Infoblox

Deployment Guide

vNIOS deployment on KVM

January 2019

Table of Contents

Overview

Introduction

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, that provides the core virtualization infrastructure and a processor specific module, kvm-intel.ko or kvm-amd.ko.

Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter etc.

KVM is an open source software. The kernel component of KVM is included in mainline Linux, as of 2.6.20. The user space component of KVM is included in mainline QEMU, as of 1.3.

vNIOS for KVM

Infoblox provides vNIOS QCOW2 images which enables customers to deploy large, robust, manageable and cost effective Infoblox grids on KVM, that provide core network services.

Supported Operating Systems and versions

vNIOS for KVM is supported and validated on following Operating Systems.

- Ubuntu 14.04, 16.04 Server and Desktop Operating System
- Red Hat Enterprise Linux(RHEL) 7.x

vNIOS deployment on KVM

Preparing the environment

Installing KVM and Bridge utilities

1. vNIOS is supported and validated on Ubuntu 14.04, 16.04 and Red Hat Enterprise Linux(RHEL) 7.x based KVM **with atleast 2 NIC cards** connected.
   Note: This deployment guide covers vNIOS deployment on Ubuntu16.04 Desktop OS based KVM

2. Login to the Ubuntu host as a root user. Download the bridge utilities by running **`apt-get install bridge-utils.`**

3. Create 2 bridges infoblox-lan1 and infoblox-mgmt by running **`brctl addbr infoblox-lan1`** and **`brctl addbr infoblox-mgmt`** respectively.

```
root@ubuntu16-kvm:~# brctl addbr infoblox-lan1
root@ubuntu16-kvm:~# brctl addbr infoblox-mgmt
root@ubuntu16-kvm:~#
```

4. Add the first interface to the bridge **`infoblox-lan1`** and add the second interface to the bridge **`infoblox-mgmt`**, by editing the **`/etc/network/interfaces`** file using **`vim`** editor.

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto infoblox-lan1
iface infoblox-lan1 inet static
        bridge_ports ens160
        address 10.196.200.26
        netmask 255.255.255.0
        gateway 10.196.200.1
        dns-nameservers 10.120.3.10

auto infoblox-mgmt
iface infoblox-mgmt inet static
        bridge_ports ens192
        address 10.196.215.100
        netmask 255.255.255.0
        gateway 10.196.215.1
        dns-nameservers 10.120.3.10
```

5. Verify that the bridges are created and associated with the respective interface by running brctl show command

```
root@ubuntu18-kvm:/var/lib/libvirt/images/infoblox# brctl show
bridge name     bridge id               STP enabled     interfaces
infoblox-lan1           8000.005056817e5f       no              ens160
infoblox-mgmt           8000.005056813a8e       no              ens192
virbr0          8000.5254004c9d6a       yes             virbr0-nic
root@ubuntu18-kvm:/var/lib/libvirt/images/infoblox#
```

6. Restart the networking service by running **`/etc/init.d/networking`** command.

```
root@ubuntu16-kvm:~# /etc/init.d/networking restart
[ ok ] Restarting networking (via systemctl): networking.service.
root@ubuntu16-kvm:~#
```

Note: Sometimes network fails to restart . In such scenario reboot the ubuntu machine to implement the changes.

7. Run `ip a` command to verify that bridges are up.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master infoblox-lan1 state UP group default qlen 1000
    link/ether 00:50:56:81:7e:5f brd ff:ff:ff:ff:ff:ff
3: ens192: <BROADCAST,MULTICAST> mtu 1500 qdisc mq master infoblox-mgmt state DOWN group default qlen 1000
    link/ether 00:50:56:81:3a:8e brd ff:ff:ff:ff:ff:ff
4: infoblox-lan1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:50:56:81:7e:5f brd ff:ff:ff:ff:ff:ff
    inet 10.196.200.26/24 brd 10.196.200.255 scope global infoblox-lan1
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe81:7e5f/64 scope link
       valid_lft forever preferred_lft forever
5: infoblox-mgmt: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 00:50:56:81:3a:8e brd ff:ff:ff:ff:ff:ff
    inet 10.196.215.100/24 brd 10.196.215.255 scope global infoblox-mgmt
       valid_lft forever preferred_lft forever
    inet6 fe80::250:56ff:fe81:3a8e/64 scope link
       valid_lft forever preferred_lft forever
```

8. Install the KVM and bridge utilities packages by running apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils virt-manager -y command.

```
root@ubuntu16-kvm:~# apt-get install qemu-kvm libvirt-daemon-system libvirt-clients virtinst -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

9. Execute virsh list --all command to verify the installation. You should get output like this

```
root@ubuntu16-kvm:~# virsh list --all
 Id    Name                           State
----------------------------------------------------
```

Creating various types networks

Bridge Networks

A network bridge is a Link Layer device which forwards traffic between networks based on MAC addresses and is therefore also referred to as a Layer 2 device.

It makes forwarding decisions based on tables of MAC addresses which it builds by learning what hosts are connected to each network

In the context of KVM, a Linux bridge is used to connect the KVM guest interface to a KVM host network interface.

Creating Bridges

1. Login to the ubuntu KVM host as a root user and navigate to /var/lib/libvirt/images directory.
2. Create a lan1.xml file and add following lines of xml code

