Nexus 9000v Deployment

This chapter contains the following sections:

Nexus 9000v Hypervisor Support

Both platforms in the Nexus 9000v platform family are designed to run as virtual machines on the supported hypervisors. Limitations of the underlying hypervisor may restrict some of the platform capabilities. This section provides the level of support and associated limitations.

KVM/QEMU Attributes

The following table provides the supported attributes for the KVM/QEMU hypervisor.

Attribute	Support
QEMU Version	4.2.0
BIOS	OVMF version 16, https://www.kraxel.org/repos/jenkins/edk2/ This URL accesses an index page containing the latest OVMF RPM package files. An example of the file is: `edk2.git-ovmf-x64-0-20200515.1388.g9099dcbd61.noarch.rpm` Download and extract the package file with an RPM utility. The package contains a number of files. Locate `OVMF-pure-efi.fd` and use it as the BIOS file. You can rename it `bios.bin` if you want.
Linux Version	Ubuntu 20.0.4

Attribute	Support
Platform	Nexus 9300v deployment Nexus 9500v deployment
Line Cards	Nexus 9300v: 1 line card Nexus 9500v: up to 16 line cards
Line Card Interfaces	Nexus 9300v: up to 64 line card interfaces Nexus 9500v: up to 400 line cards interfaces

ESXI Attributes

The following table provides the supported attributes for the ESXI hypervisor.

Attribute	Support
Version	8.0.
Platform	Nexus 9300v deployment Nexus 9500v deployment
Line Card	Nexus 9300v: 1 line card Nexus 9500v: up to 16 line cards
Line Card Interface	Nexus 9300v: up to 9 line card interfaces Nexus 9500v: up to 9 line cards interfaces

VirtualBox Attributes

The following table provides the supported attributes for the VirtualBox hypervisor.

Attribute	Support
Version	7.0
Platform	Nexus 9300v deployment
Line Card	Nexus 9300v: 1 line card
Line Card Interface	Nexus 9300v: up to 4 line card interfaces

Nexus 9000v Deployment Workflow for KVM/QEMU

This section describes the steps required to deploy Nexus 9000v platforms on KVM/QEMU hypervisors. Three types of deployment are available:

- Common Deployment
- Platform-Specific Deployment
- Interconnecting Deployment

Common Deployment Workflow

You can deploy the Cisco Nexus 9000v platforms through the KVM/QEMU hypervisor. The following table lists the supported parameters for the Cisco Nexus 9000v deployment on KVM/QEMU.

Parameter	Example	Description
/path_to/qemu	/usr/bin/qemu-system-x86_64	Path to QEMU executable. (download the QEMU software from http://wiki.qemu.org/download for different versions.)
-nographic	-nographic	Recommended, as the Cisco Nexus 9000v platforms don't support VGA.
-bios file	-bios bios.bin	Required. Cisco Nexus 9000v platforms use EFI boot and require a compatible BIOS image to operate. We recommend using the latest OVMF BIOS file with the SATA controller for better performance in terms of disk operation. QEMU 2.6 is recommended with the SATA controller. For more information, see http://www.linux-kvm.org/page/OVMF.
-smp	-smp 4	Cisco Nexus 9000v platforms support one to four vCPUs (we recommend two to four).
-m memory	-m 10240	Memory in MB.
-serial telnet:host:port,server,nowait	-serial telnet:localhost:8888,server,nowait or -serial telnet:server_ip:8888,server,nowait	Requires at least one.

Parameter	Example	Description
-net ... -net ... or -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= or -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=	The net/net or netdev/device pairs are for networking a virtual network interface card (vNIC). The _s_f represents the PCI slot number and function number. QEMU 2.0 or above can plug in at least 20 PCI slots and four functions, which accommodates about 80 vNICs in total. The slot range is 3-19, and the function number range is 0-3. The mac= option passes the MAC address of each vNIC MAC address to the VM interfaces. The first -netdev is automatically mapped to the mgmt0 interface on the VM. The second -netdev is mapped to the e1/1 interface, and so on, up to the 65th on e1/64. Check that the MAC addresses are unique for each network device.
-enable-kvm	-enable-kvm	This flag is required for the Cisco Nexus 9000v.
-drive ... -device ... (for the SATA controller)	-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	Format to use for the SATA controller. We recommend using the SATA controller with QEMU 2.6.0 because this controller offers better performance than the IDE controller. However, if there's an early QEMU version that doesn't support the SATA controller, you can use the IDE controller.
-drive ... media=cdrom	-drive file=cfg.iso,media=cdrom	CD-ROM disk containing a switch configuration file applied after the Cisco Nexus 9000v platform comes up. 1. Name a text file (nxos_config.txt). 2. Use Linux commands to make cfg.iso, mkisofs -o cfg.iso -l --iso-level 2 nxos_config.txt.

