Installing the Controller in a KVM Environment

Overview of Kernel-Based Virtual Machine Environment

Cisco Catalyst 9800 Wireless Controller for Cloud is supported on top of Ubuntu, Red Hat Enterprise Linux (RHEL) 7.2, and Red Hat Enterprise Virtualization (RHEV) using the Kernel-Based Virtual Machine (KVM). Installation on a KVM requires the creation of a virtual machine (VM) and installation using a .iso file or a .qcow2 file. The VM can be launched using the KVM command line or Virsh.

- .qcow2: Used for booting a software image in KVM environments.
- .iso: Used to manually install the Cisco Catalyst 9800 Wireless Controller for Cloud using the Virsh tool. You must also have a virsh.xml file with a sample XML configuration to launch the controller in KVM environments using virsh commands.

Supported Profile Configurations

The supported profile configurations are:

Table 1: Supported Profile Configurations

Profiles	CPUs	RAM	APs	Clients
Ultra-Low	2 vCPUs	6 GB	100	1,000
Small	4 vCPUs	8 GB	1000	10,000

Profiles	CPUs	RAM	APs	Clients
Medium	6 vCPUs	16 GB	3000	32,0000
Large	10 vCPUs	32 GB	6000	64,0000

Supported Networking Options

The following are the networking options supported:

- Linux bridge
- Open vSwitch (OVS)

Required Packages for a KVM Installation

The required packages for a KVM installation are:

- Qemu-kvm
- Qemu-utils
- Uml-utilities
- Socat
- KVM
- Libvirt-bin
- Virtinst

Installation Procedure in a KVM Environment

You can install Cisco Catalyst 9800 Wireless Controller for Cloud in a KVM environment either by using the self-installing package that guides you through the installation steps or by using one of the management software supported by KVM, such as virt-manager, virt install, or virsh.

The KVM Installer package is a self-installing package for KVM. When you run this package, it provides the following modes:

- Default: Installs the controller using the bundled image file and one of the default VM configuration options (small, medium, or large).
- Interactive: Allows customization of the VM configuration and provides the option to install the bundled image file or a separate .qcow2 image.

Note For a list of unsupported VM operations, refer to *Supported VMware Features and Operations* section in Installing in a VMware ESXi Environment chapter.

Before you begin

Download the .run executable from the Cisco Catalyst 9800 Wireless Controller for Cloud software installation image package and copy it to a local drive of the host machine.

Installing the Controller with Linux Bridge Networking Using the .qcow2 Image

This procedure provides general guidelines for manually creating the VM for the controller; the exact steps that you should perform may vary depending on the characteristics of your KVM environment and setup. For more information, see the Red Hat Linux, Ubuntu, and Virsh documentation.

Using the **virt-install** command, create an instance and boot, using the following syntax:

```
--connect=qemu:///system \
--os-type=linux \
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--console pty,target_type=virtio \
--hvm \
--import \
--name=my_c9k_vm \
--disk path=<path_to_c9800-c_qcow2>,bus=ide,format=qcow2 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--ram=4096 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio  \
--noreboot \
```

Use the following syntax for the Ultra-Low profile:

```
--connect=qemu:///system \
--os-type=linux \
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--console pty,target_type=virtio \
--hvm \
--import \
--name=my_c9k_vm \
--disk path=<path_to_c9800-c_qcow2>,bus=ide,format=qcow2 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--ram=6144 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio  \
--noreboot \
```

Note After the installation is complete, the controller VM is shutdown. Start the controller VM using the **virsh start** command.

Installing the Controller with Vrish Using the ISO Image

This procedure provides a general guideline for manually creating the VM for the controller; 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, Ubuntu and Virsh documentation.

