Installing in KVM Environments

Red Hat Enterprise Linux (RHEL) is an enterprise virtualization product produced by Red Hat. RHEL is based on Kernel-based Virtual Machine (KVM) - an open source, full virtualization solution for Linux on x86 hardware that contains virtualization extensions.

From Cisco IOS XE 17.12.1 release, Cisco Catalyst 8000V is also supported on Intel Atom® C3000 processor (Denverton) CPU-based servers with Intel x550 NIC on RHEL 8.4 KVM hypervisor. You can run Cisco Catalyst 8000V on other x86 CPUs with different versions of hypervisor operating systems, but support is only available on these listed versions.

You can install the Cisco Catalyst 8000V virtual router as a virtual machine on Red Hat KVM virtualization. The installation procedure first involves the manual creation of a VM. This is followed by the installation using a .iso file or a qcow2 file. You can install Cisco Catalyst 8000V in a KVM environment by using the:

- **GUI Tool**: Download and install the virt-manager RPM package on the KVM server. Virt-manager is a desktop user interface for managing virtual machines. Installation by using the GUI is the recommended installation method.
- **Command Line Interface**: In this method of installation, use the command line interface to install the Cisco Catalyst 8000V VM.

Note Deploying the OVA template in a KVM environment is not supported.

Cisco Catalyst 8000V supports the Virtio vNIC type on the KVM implementation. KVM supports a maximum of 26 vNICs.

Installation Requirements for KVM

The KVM requirements for Cisco Catalyst 8000V using Cisco IOS XE 17.4.x releases and later are as follows:

Table 1: KVM Versions (Linux KVM based on Red Hat Enterprise Linux)

Cisco IOS XE Release	KVM Version
Cisco IOS XE 17.15.1 release	Linux KVM based on Red Hat Enterprise Linux 9.2 and 8.4 are recommended.
Cisco IOS XE 17.14.x releases Cisco IOS XE 17.13.x release Cisco IOS XE 17.12.x release Cisco IOS XE 17.11.x releases Cisco IOS XE 17.10.x releases Cisco IOS XE 17.9.x releases Cisco IOS XE 17.8.x releases Cisco IOS XE 17.7.x releases	Linux KVM based on Red Hat Enterprise Linux 7.7 and 8.4 are recommended.
Cisco IOS XE 17.4.x releases Cisco IOS XE 17.5.x releases Cisco IOS XE 17.6.x releases	Linux KVM based on Red Hat Enterprise Linux 7.5 and 7.7 are recommended.

Table 2: KVM Versions (SUSE Linux® Enterprise Server)

Cisco IOS XE Release	KVM Version
Cisco IOS XE 17.14.x releases Cisco IOS XE 17.13.x release Cisco IOS XE 17.12.x release Cisco IOS XE 17.11.x releases Cisco IOS XE 17.10.x releases Cisco IOS XE 17.9.x releases Cisco IOS XE 17.6.3 release	Supports SUSE Linux Enterprise Server version 15 SP3
Cisco IOS XE 17.15.1 release	Supports SUSE Linux Enterprise Server version 15 SP5

Table 3: Supported VNICs

VNIC	Supported Releases
Virtio	Cisco IOS XE Release 17.4.1 and later
ixgbevf	Cisco IOS XE Release 17.4.1 and later
i40evf	Cisco IOS XE Release 17.4.1 to Cisco IOS XE 17.8.x releases

VNIC	Supported Releases
iavf	Cisco IOS XE Release 17.9.1 and later
ConnectX-5VF	Cisco IOS XE Release 17.9.1 and later
Ixgbe	Cisco IOS XE Release 17.10.1 and later

Note If a vNIC with an i40evf driver is used, the maximum number of physical VLANs is limited to 512, shared across all (Virtual Functions) VFs, and the number of VLANs for a VF can be further limited by the host (PF) driver for untrusted VFs. The latest Intel i40e PF driver limits untrusted VFs to a maximum of 8 VLANs/sub-interfaces.