Platform Specific Workflow

The Cisco Nexus 9500v platform runs in two different modes: sequential and mac-encoded mode. The Nexus 9300v and Nexus 9500v sequential mode deployment steps are the exact same on KVM/QEMU hypervisor. The maximum interfaces for both platforms in this case are 401 interfaces (1 management or 400 data ports).

The Nexus 9500v emulates interface traffic on multiple line cards. The virtual switch uses a single VM on KVM/QEMU for up to a total number of 400 interfaces. Based on the Nexus 9500v mac-encoded schema, specify each network adapter MAC address with the encoded slot and port number when the KVM/QEMU CLI command is invoked.

Interconnecting Platforms

Interconnecting between Nexus 9000v platform instances or any other virtual platform is based on Linux bridges and taps. Prior to invoke any CLI commands, make sure that the following is available (example configuration provided).

In the configuration example below, you can create bridges and tap interfaces along with two N9Kv swicthes with one management and one data interface each. Management interfaces “interface mgmt0” are connected to management network with the bridge “mgmt_bridge. The data port interfaces “interface Eth1/1” from both switches are connected back to back by using the bridge “interconnect_br”.

Note The minimum QEMU version required is 4.2.0.

- Bridges (similar to vSwitch in ESXi hypervisor) are created and set to the "up" state.

  Linux commands to create bridges and bring them up:

  sudo brctl addbr *mgmt_bridge*

  sudo brctl addbr *interconnect_br*

  sudo ifconfig *mgmt_bridge* **up**

  sudo ifconfig *interconnect_br* **up**

- Tap interfaces are created based on number of interfaces the Nexus 9000v is using.

  Linux command to create tap interfaces:

  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*

- Bridges are connected to tap interfaces.

  Linux commands to connect bridges to tap interfaces:

  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*

- All tap interfaces must be in the "up" state.

  Linux commands for bringing tap interfaces up:

  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**

- Verify that all tap interfaces are connected to bridges

  Linux commands to confirm that tap interfaces are connected to bridges:

  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
```

To bring up two Nexus 9000v platforms, connecting one interface each back to back, you can use the following commands as examples. The connection can be a socket-based or bridge-based connection. In this example, bridges are used to connect instances of management interface and one data port. Similarly, more Nexus 9000v data ports can be connected in the same way by adding more net device in the command line options. In this example, two interfaces each (interface mgmt0 and interface eth1/1) on both the Nexus 9000v instances are mapped.

For a Nexus 9000v first instance:

```
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
```

For a Nexus 9000v second instance:

```
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
```

The qemu-system-x86_64 or above KVM command is equivalent depending on how Linux is deployed. After successful invocation, you should be able to access both instances of the serial console via “telnet localhost 9000” or “telnet localhost 9100 respectively.

To pass traffic for LLDP and LACP multicast-specific packets through a Linux bridge, set the following values on all bridges connecting to each instance:

- Set LLDP and LACP communication between the VMs:

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

- Allow Multicast packet flow through the Linux bridge:

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

Nexus 9000v Deployment Workflow for ESXi

This section describes the steps required to deploy Nexus 9000v platforms on ESXi hypervisors. Three types of deployment are available:

- Common Deployment
- Platform-Specific Deployment
- Interconnecting Deployment

Common Deployment Workflow

Before you begin

The following procedure provisions a Cisco Nexus 9300v or 9500v platform in the ESXi hypervisor using the distributed OVA.

Ensure the following:

- You have installed the ESXi 8.0 hypervisor
- You have a valid license for ESXi 8.0 to run on both server and vCenter.
- The distributed OVA file has been downloaded to the desktop.

Step 1 Log into the ESXi vCenter.

Step 2 Right-click version 8.0 and select **Deploy OVF Template**.

Note Perform the self-guided instructions in the subsequent screens that appear.

Step 3 In the **Need name** screen, choose **Local file** and click **Browse**. Choose the downloaded distribute OVA file from your desktop.

Step 4 In the **need name** screen, choose the data center (or a folder and enter the VM name.

Step 5 In the **need name** screen, select an ESXi server for the Virtual Machine to be deployed into, and click **Finish** after the validation.

Step 6 In the **need name** screen, review the details, and click **Next**.

Step 7 In the **Configuration** screen click **Next**.

Step 8 In the **Select Storage** screen, select the data store, and click **Next**.

Step 9 In the **Select Networks** screen, ensure that the following values are selected:

- Source Network name - mgmt 0
- Destination Network - lab management LAN vSwitch

Don't select other vNIC destinations as the lab management LAN vSwitch. Failure to do so results in management connectivity issues because the Cisco Nexus 9000v data ports will conflict with the physical switches.

Step 10 In the **Ready to Complete** screen, click **Finish**, and wait for the completion of the process.

Step 11 Under the **Virtual Hardware** tab, select **Serial Port 1**. For the serial port type, select the **Use Network** panel, and select the following options:

- Direction - Server
- Port URL - telnet://0.0.0.0:1000, where 1000 is the unique port number in this server.

Note Nexus 9000v only supports E1000 network adapters. When you add any network adapter, verify that the adapter type is E1000.

Step 12 Under the **VM Options** tab, select the **Boot Options** panel, and choose **EFI**.

Step 13 Under the **VM Options** tab, select the **Advance** panel and in the **Edit Configuration** screen, add the following values using the **Add Configuration Params** option:

- Name - efi.serialconsole.enabled
- Value - TRUE

Click **OK** to view the boot up process in both the VGA and the serial console mode.

Note Nexus 9000v platforms require the serial console to be provisioned in order to access the switch prompt (although some of the initial grub boot messages are shown on VGA console). Ensure that the serial console is provisioned on the VM correctly. Successful bootup should show kernel boot up messages after "**Image Signature verification for Nexus9000v is not performed**" is displayed from the VGA or serial console if "efi.serialconsole.enabled=TRUE" is provisioned.

Step 14 Power on the virtual machine.

Platform Specific Workflow

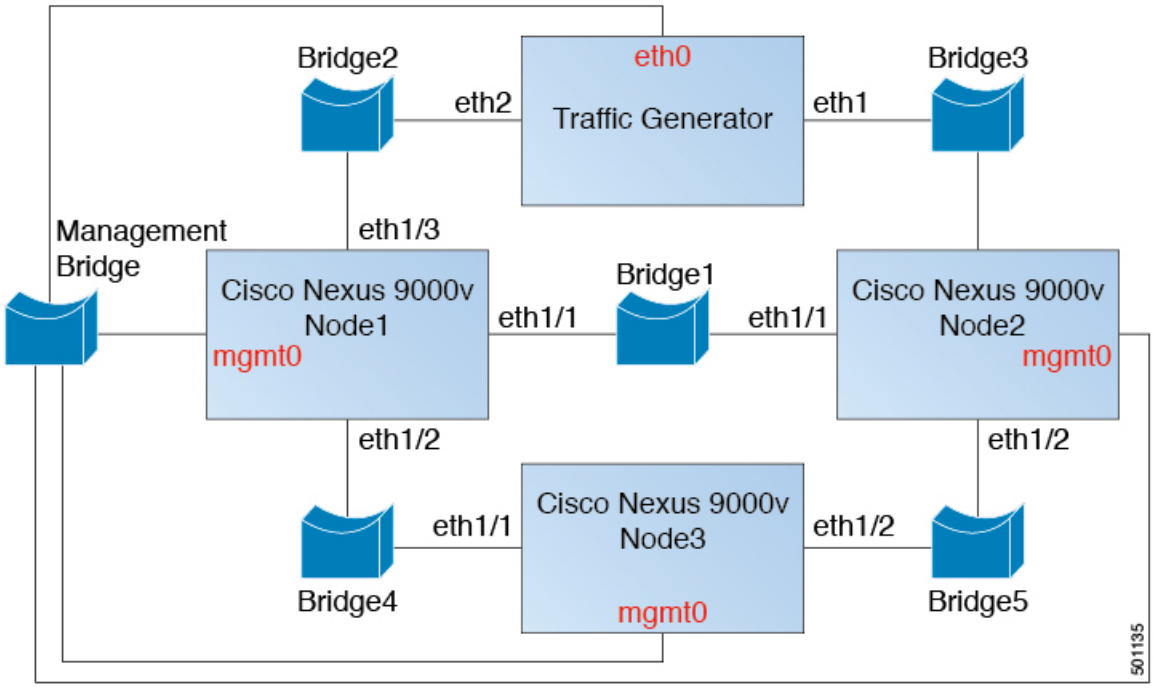
The Cisco Nexus 9500v runs in two different modes: sequential and mac-encoded mode. Nexus 9300v and Nexus 9500v sequential mode deployment steps are the exact same on ESXi hypervisor. The maximum number of interfaces for both platform types is 10 (one management port and nine data ports); this is a hypervisor limitation.

The Nexus 9500v emulates multiple-line-card interface traffic in single VM on ESXi hypervisor even though the total number of interfaces is limited to 10. If you choose to use the Nexus 9500v mac-encoded schema, change each network adapter MAC address to match slots and ports that are being emulated.

Interconnecting Platforms

Networking between Nexus 9300v and Nexus 9500v, or any other virtual platform, is based on vSwitch as the bridge on the ESXi hypervisor. You can have any topology as designed to simulate various customer use cases.

Figure 1: Interconnecting Cisco Nexus 9000v Platforms through EXSi



Nexus 9000v Deployment Workflow for Vagrant

This section describes the steps required to deploy Nexus 9000v platforms on Vagrant hypervisors. Three types of deployment are available:

- Common Deployment
- Platform-Specific Deployment
- Interconnecting Deployment

Common Deployment Workflow

You can't deploy the Cisco Nexus 9300v in the Vagrant/VBox environment. The virtual artifacts .box file is only available on distribution.

Platform Specific Workflow

Deploy the nexus9300v.9.3.3.IDI9.0.XXX.box on a VirtualBox. See the following customization guidelines and caveats for using Vagrant/Vbox:

- The user customization in Vagrant file isn't required.
- There's no need to change named pipe for Windows. Access the serial console using default port 2023, for both Mac or Windows. If needed, use this serial console via **telnet localhost 2023** to monitor the switch boot up process.
- The standard box process is used as any other appliance distribution. You can simply bring up a VM using the base box name.
- The box name can be changed to a different name other than "base" using the **config.vm.box** field from the Vagrant file.
- The bootstrap configuration is possible if you want to apply a different configuration on the switch, other than the existing generic configuration in **.box** from the release image file. In this case, use **vb.customize pre-boot**. For example:

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

- Customize the VM interface MAC address by using the **config.vm.base_mac** field. This modification must be performed prior to entering the **vagrant up** CLI command and after entering the **vagrant init** CLI command. If you want to modify the MAC address after entering the **vagrant up** CLI command, or after the VM is created, use the box commands to modify the VM.

Support for Sync Folder in Vagrant

Starting with Release 10.1(1), Nexus 9300v supports Vagrant sync folder with which a directory/folder on a host machine can be shared with a Nexus 9300v machine. The **vagrant up** command in the Vagrant scripts logs into the virtual box and mounts the directory based on user configuration in the Vagrantfile. By default, the Vagrant scripts use the *vagrant* username, and expect bash to be the login shell. In order to facilitate this feature, the default login shell for pre-configured *vagrant* username has been changed to bash. However, you have the option to change the default shell (for user *vagrant*) to NX-OS CLI with explicit configuration in Nexus or in the Vagrantfile .

By default, Vagrant mounts the current working directory on the host at `directory/vagrant` on the Nexus 9300v. If you do not want the current folder on host to be shared with the Nexus 9300v, you must include the following line in the Vagrantfile.

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

Sample Vagrantfile - when you want to share the host folder, say, **/home/james/my_shared_folder/** on Nexus 9300v at **/bootflash/home/vagrant**:

```
# -*- 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
```

Given below is Nexus 9300v platform-specific deployment example:

```
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$
```

Changing Default Shell to NX-OS CLI

When you need to login to NX-OS CLI, use one of these options:

- By manually executing the **vsh** command on bash prompt on every login.
- You may make use of a pre-packaged script in Nexus 9300v virtual box and execute it from Vagrantfile as shown below.

```
 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"
```

- You may login with username *admin* instead of username *vagrant* (Username *vagrant* is used by default when you use the **vagrant ssh** command)

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

Using Ansible with Nexus 9300v

Vagrant is a generic orchestrator which supports configuration and management of boxes with various *provisioners* such as, Ansible, Shell scripts, Ruby scripts, Puppet, Chef, Docker, Salt etc.

Vagrant file may contain sections for one (or more) provisioners along with its configurations. An example for Ansible, is shown here.

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

These provisioners are automatically triggered every time a virtual box boots up or when triggered manually with the **vagrant provision** command or with the **vagrant provision –provision-wth** command. Provide login credentials in an Ansible host config file for Ansible to log into the virtual box and execute NX-OS CLIs. Since Ansible would expect to see NX-OS CLI after logging in, you can use the pre-configured username *admin* or create a new username manually, and use it in the Ansible host configuration files.

Shutdown VM

Use the following to shutdown the VM:

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

Destroy VM for cleanup

Use the following if you want to completely delete the VM instance:

```
$ 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..
```

Interconnecting Platforms

Networking between Nexus 9300v and other virtual platforms, is based on VBox Internal Network. See the following connection diagram:

Figure 2: Interconnecting Cisco Nexus 9000v Platforms through Vagrant VM

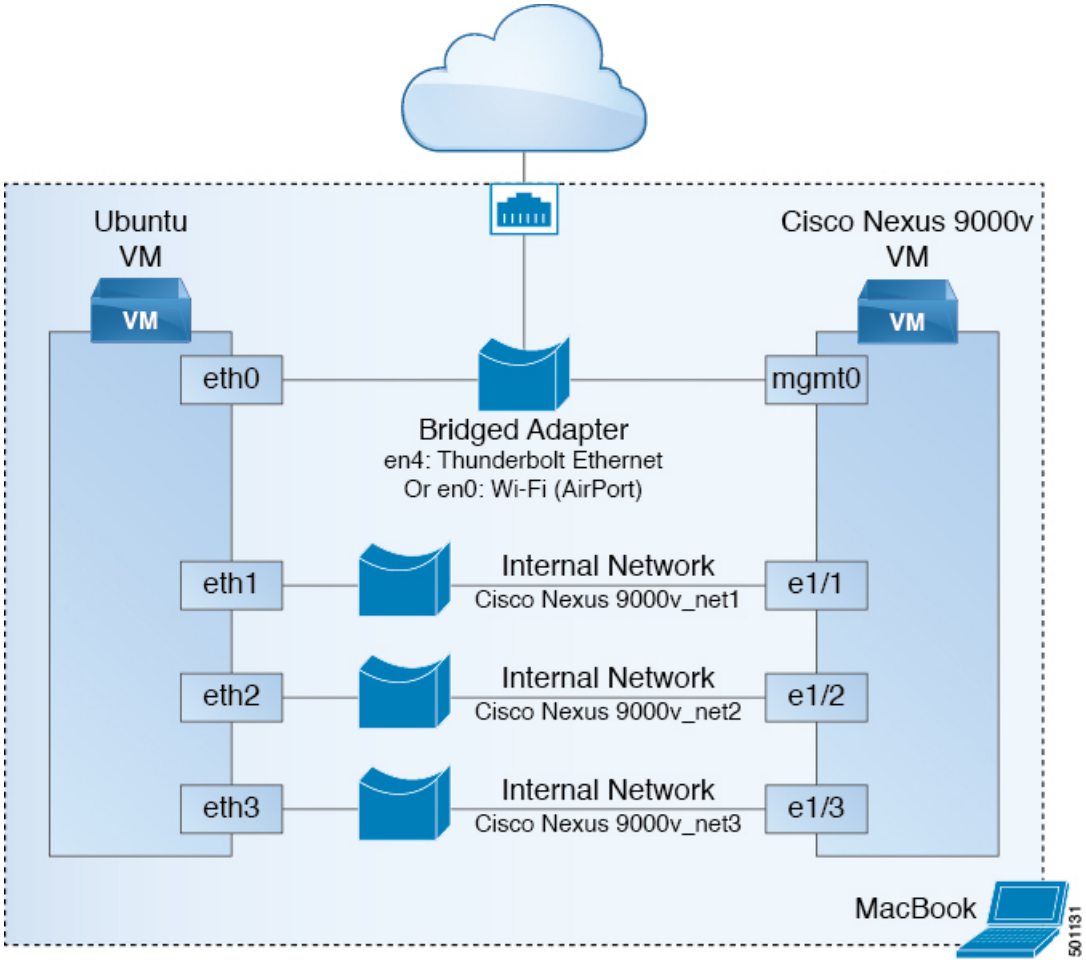


Image Upgrade Workflow

This section describes the typical upgrade steps for the Cisco Nexus 9000v platforms.

Deploying from a New Artifact

Depending on the environment, use the appropriate virtual artifact and refer to one of the following sections to deploy the VM:

- Nexus 9000v Deployment Workflow for KVM/QEMU, on page 2
- Nexus 9000v Deployment Workflow for ESXi, on page 7
- Nexus 9000v Deployment Workflow for Vagrant, on page 9

Upgrading from a New NX-OS Image

Nexus 9300v upgrades are only allowed from a VM created with virtual artifacts from Cisco Nexus 9000v, Release 9.3(1) and onwards. Before upgrading, ensure there's 400Mb + of new NX-OS binary image on the bootflash. To upgrade, copy the new binary to the bootflash and then upgrade using the standard NX-OS workflow (for example: ‘install all nxos bootflash:///<nxos.bin>’).

Nexus 9500v upgrades aren't supported as this is the first release of the platform.

For Nexus 9300v and 9500v lite, ISSU from earlier binary image to lite binary image is not supported. Even if you can bring up the image using cold boot, delete the previous configuration first, and then install the lite binary.