Step 1 Create an 16 GB disk image in **.qcow2** format using the **qemu-img** command:

```
qemu-img create -f qcow2 c9000-c_disk.qcow2 16G
```

Step 2 Use the **virt-install** command to install the controller. This requires the correct permissions to create a new VM. The following example shows how to create a 1-vCPU VM with 4-GB of RAM, and three network interfaces.

```
virt-install \
--connect=qemu:///system \
--os-type=linux \
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--hvm \
--import \
--name=my_c9k_vm \
--cdrom=<path_to_c9800-c_iso> \
--disk path=c9000-c_disk.qcow2,bus=virtio,size=8,sparse=false,cache=none,format=qcow2 \
--ram=4096 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--noreboot \
```

The following example shows how to create a 1-vCPU VM with 4-GB of RAM, and three network interfaces, in the Ultra-Low profile:

```
virt-install \
--connect=qemu:///system \
--os-type=linux \
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--hvm \
--import \
--name=my_c9k_vm \
--cdrom=<path_to_c9800-c_iso> \
--disk path=c9000-c_disk.qcow2,bus=virtio,size=8,sparse=false,cache=none,format=qcow2 \
--ram=6144 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--noreboot \
```

Note The **virt-install** command creates a new VM instance and the controller installs the image on the specified disk file. After the installation is complete, the controller VM is shutdown. Start the controller VM using the **virsh start** command.

Installing the Controller with OVS Networking Using the .qcow2 Image

This procedure provides a general guideline for manually creating the VM for the controller; 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, Ubuntu and Virsh documentation.

Using the **virt-install** command, create an instance and boot, using the following syntax:

```
--connect=qemu:///system \
--os-type=linux \
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--console pty,target_type=virtio \
--hvm \
--import \
--name=my_c9k_vm \
--cdrom=<path_to_c9800-c_iso> \
--disk path=c9000-c_disk.qcow2,bus=virtio,size=8,sparse=false,cache=none,format=qcow2 \
--ram=4096 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--noreboot \
```

Use the following syntax for Ultra-Low profile:

```
--connect=qemu:///system \
--os-type=linux \
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--console pty,target_type=virtio \
--hvm \
--import \
--name=my_c9k_vm \
--cdrom=<path_to_c9800-c_iso> \
--disk path=c9000-c_disk.qcow2,bus=virtio,size=8,sparse=false,cache=none,format=qcow2 \
--ram=6144 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--noreboot \
```

Note After the installation is complete, the controller VM is shutdown. Start the controller VM using the **virsh start** command.

Installing the Controller with Vrish Using Bootstrap Configuration

This procedure provides a general guideline for manually creating the VM for the controller; 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, Ubuntu and Virsh documentation.

Before you begin

Create a text file named *iosxe_config.txt* with the required configuration and create a .iso image using the following command by providing the iosxe_config.txt file as input: **mkisofs -l -o** *iso-file-name.iso iosxe_config.txt*

```
mkisofs -l -o test.iso iosxe_config.txt
```

A sample configuration file is given below:

```
hostname C9800-CL
license smart enable
username lab privilege 15 password lab
ip domain-name cisco.com
interface GigabitEthernet1
 ip address 10.0.0.5 255.255.255.0
 no shut
exit
ip route 0.0.0.0 0.0.0.0 10.0.0.1
line vty 0 4
  login local
exit
```

Use the **virt-install** command to install the controller. Use of this command requires proper privileges to create a new VM. The following example shows how to create a 1-vCPU VM with 4-GB of RAM, and three network interfaces.

```
virt-install \
--connect=qemu:///system \
--os-type=linux \
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--console pty,target_type=virtio \
--hvm \
--import \
--name=my_c9k_vm \
--disk path=<path_to_c9800-c_qcow2>,bus=ide,format=qcow2 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--ram=4096 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--noreboot \
```

The following example shows how to create a 1-vCPU VM with 4-GB of RAM, and three network interfaces, for an Ultra-Low profile:

```
virt-install \
--connect=qemu:///system \
--os-type=linux \
```

```
--os-variant=rhel4 \
--arch=x86_64 \
--cpu host \
--console pty,target_type=virtio \
--hvm \
--import \
--name=my_c9k_vm \
--disk path=<path_to_c9800-c_qcow2>,bus=ide,format=qcow2 \
--vcpus=1,sockets=1,cores=1,threads=1 \
--ram=6144 \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--network=network:<network name>,model=virtio \
--noreboot \
```

Creating Controller Instance Through VMM Using ISO Image

Step 1 Start the virt-manager using **Applications > System Tools > Virtual Machine Manager**.

You may be asked to select the hypervisor and enter your root password.

Step 2 Choose **File** option on top and select **New Virtual Machine** option.

Step 3 Enter the virtual machine details:

a) Enter a **Name** for the VM.
b) In the operating system option, select **Local install media**.
c) Click **Forward**.

Step 4 Select the **ISO image** from the disk.