Maximum number of vNICs supported per VM instance - 26

- vCPUs. The following vCPU configurations are supported:
  - 1 vCPU: requires minimum 4 GB RAM allocation
  - 2 vCPUs: requires minimum 4 GB RAM allocation
  - 4 vCPUs: requires minimum 4 GB RAM allocation
  - 8 vCPUs: requires minimum 8 GB RAM allocation
  - 16 vCPUs: requires minimum 8 GB RAM allocation (supported from Cisco IOS XE 17.11.1a)
- Virtual CPU cores - 1 vCPU is required
- Virtual hard disk size - 8 GB minimum
- Virtual CD/DVD drive installed (applicable only when installing using an .iso file or when providing Day0 configuration via an ISO) - required

Creating a KVM Instance

Creating the VM Using the GUI Tool

Before you begin

Download and install the virt-manager RPM package on the KVM server.

Download either the .qcow2 image or the .iso image from the Cisco Software Download page, and copy the file onto a local device or a network device.

Step 1 Launch the virt-manager GUI.

Step 2 Click **Create a New Virtual Machine**.

Step 3 Do one of the following:

a) If you have downloaded the .qcow2 file, select **Import Existing Disk Image**.
b) If you have downloaded the .iso file, select **Local Install Media (ISO Image or CDROM)**.

Step 4 Select the Cisco Catalyst 8000V qcow2 or iso file location.

Step 5 Configure the memory and the CPU parameters.

Step 6 Configure the virtual machine storage.

Step 7 (Optional) To add additional hardware before creating the VM, select **Customize configuration before install**. The system displays the **Add Hardware** button. Click this button to add various hardware options, such as additional disks or a serial port interface.

Step 8 (Optional) To add a serial console, follow the procedure as mentioned in Adding a Serial Console, on page 4.

Step 9 (Optional) If you want to customize your configuration before you create the VM, see Customizing Configuration Before Creating the VM, on page 4.

Step 10 Click **Finish**.

Step 11 Access the Cisco Catalyst 8000V console by performing one of the following actions:

a) If you are using a virtual console, double-click the VM instance to access the VM console.
b) If you are using a serial console, see Booting the Cisco Catalyst 8000V and Accessing the Console.

Adding a Serial Console

Perfrom this task to enables acces to the Cisco Catalyst 8000V instance by adding a serial console.

Step 1 Click **Add Hardware**.

Step 2 Select the **Serial** option from the menu.

Step 3 From the **Device Type** drop-down menu, select **TCP net console (tcp)**.

Step 4 Specify the port number, and select the **Use Telnet** checkbox.

Step 5 Click **Finish**.

Step 6 After adding all necessary hardware, click **Begin Installation**.

Customizing Configuration Before Creating the VM

Before you begin

Perform the Creating the VM Using the GUI Tool, on page 3 task by using a .qcow2 or an .iso image. Before you click **Finish**, select the **Customize configuration before install** option. The **Add Hardware** button appears.

Proceed to this procedure which describes the optional steps after selecting the **Customize Configuration Before Install** option.

Step 1 Click **Add Hardware**.

Step 2 Select the **Storage** option.

Step 3 Select the **Select Managed Or Other Existing Storage** checkbox.

Step 4 Click **Browse** and navigate to the **c8000v_config.iso** location. This step is applicable only when you add a Day0 or bootstrap configuration.

Step 5 From the **Device-type** drop-down menu, select **IDE CDROM**.

Step 6 Click **Finish**.

Step 7 After adding all the necessary hardware, click **Begin Installation**.

To perform the bootstrap configuration, see Day 0 Configuration.

Creating the VM Using CLI

- Download and install the virt-install RPM package on the KVM server.
- Download the .**qcow2** image from the Cisco Catalyst 8000V software installation image package and copy it onto a local or network device.

Step 1 To create the VM for a .qcow2 image, use the virt-install command to create the instance and boot. Use the following syntax:

Example:

```
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
```

Step 2 To create the VM, for a .iso image, perform the following steps:

a) Create an 8G disk image in the **.qcow2** format using the **qemu-img** command.

Example:

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

b) Use the **virt-install** command to install the Cisco Catalyst 8000V instance. This requires the correct permissions to create a new VM. The following example creates a 1 vCPU Cisco Catalyst 8000V with 4G of RAM, one network interface, and one serial port.

Example:

```
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
```

The **virt-install** command creates a new VM instance and Cisco Catalyst 8000V installs the image onto the specified disk file.

