Installing the Cisco IOS XRv 9000 Router in KVM Environments

These file types are needed to install Cisco IOS XRv 9000 Router on the KVM hypervisor:

- .qcow2—Used for booting the software image in KVM OpenStack environments. The qcow2 disk image has an instance of the Cisco IOS XRv 9000 Router pre-installed.
- .iso and .qcow2—Used to manually create the Cisco IOS XRv 9000 Router VM using Virsh application. You must also have virsh.xml file that has sample XML configuration used to launch the Cisco IOS XRv 9000 Router in KVM environments using Virsh commands.

Installation Requirements for KVM

Cisco IOS XRv 9000 Router is supported on top of Ubuntu and Red Hat Distrutions using the Kernel Virtual Machine (KVM). The Cisco IOS XRv 9000 Router installation on KVM requires the creation of a VM and installation using the either .iso file or .qcow2 image. The VM can be launched either using the KVM command line, Virsh, or OpenStack.

For information on the installation requirements for KVM, refer to the latest *Release Notes for Cisco IOS XRv 9000 Router.*

See the Table 1 for release notes links.

Note

- From Release 6.0, minimum of 45GB VM hard disk size is supported.
- The **du** command is used to check the real amount of disk space consumed by the virtual disk image.
- The option **intel_iommu=on** command must be set in the grub configuration for PCIe-passthrough interfaces to work in a VM.
- The CPU flags must be passed into the VM and must include the **sse4_2** flag in order for the embedded Dataplane to work.

For information on KVM support on OpenStack, see the section KVM support on OpenStack.

Installing the Cisco IOS XRv 9000 Router Using KVM Command Line

This procedure provides a general guideline for manually creating the VM for the Cisco IOS XRv 9000 router; the exact steps that you need to perform may vary depending on the characteristics of your KVM environment and setup. For more information, see the Red Hat Linux or Ubuntu documentation.

This procedure explains how to create ISO boot configuration with 3 traffic interfaces and 3 required interfaces (one is for XR management, and two are reserved).

a.
```
/usr/bin/kvm \
```
Invokes the KVM

b.
```
-smbios type=1,manufacturer="cisco",product="Cisco IOS XRv
9000",uuid=97fc351b-431d-4cf2-9c01-43c283faf2a3 \
```
Sets Universally Unique Identifier (UUID) for the VM instance. The UUID is used as part of licensing to identify VM instance.

c.
```
-cpu host \
```
Pass host CPU flags into guest.

d.
```
-drive file=/home/username/bnbMay13/workdir-username/disk1.raw,if=virtio,media=disk,index=1 \
-drive file=/home/username/bnbMay13/workdir-username/xrv9k-fullk9-x.iso.baked,media=cdrom,index=2
 \
```
Empties harddisk and boots ISO.

e.
```
-m 16384 \
```
Creates memory.

f.
```
-smp cores=4,threads=1,sockets=1 \
```
Creates four virtual CPUs and one socket.

g.
```
-enable-kvm \
```

Enables hardware acceleration.

h. `-display none \`

Emulates VGA console when using serial ports for console access. This step is recommended.

i. `-rtc base=utc \`

Sets the real time clock (RTC).

j.

```
      -netdev tap,id=host1,ifname=usernameLx1,script=no,downscript=no \
      -netdev tap,id=host2,ifname=usernameLx2,script=no,downscript=no \
      -netdev tap,id=host3,ifname=usernameLx3,script=no,downscript=no \
      -device virtio-net-pci,romfile=,netdev=host1,id=host1,mac=52:46:84:57:A0:DA \
      -device virtio-net-pci,romfile=,netdev=host2,id=host2,mac=52:46:C4:F4:36:0F \
      -device virtio-net-pci,romfile=,netdev=host3,id=host3,mac=52:46:A5:C0:D0:C5 \
```

Creates 3 NICs; the first is mapped to XR management interface, the second and third are reserved.