Step 5 Select **Automatically Detect operating system based on install media**.

Step 6 Configure the memory and CPU options:

a) Set **Memory (RAM)**.
b) Set **CPUs**.
c) Click **Forward** to continue.

Step 7 Set disk image size as 16 GB and click **Forward**.

Step 8 Enter the instance name.

Step 9 Check the **Customize configuration before install** box first before you click **Finish**.

This allows you to add additional NICs.

Step 10 Select the **Network** tab to add additional NICs.

Step 11 Select the**Network** from the **Network source** drop-down.

Note Only virtio network driver is supported.

Step 12 Select the **Portgroup** using the drop-down.

Step 13 Click **Finish**.

Bootstrap Configuration with KVM VMM (virt-manager)

The virt-manager, also known as Virtual Machine Manager, is a desktop application for managing virtual machines through libvirt. It presents a summary view of running domains, their live performance and resource utilization statistics. Wizards enable the creation of new domains, and configuration and adjustment of a domain's resource allocation and virtual hardware. An embedded VNC and SPICE client viewer presents a full graphical console to the guest domain.

Step 1 Start virt-manager **Applications > System Tools > Virtual Machine Manager**.

You may be asked to select the hypervisor and/or enter your root password.

Step 2 Select **File** option on top and click **New Virtual Machine** option.

Step 3 Enter the virtual machine details:

a) Specify a **Name**.

b) For the operating system, select **Import existing disk image**.

This method allows you to import a disk image (containing a pre-installed, bootable operating system, if you select the qcow2 image) to it.

c) Click **Forward** to continue.

Step 4 Select the controller qcow2 image path.

Step 5 Configure the memory and CPU options:

a) Set **Memory (RAM)** to *8192*.

b) Set **CPUs** to *4*.

c) Click **Forward** to continue.

Note These steps help you to configure the Cisco Catalyst 9800-CL Cloud Wireless Controller Small Profile. For more information about resource requirements for all variants, refer to Table 1

Step 6 Enter the instance name.

Step 7 Check the **Customize configuration before install** box first before you click **Finish**.

This allows you to add more NICs.

Step 8 Select the **Network**.

Choose either bridge or network.

Step 9 Click **Finish**.

Step 10 Double click on the **Instance name** to edit it.

Step 11 Select ***i***to get the Instance information

Step 12 Select **Begin Installation** to start the Instance.

Step 13 Click the **Monitor** symbol to go to the Virtual Console.

Configuring SR-IOV for KVM

Recommended Software Versions for SR-IOV

Table 2: List of Supported NIC Types

NIC	Firmware	Driver Version	Host OS
Intel x710	7.10	I40e 2.10.19.82	KVM RedHat Version 7.5 and above
Ciscoized x710	7.0	I40e 2.10.19.82	KVM RedHat Version 7.5 and above

Enabling Intel VT-D

Note You need to have root permissions to perform subsequent tasks.

To enable Intel VT-D, perform the following steps:

Step 1 In the **/etc/sysconfig/grub** file and **GRUB_CMDLINX_LINUX** line, add the **intel_iommu=on** and **iommu=pt** parameters at the end.

Step 2 Regenerate the **/etc/grub2.cfg** file by executing the following command:

grub2-mkconfig -o /etc/grub2.cfg

Note In case of EFI, execute the following command:

grub2-mkconfig -o /etc/grub2-efi.cfg

Step 3 Reboot the system for the changes to take effect.

Your system is now capable of PCI device assignment.

Configuring SR-IOV Mode Virtual Functions (VFs) on the Interface

If VF is not available, configure SR-IOV VF using the following commands:

Step 1 Configure VF on the interface:

echo “no_of_vfs” > /sys/class/net/<interface_name>/device/sriov_numvfs

Sample output:

```
echo 1 > /sys/class/net/enp129s0f0/device/sriov_numvfs
```

Here, one VF is created for each port for maximum performance.

Step 2 Configure spoofcheck, trust mode, and MAC on the VF using the following commands:

```
ip link set dev enp129s0f0 vf 0 trust on
ip link set enp129s0f0 vf 0 spoofchk off
ip link set enp129s0f0 vf 0 mac 3c:fd:fe:de:cc:bc
```

Note The MAC addresses must be unique.

