンスを最適化します。

設定	推奨される設定
Intel Hyper-Threading Technology	無効
Number of Enable Cores	すべて
Execute Disable	有効
Intel VT	有効
Intel VT-D	有効
Intel VT-D coherency サポート	有効
Intel VT-D ATS サポート	有効
CPU パフォーマンス	高スループット
Hardware Prefetcher	無効
Adjacent Cache Line Prefetcher	無効
DCU Streamer Prefetch	無効
Power Technology	カスタム
[Enhanced Intel Speedstep Technology]	無効
[Intel Turbo Boost Technology]	有効
[Processor Power State C6]	無効
Processor Power State C1 Enhanced	無効
Frequency Poor Override	有効

設定	推奨される設定
P-State Coordination	HW_ALL
Energy Performance	パフォーマンス

Red Hat Enterprise Linux の要件については、後続のセクションを参照してください。

ホスト OS 設定

ホスト側では、hugepage を使用し、エミュレータのピン止めを有効にすることをお勧めします。次に、ホスト側の推奨設定の一部を示します。

- IOMMU=pt の有効化
- intel_iommu=on の有効化
- hugepage の有効化
- ネットワーキング パフォーマンスを向上させるためにシステムがサポートしている場合は、SR-IOV を使用。システムに存在する可能性のある SR-IOV の制限を確認してください。

hugepage とエミュレータのピン止めを有効にすることに加えて、次の設定も推奨されます。
nmi_watchdog=0 elevator=cfq transparent_hugepage=never

（注） VPP または OVS-DPDK で Virtio VHOST USER を使用する場合、QEMU のバージョンでサポートされていれば、バッファサイズを 1024 に増やすことができます（rx_queue_size='1024'）。

IO 設定

SR-IOV を使用してパフォーマンスを向上させることができます。ただし、これにより、仮想機能（VF）の数、QoS サポートなどの SR-IOV の OpenStack の制限、ライブ移行、セキュリティグループのサポートなど、いくつかの制限が発生する可能性があることに注意してください。

fd.io VPP や OVS-DPDK などの最新の vSwitch を使用する場合は、VPP ワーカースレッドまたは OVS-DPDK PMD スレッド用に少なくとも 2 つのコアを予約します。

コマンドラインから VPP を実行するには、次のパラメータを設定します。

- -cpu host：このパラメータにより、VM はホスト OS フラグを継承します。これを xml 設定に含めるには、libvirt 0.9.11 以降が必要です。
- -m 8192：最適なゼロパケットドロップ率を実現するには、8GB RAM が必要です。
- rombar=0：PXE 起動遅延を無効にするには、各デバイスオプションリストの最後に rombar=0 を設定するか、"<rom bar=off /> " をデバイスの xml 設定に追加します。

KVM パフォーマンス向上のためのサンプル XML

NUMA チューニングのサンプル XML

```
  <numatune>
    <memory mode='strict' nodeset='0'/>
  </numatune>
```

vCPU およびエミュレータのピン止めのサンプル XML

```
 <cputune>
    <vcpupin vcpu='0' cpuset='3'/>
    <emulatorpin cpuset='3'/>
  </cputune>
```

Hugepage のサンプル XML

```
 <currentMemory unit='KiB'>4194304</currentMemory>
  <memoryBacking>
    <hugepages>
      <page size='1048576' unit='KiB' nodeset='0'/>
    </hugepages>
    <nosharepages/>
  </memoryBacking>
```

IDE の代わりの virtio のサンプル XML

```
<devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='/var/lib/libvirt/images/rhel7.0.qcow2'/>
      <backingStore/>
      <target dev='vda' bus='virtio'/>
      <boot order='1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
    </disk>
```

VNC グラフィックのサンプル XML

```
<graphics type='vnc' port='5900' autoport='yes' listen='127.0.0.1' keymap='en-us'>
      <listen type='address' address='127.0.0.1'/>
    </graphics>
```

memballon を無効にするための XML

```
<memballon model='none'>
```

halt_poll_ns パラメータの設定

halt_poll_ns は、アイドル状態の KVM ゲスト仮想 CPU（vcpus）の処理方法の動作を変更できる KVM パラメータです。

KVM ゲストの仮想CPUに実行するスレッドがない場合、QEMUは従来、アイドル状態のCPUを停止します。この設定は、デフォルトで 400 ナノ秒の期間を指定します。この場合、仮想 CPU は CPU アイドル状態になる前に待機してポーリングします。

仮想 CPU が停止する前のポーリング期間中に新しい作業が到着すると、仮想 CPU はすぐに作業を実行できるようになります。新しい作業が到着したときに仮想CPUがアイドル状態であった場合は、新しい作業を開始する前に仮想 CPU をアイドル状態から戻す必要があります。アイドル状態から実行状態になるまでにかかる時間は、遅延の影響を受けやすいワークロードに悪影響を与える追加の遅延を引き起こします。

デフォルトのカーネルパラメータでは、ゲスト Cisco Catalyst 8000V ルータの CPU がホスト CPU の 100% を消費します。

halt_poll_ns は、次の 2 つの方法で設定できます。

- **Large Holt_poll_ns**：この場合、仮想 CPU をスリープ解除するイベントのビジースピンに多くの CPU が費やされ、acpi ディープスリープの発生が少なくなります。これは、より多くの電力が消費されることを意味します。ただし、ディープスリープ状態からのウェイクアップは少なく、設定されている状態によっては、キャッシュミスなどの問題が発生する可能性があります。
- **Small halt_poll_ns**：この場合、CPUをスリープ解除するイベントのビジースピンに費やされるCPU時間が少なくなり、より多くのacpiディープスリープが発生します。この場合、消費電力は少なくなりますが、ディープスリープ状態からのウェイクアップが多くなります。ウェイクアップが多いと、ディープ スリープ インスタンスが大量に発生する可能性があり、設定によっては大量のキャッシュミスが発生し、ウェイクアップ時間が長くなる可能性があります。

halt_poll_ns パラメータの設定

halt_poll_ns パラメータは次のように設定できます。

1. 実行時に、`echo 0 > /sys/module/kvm/parameters/halt_poll_ns` を実行します。
2. モジュールをロードする場合は、次の設定を実行します。

```
# rmmod kvm_intel
# rmmod kvm
# modprobe kvm halt_poll_ns=0
# mpdprobe kvm_intel
```

3. デバイスを起動するときに、grub2 のパラメータセクションに `kvm.halt_poll_ns=<specify value>` を追加します。

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。