After the installation is complete, the Cisco Catalyst 8000V VM is shutdown. You can start the VM by executing the **virsh start** command.

Note If you want to provide the day0 configuration through the c8000v_config.iso disk image, add an additional parameter to the **virt-install** command. For example, `--disk path=/my/path/c8000v_config.iso,device=cdrom,bus=ide`. For more information, see Day 0 Configuration.

Red Hat Enterprise Linux - Setting Host Mode

Due to an issue specific to Red Hat Enterprise Linux, when you launch Cisco Catalyst 8000V in a Red Hat Enterprise Linux environment using **virt-install**, set the host mode as follows:

- In Red Hat Enterprise Linux 6, use:

```
--cpu host
```

- In Red Hat Enterprise Linux 7, use:

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

Cloning the VM

Issue

In a KVM environment, when you clone a Cisco Catalyst 8000V virtual machine using the **virt-manager** virtual machine manager, it results in a Cisco Catalyst 8000V virtual machine that you might not be able to boot. The issue is caused by an increase in the size of the cloned image size created by **virt-manager** compared to the original Cisco Catalyst 8000V VM image. The extra bytes (in the KB range) cause the boot failure.

Workaround

There are three workarounds:

- Use the **virt-clone** command to clone the Cisco Catalyst 8000V VM image.
- For a cloned Cisco Catalyst 8000V VM image created by **virt-manager** during the bootup, select the GOLDEN image to boot instead of packages.conf.

- In the Create a new virtual machine window, deselect **Allocate Entire Disk Now** before the new Cisco Catalyst 8000V VM is created. This ensures that the cloned Cisco Catalyst 8000V VM image is able to boot up. However, this workaround does not support nested cloning. Use this method only on the first cloned Cisco Catalyst 8000V VM image.

Increasing the KVM Configuration Performance

You can increase the performance for a Cisco Catalyst 8000V running in a KVM environment by modifying some settings on the KVM host. These settings are independent of the IOS XE configuration settings on the Cisco Catalyst 8000V instance.

To improve the KVM configuration performance, Cisco recommends that you:

- Enable vCPU pinning
- Enable emulator pinning
- Enable numa tuning. Ensure that all the vCPUs are pinned to the physical cores on the same socket.
- Set hugepage memory backing
- Use virtio instead of IDE
- Use graphics VNC instead of SPICE
- Remove unused devices USB, tablet etc.
- Disable memballoon

Note These settings might impact the number of VMs that you can be instantiate on a server.

Tuning steps are most impactful for a small number of VMs that you instantiate on a host.

In addition to the above mentioned, do the following:

Enable CPU Pinning

Increase the performance for the KVM environments by using the KVM CPU Affinity option to assign a virtual machine to a specific processor. To use this option, configure CPU pinning on the KVM host.

In the KVM host environment, use the following commands:

- **virsh nodeinfo**: To verify the host topology to find out how many vCPUs are available for pinning by using the following command.
- **virsh capabilities**: To verify the available vCPU numbers.
- **virsh vcpupin** *<vmname> <vcpu#> <host core#>*: To pin the virtual CPUs to sets of processor cores.

  This KVM command must be executed for each vCPU on your Cisco Catalyst 8000V instance. The following example pins virtual CPU 1 to host core 3:

  virsh vcpupin c8000v 1 3

The following example shows the KVM commands needed if you have aCisco Catalyst 8000V configuration with four vCPUs and the host has eight cores:

virsh vcpupin c8000v 0 2

virsh vcpupin c8000v 1 3

virsh vcpupin c8000v 2 4

virsh vcpupin c8000v 3 5

The host core number can be any number from 0 to 7. For more information, see the KVM documentation.

Note When you configure CPU pinning, consider the CPU topology of the host server. If you are using a Cisco Catalyst 8000V instance with multiple cores, do not configure CPU pinning across multiple sockets.