Step 3 Verify the settings using the following command:

ip link show *interface_name*

Sample output:

```
ip link show enp129s0f0
6: enp129s0f0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen
 1000
link/ether 3c:fd:fe:de:01:bc brd ff:ff:ff:ff:ff:ff
vf 0 MAC 3c:fd:fe:de:cc:bc, spoof checking off, link-state auto, trust on
```

Configuring SR-IOV Setting Persistence

SR-IOV configurations configured in the above way are not persistent across reboots. To resolve this issue, you can execute the above configuration as a service that is auto-enabled on host reboots.

Step 1 Create a bash script with the commands to be persisted. You need to write the script in **/usr/bin/sriov-config** file as follows:

```
#!/bin/sh
echo "no_of_vfs" > /sys/class/net/<interface_name>/device/sriov_numvfs
ip link set dev <interface_name> vf 0 trust on
ip link set <interface_name> vf 0 spoofchk off
ip link set <interface_name> vf 0 mac 3c:fd:fe:de:cc:bc
```

Sample output:

```
#!/bin/sh
echo 1 > /sys/class/net/enp129s0f0/device/sriov_numvfs
ip link set dev enp129s0f0 vf 0 trust on
ip link set enp129s0f0 vf 0 spoofchk off
ip link set enp129s0f0 vf 0 mac 3c:fd:fe:de:cc:bc
```

Note You need to repeat the same steps for all VFs.

Step 2 Provide execute permission for the script:

chmod 777 /usr/bin/sriov-config

Step 3 Create the system service: Define a new system service to be executed at the end of the boot. This service executes the bash script which has the required sriov commands as mentioned in **Step 1**.

Note Create a new file named **sriov.service** in **/usr/lib/systemd/system** and add the following content:

```
[Unit]
Description=SR-IOV configuration
After=rc-local.service
Before=getty.target
[Service]
Type=oneshot
ExecStart=/usr/bin/sriov-config
[Install]
WantedBy=multi-user.target
```

Note The **ExecStart=/usr/bin/sriov-config** command line executes the script.

Step 4 Enable and start the **sriov.service** using the following command:

systemctl --now enable sriov.service

Note This command starts the service immediately and ensures that the service is run every time the host reboots.

For more information on the SR-IOV configuration for KVM, see:

https://www.intel.com/content/www/us/en/embedded/products/networking/xl710-sr-iov-config-guide-gbe-linux-brief.html

Attaching the SR-IOV to the Controller

Attaching to a New Virtual Machine Using Command Line

Use the **host device** option of **virt-install** to add the PCI VF devices. Use the information from **Step 1** (Configuring SR-IOV Mode Virtual Functions (VFs) on the Interface, on page 9) and PCI BDF number to attach the devices.

Virtual Functions on Intel Corporation Ethernet Controller X710 for 10GbE SFP+. (enp129s0f0):

```
PCI BDF                                          Interface
=========                                        ==========
0000:18:06.0                          enp129s0f0
0000:18:06.1                                     enp129s0f1
```

Creating and Launching a VM

To create and launch a VM, use the following example for the Ultra-Low profile:

```
sudo virt-install   --virt-type=kvm   --name test_vm  --ram 6144  --vcpus=2   --osinfo
detect=on,require=off  --hvm
--cdrom=/home/admin/C9800-CL-universalk9_vga.BLD_V1712_THROTTLE_LATEST_20240307_033213.iso
    --network bridge:br1  --graphics vnc  --disk
path=/var/lib/vwlc-ul-1712.qcow2,size=16,bus=virtio,format=qcow2
```

You get to view the VM console using the following command:

```
virsh console ewlc_sriov_3-18
```

```
Connected to domain ewlc_sriov_3-18
Escape character is ^]
```

You can enter the following command to verify the SR-IOV drivers for the interface:

Device > enable

Device #show platform software vnic-if interface-mapping

```
Device # show platform software vnic-if interface-mapping
------------------------------------------------------------
Interface Name      Driver Name     Mac Addr
------------------------------------------------------------
GigabitEthernet2    net_i40e_vf     3cfd.fede.ccbd
GigabitEthernet1    net_i40e_vf     3cfd.fede.ccbc
------------------------------------------------------------
```

Note The MAC address mentioned above is the same as the one that is set for the VF.