```
<network>
 <name>LAN1</name>
 <forward mode='bridge'/>
 <bridge name='infoblox-lan1' />
```

```
<network>
 <name>LAN1</name>
 <forward mode='bridge'/>
 <bridge name='infoblox-lan1' />
</network>
```

```
</network>
```

3. Create a mgmt.xml file and add following lines of xml code

```
<network>
 <name>MGMT</name>
 <forward mode='bridge'/>
 <bridge name='infoblox-mgmt' />
</network>
```

```
<network>
 <name>MGMT</name>
 <forward mode='bridge'/>
 <bridge name='infoblox-mgmt' />
</network>
```

4. Use `virsh net-define` command to define the `lan1` and `mgmt` networks from the corresponding xml files.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-define lan1.xml
Network LAN1 defined from lan1.xml

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-define mgmt.xml
Network MGMT defined from mgmt.xml
```

5. Use `virsh net-start LAN1` and `virsh net-start MGMT` command to activate and start networks.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-start LAN1
Network LAN1 started

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-start MGMT
Network MGMT started
```

6. Use virsh `net-list --all` command to verify the list and status of networks.

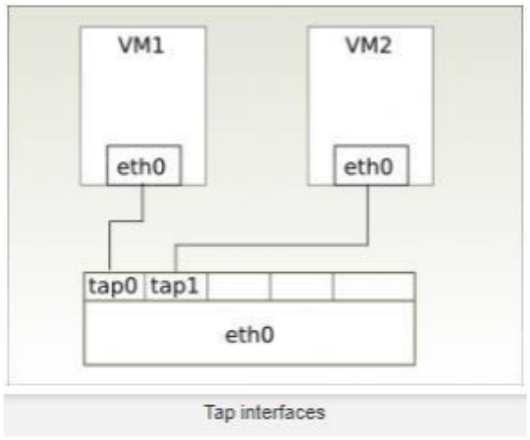
```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-list --all
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes
 LAN1                 active     no            yes
 MGMT                 active     no            yes
```

Macvtap Network

The Macvlan driver is a separate Linux kernel driver that the Macvtap driver depends on. Macvlan makes it possible to create virtual network interfaces that “cling on” a physical network interface.

Each virtual interface has its own MAC address distinct from the physical interface’s MAC address. Frames sent to or from the virtual interfaces are mapped to the physical interface, which is called the lower interface. Instead of passing frames to and from a physical Ethernet card, the frames are read and written by a user space program. The kernel makes the Tap interface available via the /dev/tapN device file, where N is the index of the network interface.

A Macvtap interface combines the properties of these two; it is a virtual interface with a tap-like software interface. A Macvtap interface can be created using the ip command.



1. Login to the ubuntu KVM host as a root user and navigate to **`/var/lib/libvirt/images`** directory
2. Create a **`lan1_macvtap.xml`** file with the following xml code.

```
<network>
 <name>LAN1_MACVTAP</name>
 <forward mode='bridge'/>
 <interface dev='lan1_bridge_name'/>
</forward>
</network>
```

```
<network>
 <name>LAN1_MACVTAP</name>
 <forward mode='bridge'>
 <interface dev='infoblox-lan1' />
  </forward>
</network>
```

3. Create a mgmt_macvtap.xml file with the following xml code.

```
<network>
 <name>MGMT_MACVTAP</name>
 <forward mode='bridge'/>
 <interface dev='mgmt_bridge_name'/>
</forward>
</network>
```

```
<network>
 <name>MGMT_MACVTAP</name>
 <forward mode='bridge'>
 <interface dev='infoblox-mgmt'/>
 </forward>
</network>
```

4. Use **`virsh net-define`** command to define the lan1 and mgmt macvtap networks from the corresponding xml files.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-define lan1_macvtap.xml
Network LAN1_MACVCTAP defined from lan1_macvtap.xml

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-define mgmt_macvtap.xml
Network MGMT_MACVTAP defined from mgmt_macvtap.xml

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox#
```

5. Use **`virsh net-start LAN1_MACVTAP`** and **`virsh net-start MGMT_MACVTAP`** command to activate and start macvtap networks.