k.

```
      -netdev tap,id=data1,ifname=usernameXr1,script=no,downscript=no \
      -netdev tap,id=data2,ifname=usernameXr2,script=no,downscript=no \
      -netdev tap,id=data3,ifname=usernameXr3,script=no,downscript=no \
      -device e1000,romfile=,netdev=data1,id=data1,mac=52:46:87:18:62:DF \
      -device e1000,romfile=,netdev=data2,id=data2,mac=52:46:32:02:90:6F \
      -device e1000,romfile=,netdev=data3,id=data3,mac=52:46:34:93:52:1F \
```

The fourth to eleventh NICs are mapped to traffic ports. A minimum of one traffic interface is recommended.

l.

```
      -serial telnet:0.0.0.0:10621,nowait,server \
      -serial telnet:0.0.0.0:14713,nowait,server \
      -serial telnet:0.0.0.0:18090,nowait,server \
      -serial telnet:0.0.0.0:17181,nowait,server \
```

Creates four serial ports to access console. The first serial port is mapped to XR console. See Console Mapping section for additional information. A minimum of one serial port is recommended.

For information on configuring the serial console access, see Configuring the Serial Console Access in KVM using QEMU section.

m. `-boot once=d &`

Boot the ISO only once.

Example:

```
/usr/bin/kvm \
      -smbios type=1,manufacturer="cisco",product="Cisco IOS XRv
9000",uuid=97fc351b-431d-4cf2-9c01-43c283faf2a3 \
      -cpu host \
      -drive file=/home/username/bnbMay13/workdir-username/disk1.raw,if=virtio,media=disk,index=1 \
     -drive file=/home/username/bnbMay13/workdir-username/xrv9k-fullk9-x.iso.baked,media=cdrom,index=2
 \
      -m 16384 \
      -smp cores=4,threads=1,sockets=1 \
      -enable-kvm \
      -daemonize \
      -display none \
      -rtc base=utc \
      -name IOS-XRv-9000:username \
      -runas username \
      -netdev tap,id=host1,ifname=usernameLx1,script=no,downscript=no \
-netdev tap,id=host2,ifname=usernameLx2,script=no,downscript=no \
-netdev tap,id=host3,ifname=usernameLx3,script=no,downscript=no \
-device virtio-net-pci,romfile=,netdev=host1,id=host1,mac=52:46:84:57:A0:DA \
-device virtio-net-pci,romfile=,netdev=host2,id=host2,mac=52:46:C4:F4:36:0F \
-device virtio-net-pci,romfile=,netdev=host3,id=host3,mac=52:46:A5:C0:D0:C5 \
-netdev tap,id=data1,ifname=usernameXr1,script=no,downscript=no \
-netdev tap,id=data2,ifname=usernameXr2,script=no,downscript=no \
-netdev tap,id=data3,ifname=usernameXr3,script=no,downscript=no \
-device e1000,romfile=,netdev=data1,id=data1,mac=52:46:87:18:62:DF \
-device e1000,romfile=,netdev=data2,id=data2,mac=52:46:32:02:90:6F \
-device e1000,romfile=,netdev=data3,id=data3,mac=52:46:34:93:52:1F \
-monitor telnet:0.0.0.0:11063,server,nowait \
-serial telnet:0.0.0.0:10621,nowait,server \
-serial telnet:0.0.0.0:14713,nowait,server \
-serial telnet:0.0.0.0:18090,nowait,server \
-serial telnet:0.0.0.0:17181,nowait,server \
-boot once=d &
```

Note The disk labeled disk1.raw in the example above can be created with **qemu-img**. The **qemu-img** is an utility to convert the virtual hard disk format. Instead of a raw disk format, qcow2 disk format can be used in above example.

A preinstalled qcow2 image can also be used; in that case the cdrom parameter is removed.

Installing the Cisco IOS XRv 9000 Router in KVM Using Virsh

This procedure provides a general guideline for manually creating the VM for the Cisco IOS XRv 9000; the exact steps that you need to perform may vary depending on the characteristics of your KVM environment and setup. For more information, refer the Red Hat Linux, Ubuntu and Virsh documentation.