You can verify the processor, memory, vNIC, hypervisor, and throughput profile details using the following command:

```
Device# show platform software system all
Controller Details:
=================
VM Template: medium
Throughput Profile: high
AP Scale: 3000
Client Scale: 32000
WNCD instances: 3

Processor Details
=================
Number of Processors : 9
Processor : 1 - 9
vendor_id : GenuineIntel
cpu MHz  : 2593.748
cache size : 4096 KB
Crypto Supported : Yes
model name : Intel Core Processor (Haswell, IBRS)
Memory Details
==============
Physical Memory : 16363364KB

VNIC Details
============
Name                     Mac Address     Driver Name       Status  Platform MTU
GigabitEthernet1   3cfd.fede.ccbc   net_i40e_vf        DOWN    1522
GigabitEthernet2   3cfd.fede.ccbd   net_i40e_vf        DOWN    1522

Hypervisor Details
====================
Hypervisor: KVM
Manufacturer: Red Hat
Product Name: KVM
Serial Number: Not Specified
UUID: 0E3546DD-DE6E-400D-9B3D-025215519CB8
image_variant :

Boot Details
====================
```

```
Boot mode: BIOS
Bootloader version: 1.1
```

For the Ultra-Low profile, you can verify the processor, memory, vNIC, hypervisor, and throughput profile details using the following command:

```
Device# show platform software system all

Controller Details:
==================
VM Template: ultra-low
Throughput Profile: low
AP Scale: 100
Client Scale: 1000
WNCD instances: 1

Processor Details
==================
Number of Processors : 2
Processor : 1 - 2
vendor_id : GenuineIntel
cpu MHz  : 3192.307
cache size : 25600 KB
Crypto Supported : Yes
model name : Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz

Memory Details
==============
Physical Memory : 6018440KB

VNIC Details
============
Name                Mac Address     Driver Name       Status  Platform MTU
GigabitEthernet1    000c.290b.a787  net_vmxnet3       UP      1500
GigabitEthernet2    000c.290b.a791  net_vmxnet3       UP      1500
GigabitEthernet3    000c.290b.a79b  net_vmxnet3       UP      1500

Hypervisor Details
===================
Manufacturer: VMware, Inc.
Product Name: VMware Virtual Platform
Serial Number: VMware-56 4d b8 4a d8 db 99 d1-a8 a8 8f c4 be 0b a7 87
UUID: 4ab84d56-dbd8-d199-a8a8-8fc4be0ba787
image_variant :

Boot Details
====================
Boot mode: BIOS
Bootloader version: 3.3
```

Attaching an Interface to the Controller Using KVM VMM (virt-manager)

In the virt-manager, select **Hardware > Add Hardware** to add the PCI host device to the VM. Navigate to the NIC card and choose the VF that needs to be attached to the VM.

Once the PCI is added to the VM, you can start the VM.

Verifying SR-IOV Driver and Firmware Version

You can verify the ethernet and driver versions using the following command:

```
ethtool -i <interface_name>
```

Note You need to execute this command on the host machine.

```
[root@cpp-rhel-perf ~]# ethtool -i enp129s0f0
driver: i40e
version: 2.10.19.82
firmware-version: 7.10 0x8000646c 1.2527.0
expansion-rom-version:
bus-info: 0000:81:00.0
```

You can print the ethernet information, driver versions, and SR-IOV VF names using the following command:

```
lspci | grep -i eth
```

```
[root@cpp-rhel-perf ~]# lspci | grep -i eth
81:00.0 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev
 02)
81:00.1 Ethernet controller: Intel Corporation Ethernet Controller X710 for 10GbE SFP+ (rev
 02)
81:02.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev
02)
81:0a.0 Ethernet controller: Intel Corporation Ethernet Virtual Function 700 Series (rev
02)
```

For information on the firmware for Intel NIC, see:

https://downloadcenter.intel.com/product/82947/Intel-Ethernet-Controller-X710-Series

For information on the driver for Intel and Cisco NIC, see:

https://downloadcenter.intel.com/download/24411/Intel-Network-Adapter-Driver-for-PCIe-40-Gigabit-Ethernet-Network-Connections-Under-Linux-?product=82947

For information on the firmware for Cisco NIC, see:

https://www.cisco.com/c/en/us/support/servers-unified-computing/ucs-c-series-rack-servers/tsd-products-support-series-home.html