BIOS Settings

Optimize the performance of the KVM configuration by applying the recommended BIOS settings as mentioned in the following table:

Configuration	Recommended Setting
Intel Hyper-Threading Technology	Disabled
Number of Enable Cores	ALL
Execute Disable	Enabled
Intel VT	Enabled
Intel VT-D	Enabled
Intel VT-D coherency support	Enabled
Intel VT-D ATS support	Enabled
CPU Performance	High throughput
Hardware Prefetcher	Disabled
Adjacent Cache Line Prefetcher	Disabled
DCU Streamer Prefetch	Disable
Power Technology	Custom
Enhanced Intel Speedstep Technology	Disabled
Intel Turbo Boost Technology	Enabled
Processor Power State C6	Disabled
Processor Power State C1 Enhanced	Disabled

Configuration	Recommended Setting
Frequency Poor Override	Enabled
P-State Coordination	HW_ALL
Energy Performance	Performance

For information about Red Hat Enterprise Linux requirements, see the subsequent sections.

Host OS Settings

In the host side, Cisco recommends that you use hugepages and enable emulator pinning. The following are some of the recommended settings in the host side:

- Enable IOMMU=pt
- Enable intel_iommu=on
- Enable hugepages
- Use SR-IOV if your system supports it for higher networking performance. Please check SR-IOV limitations your system might have.

In addition to enabling hugepages and emulator pinning, the following settings are also recommended: nmi_watchdog=0 elevator=cfq transparent_hugepage=never

Note If you use Virtio VHOST USER with VPP or OVS-DPDK, you can increase the buffer size to 1024 (rx_queue_size='1024') provided the version of your QEMU supports it.

IO Settings

You can use SR-IOV for better performance. However, note that this might bring in some limitations such as number of virtual functions (VF), OpenStack limitations for SR-IOV like QoS support, live migration and security group support.

If you use a modern vSwitch like fd.io VPP or OVS-DPDK, reserve at least 2 cores for the VPP worker threads or the OVS-DPDK PMD threads.

Configure the following parameters to run the VPP through command line:

- -cpu host: This parameter causes the VM to inherit the host OS flags. You require libvirt 0.9.11 or greater for this to be included in the xml configuration.
- -m 8192: You require 8GB RAM for optimal zero packet drop rates.
- rombar=0: To disable PXE boot delays, set rombar=0 to the end of each device option list or add "<rom bar=off />" to the device xml configuration.

Sample XMLs for KVM Performance Improvement

Sample XML for numa tuning

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

Sample XML for vCPU and emulator pinning

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

Sample XML for hugepages

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

Sample XML for virtio instead of IDE

```
<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>
```

Sample XML for VNC graphics

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

XML for disabling memballon

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

Configure the halt_poll_ns Parameter

`halt_poll_ns` is a KVM parameter that allows you to alter the behaviour of how idle KVM guest virtual CPUs (vcpus) are handled.

When a virtual CPU in a KVM guest has no threads to run, the QEMU traditionally halts the idle CPU. This setting specifies a period of 400 nanoseconds by default, where a virtual CPU waits and polls before entering a CPU Idle state.

When new work arrives during the polling period before the vcpu is halted, the vcpu is immediately ready to execute the work. If the vcpu has been idle when new work arrives, the vcpu must be brought out of the idle state before the new work can be started. The time taken from idle to running state induces additional latency which negatively impacts latency sensitive workloads.

With the default kernel parameters, the guest Cisco Catalyst 8000V router CPU consumes 100% of the host CPU.

You can configure halt_poll_ns in two ways:

- **Large halt_poll_ns**: In this case, more CPU is spent busy-spinning for events that wake the virtual CPU, and less acpi deep sleeps occur. This means more power is consumed. However, there are less wakeups from deep states states, which depending on the state that's configured, can cause issues like cache misses etc.
- **Small halt_poll_ns**: In this case, less CPU time is spent busy-spinning for events that wake the CPU, more acpi deep sleeps occur. Here, less power consumed, but more wakeups from deep sleep states are required. More wakeups can cause large amounts of deep sleep instances, which depending on the configuration, can cause large amounts of cache misses and long wakeup time.

Configuring the halt_poll_ns parameter

You can configure the halt_poll_ns parameter in the following ways:

1. At run time, run the following: `echo 0 > /sys/module/kvm/parameters/halt_poll_ns`.
2. When you load the module, perform the following configuration:

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

3. When you boot the device, add `kvm.halt_poll_ns=<specify value>` in the parameters section of grub2.