Before you begin

The VM memory ballooning is not supported and hence the 'memory' and 'currentmemory' unit values (as shown below) must be the same in Virsh file.

```
<memory unit='MB'>XXX</memory>
<currentMemory unit='MB'>XXX</currentMemory>
```

Step 1 From Cisco.com download the .iso or .qcow2 image along with the sample Virsh XML file.

Step 2 Edit the XML file to point to the .iso or .qcow2 image, and edit the interface sources to designate the connectivity desired.

```
<!-- CDROM HDB Disk -->
     <disk type='file' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <source file='/home/<username>/xrv9k-fullk9-x.iso'/>
      <target dev='hdc' bus='ide'/>
      <alias name='ide-cdrom'/>
     </disk>
```

Step 3 If qcow2 image is used, comment out the CDROM section.

Step 4 If iso image is used, then:

a. Create an empty qcow2 disk:

```
qemu-img create -f qcow2 xrv9000.qcow2 45G
```

Note The minimum supported VM hard disk size is 45GB for release 6.0 and 55GB for release 5.4.

b. Edit the HDA Disk section in the XML to point to the empty qcow2 disk just created:

```
<!-- HDA Disk -->
    <disk type='file' device='disk'>
     <driver name='qemu' type='qcow2'/>
     <source file='/home/<username>/xrv9000.qcow2'/>
     <target dev='vda' bus='virtio'/>
     <alias name='virtio-disk0'/>
    </disk>
```

Step 5 Create the Virsh domain.

```
virsh create xrv9k-fullk9-x.virsh.xml
```

Step 6 Validate the Virsh domain created in Step 5:

```
virsh list xrv9k-fullk9-x.virsh.xml
Id    Name                           State
----------------------------------------------------
 149   IOS-XRv-9000_username_virsh        running
```

Step 7 Use the Virsh commands to manage the VM, such as:

- **reboot**

  ```
  virsh reboot IOS-XRv-9000_USER1_virsh
  ```

- **shutdown**

  ```
  virsh shutdown IOS-XRv-9000_USER1_virsh
  ```

- **destroy**

  ```
  virsh destroy IOS-XRv-9000_USER1_virsh
  ```

For information on configuring the serial console access, see Configuring the Serial Console Access in KVM using Virsh section.

See Virsh command reference documentation for more details.

Creating the Cisco IOS XRv 9000 Router KVM Instance on OpenStack

This procedure shows how to bring the Cisco IOS XRv 9000 Router up on the RHEL 7 and later version and OpenStack 5 and later version . The procedure uses the OpenStack command line interface. It is expected that the reader is familiar with OpenStack commands and procedures. See the OpenStack documentation for further information.

At the end of the procedure, you will have a single Cisco IOS XRv 9000 Router running with 3 data plane interfaces connected to 3 neutron networks.

Note In case trunk interfaces are required, please use Neutron ML2 core plugin with Nexus 1kv mechanism plugin. The following procedure creates only Access (non-tagging) type of interfaces.

Before you begin

You must have:

- OpenStack 5, 6 or 7 with Neutron ML2 core plugin, Open vSwitch as a mechanism plugin and VLAN as Tenants network type.
- Cisco IOS XRv 9000 Router ISO image (VGA type).

Step 1 **Image preparation**

For Cisco IOS XRv 9000 Router deployment on OpenStack you need:

- Cisco IOS XRv 9000 ISO image (VGA type)
- a blank 45GB cinder volume (virtual hard disk)

Note The minimum supported VM hard disk size is 45GB for release 6.0 and 55GB for release 5.4.

When the ISO image is booted, the Cisco XRv 9000 Router gets installed on the second disk (blank 45GB volume). Later, the router can be booted from the created cinder volume. Optionally, a pre-installed Cisco XRv 9000 qcow2 disk can be downloaded in place of the ISO image.

a. Cisco IOS XRv 9000 ISO image (VGA type) must be imported into OpenStack glance. Use the following command line to import the image into glance:

```
glance image-create --name xrv9k-fullk9_vga-x --disk-format iso --container bare \
--file xrv9k-fullk9_vga-x.iso
```

b. Verify the image in OpenStack glance:

```
 glance image-list
+------------------------------------+-------------------------------------+-----------+--------------+-----------+-------+
| ID                                   | Name                                    | Disk Format |
 Container Format | Size        | Status |
+------------------------------------+-------------------------------------+-----------+--------------+-----------+-------+
| 71b44355-32a8-45e7-abfd-f52593f2dc1a | csr1000v-universalk9.03.15.00.S.155-2.S | qcow2       |
 bare             | 1335427072 | active |
| e779245a-9491-4bee-bc85-a73e9394b981 | xrv9k-fullk9_vga-x                      | iso         |
 bare             | 775370752  | active |
| 3b3ade31-fae6-4354-9902-fb77452a65ab | xrvr-initial-config                     | iso         |
 bare             | 358400     | active |
+------------------------------------+-------------------------------------+-----------+--------------+-----------+-------+
```

Step 2 **Cinder volumes preparation**

Use the following command line to create Cinder volumes:

a. Create Cinder volume of Cisco IOS XRv 9000 Router's disk image and make it bootable:

```
cinder create --display-name xrv9k-disk 45
cinder set-bootable volume-id True
```

The **cinder list** command displays the volume ID of router's ISO image.

b. Check Cinder volumes:

```
cinder list
+--------------------------------------+-----------+---------------+------+-------------+----------------------------------------+
|                  ID                  | Status    |  Display Name | Size | Volume Type |
Bootable |             Attached to              |
+--------------------------------------+-----------+---------------+------+-------------+----------------------------------------+
|
| cbef86dd-9819-4daa-81b2-4b905b287974 | Available |   xrv9k-disk    |  45  |    None     |
true   | 5262e1fe-37f5-4535-bf76-aab26cb86366 |
+--------------------------------------+--------+-------------+------+-----------+------------------------------------------
```

You must see Cinder volume with name xrv9k-disk with **Status-Available** and **Bootable-True.**

Step 3 **Configuring Nova Flavor**

The virtual hardware templates are called **flavors** in OpenStack. The **nova flavor-create** command allows users to create new flavors. The nova’s flavor is used to pass the information about RAM size, disk, and number of cores to the Nova scheduler process when new instances of VM are launched.

The Cisco IOS XRv 9000 Router requires 16GB of RAM, 45GB hard disk, and 4 vCPUs.

```
nova flavor-create xrv9k-flavor auto 16384 45 4
```

xrv9k-flavor is flavor's name.

Step 4 **Configuring network**

Cisco IOS XRv 9000 Router requires a minimum of 4 network interfaces. The first NIC or vNIC is mapped with the XR management interface, the second and third NICs are reserved, and the remaining NICs are mapped to traffic interfaces. For more information on interface mapping, see the Mapping the Router Network Interfaces to VM Network Interface Cards section.

For a sample network configuration, we can have 6 neutron networks, 6 neutron subnetworks and then pass 6 network IDs as parameters to VM instance. This equates to 3 traffic interfaces. Use **neutron net-create <*neutron-network-name*>** to create networks in Neutron.

For example:

```
neutron net-create management-xr
neutron net-create management-other
neutron net-create management-host
neutron net-create datalink-1
neutron net-create datalink-2
neutron net-create datalink-3
```

In the above example, first three commands create control plane networks, and last three commands create data plane networks.

Now using corresponding network names, create subnetworks in Neutron. Use **neutron subnet-create *<neutron-network-name>* *<IP-subnet>* --name *<neutron-network-name>*** command. For consistency, keep the neutron names and subnetwork names same.

For example:

```
neutron subnet-create management-xr  10.50.70.0/26 --name management-xr
neutron subnet-create management-other  10.50.70.64/26 --name management-other
neutron subnet-create management-host  10.50.70.128/26 --name management-host
neutron subnet-create datalink-1  10.57.11.0/24 --name datalink-1
neutron subnet-create datalink-2  10.57.12.0/24 --name datalink-2
neutron subnet-create datalink-3  10.57.13.0/24 --name datalink-3
```

Step 5 **Attach management-xr and management-host subnets to the neutron router**

Use these commands to attach the management-xr and management-host subnets to the neutron router:

```
neutron router-interface-add <Neutron router name> <subnet id of management-xr>
neutron router-interface-add <Neutron router name> <subnet id of management-host>
```

Note
- This step must be performed before launching the Cisco IOS XRv 9000 Router to avoid DHCP issues.
- The **neutron router-list** command displays list of Neutron router name.
- The **neutron subnet-list** command displays list of Neutron subnet.

Step 6 **Launch the Cisco IOS XRv 9000 Router**

Follow this command line to boot the router.

```
nova boot
 --flavor xrv9k-flovor \
 --nic net-id={Control plane network ID of management-xr} \
 --nic net-id={Control plane network ID of management-other} \
 --nic net-id={Control plane network ID of management-host} \
 --nic net-id={Data plane network ID of datalink-1} \
 --nic net-id={Data plane network ID of datalink-2} \
 --nic net-id={Data plane network ID of datalink-3} \
 --block-device id={glance ID of Cisco IOS XRv 9000 router's CD volume available in Step 1},\
 --source=image,dest=volume,bus=ide,device=/dev/hdc,size=1,type=cdrom,bootindex=1 \
 --block-device source=volume,id={cinder ID of Cisco IOS XRv 9000 Router's disk volume available in
 Step 1}, dest=volume,size=50,bootindex=0 xrv9k-1
nova list (your instance should be in Active state)
nova get-vnc-console xrv9k-1 novnc
```

The **nova get-vnc-console** command returns an URL that is used to access the Cisco IOS XRv 9000 Router console.

Note The **config-drive** allows the VM to boot with an initial configuration so that when the VM is powered on, it can perform services, configured in the config-drive. As the Cisco IOS XRv 9000 Router is running on the VM, the router accepts a valid XR configuration through a plain text file. During the boot process the text file is parsed by the command parser, just as if the commands were entered through CLI.

If **config-drive** support is desired, a plaintext file with the initial XR configuration can be passed by adding this line under **nova boot** command:

```
config-drive true user-data <path>/iosxr_config.txt file
/iosxr_config.txt=<path>/iosxr_config.txt
```

For information on Cisco Virtual Appliance Configuration (CVAC), see section CVAC - Bootstrap Configuration Support.

Increasing Performance on KVM Configuration

You can increase the performance for a Cisco IOS XRv 9000 Router in the KVM environment by changing settings on the KVM host. These settings are independent of the Cisco IOS XR configuration settings on the router. This option is available in Red Hat Enterprise Linux 7.0 KVM.

You can improve performance on KVM configurations by enabling CPU pinning.

Note The Cisco IOS XR Software Release 5.4 does not support jumbo packets larger than 1518 bytes for KVM on a VirtIO interface. The packets larger than 1518 bytes are dropped.

Enabling CPU Pinning

To increase the performance of the Cisco IOS XRv 9000 Router in KVM environments, you can use the KVM CPU affinity option to assign a VM to a specific processor. To use this option, configure CPU pinning on the KVM host.

Follow these steps to configure CPU pinning:

1. In the KVM host environment, verify the host topology to find out how many vCPUs are available for pinning:

   ```
   virsh nodeinfo
   ```

2. Verify the available vCPU numbers:

   ```
   virsh capabilities
   ```

3. Pin the vCPUs to sets of processor cores:

   ```
   virsh vcpupin <vm-name> <vcpu-number> <host-core-number>
   ```

   The **virsh vcpupin** command must be executed for each vCPU on your Cisco IOS XRv 9000 Router. The following example shows the KVM commands needed if you have a Cisco IOS XRv 9000 Router configuration with four vCPUs and the host has eight cores:

   ```
   virsh vcpupin xrv9000 0 2
   ```

```
virsh vcpupin xrv9000 1 3
virsh vcpupin xrv9000 2 4
virsh vcpupin xrv9000 3 5
```

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

Note When configuring CPU pinning, carefully consider the CPU topology of the host server. If using a Cisco IOS XRv 9000 Router configured with multiple cores, do not configure CPU pinning across multiple sockets.

The downside of improving performance on KVM configuration is that it requires dedicated system resources.