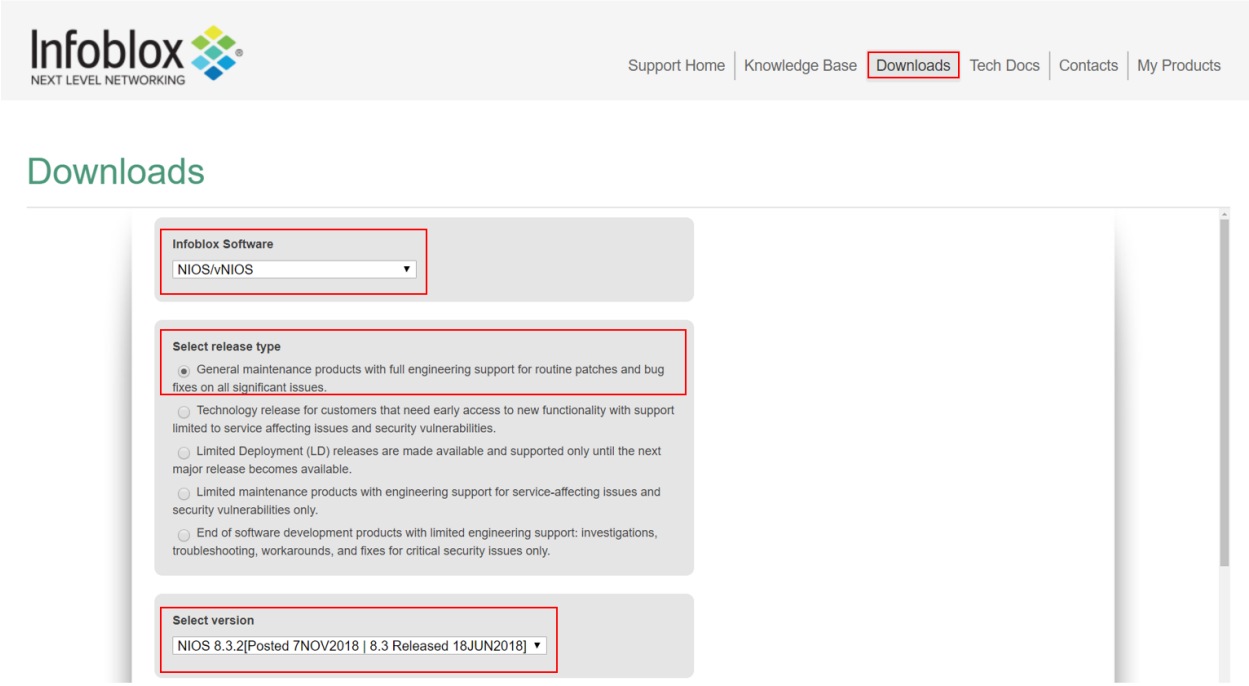
```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-start LAN1_MACVCTAP
Network LAN1_MACVCTAP started

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-start MGMT_MACVTAP
Network MGMT_MACVTAP started
```

6. Use virsh `net-list --all` command to verify the list and status of networks.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh net-list --all
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     yes           yes
 LAN1                 active     no            yes
 LAN1_MACVCTAP        active     no            yes
 MGMT                 active     no            yes
 MGMT_MACVTAP         active     no            yes
```

Downloading vNIOS QCOW2 image

1. Login to the https://support.infoblox.com portal. Navigate to **Downloads**. In the **Infoblox Software** drop-down menu select **NIOS/vNIOS**. Under **Select release type** select the first option. In the **Select version** drop box select **NIOS 8.3.2**



2. Scroll down and expand **vNIOS for KVM** section. Click on **Use for DDI** option to download **vNIOS QCOW2** DDI image

vNIOS for KVM

The Infoblox vNIOS for KVM is a virtual appliance designed for KVM (Kernel-based Virtual Machine) hypervisor and KVM-based OpenStack deployments. The Infoblox vNIOS for KVM functions as a hardware virtual machine guest on the Linux system. It provides core network services and a framework for integrating all components of the modular Infoblox solution. You can configure some of the supported vNIOS for KVM appliances as independent or HA (high availability) Grid Masters, Grid Master Candidates, and Grid members. For information about vNIOS for KVM hypervisor, refer to the Infoblox Installation Guide for vNIOS for KVM Hypervisor and KVM-based OpenStack.	
Grid Role	A qcow2 format disk image.
Member or Master	**IB-TE-V1410 160G** **IB-TE-V1420 160G** **IB-TE-V2210 160G** **IB-TE-V2220 160G** **IB-TE-V4010 160G**
Member	**IB-TE-V100 55G** **IB-TE-V810 55G** **IB-TE-V1410 55G** **IB-TE-V820 55G** **Cloud Platform** **CP-V800 160G** **CP-V1400 160G** **CP-V2200 160G**
Network Insight	**ND-V800 160G** **ND-V1400 160G** **ND-V2200 160G**
Reporting	**IB-TE-V800-300G disk1** **IB-TE-V800-300G disk2** **IB-TE-V1400 500G disk1** **IB-TE-V1400 500G disk2**
Member, Grid Master, and Reporting	**Use for DDI: v815, v825, v1415, v1425, v2215, v2225, v4015, Flex and Reporting: v805, v1405, v2205, v5005**
Discovery	**Use for Discovery: ND-v805, ND-v1405, ND-v2205**

Copying vNIOS qcow2 image

1. Create a directory **infoblox** under `/var/lib/libvirt/images` location and copy vNIOS qcow2 image to this location.

```
root@ubuntu16-kvm:/var/lib/libvirt/images# ls
infoblox
root@ubuntu16-kvm:/var/lib/libvirt/images# cd infoblox/
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# ls
lan1.xml  mgmt.xml  nios-8.3.2-376768-2018-11-02-02-41-25-ddi.qcow2
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox#
```

Deploying vNIOS with Bridge Networking

vNIOS can be deployed on KVM using either of the following approaches.

Deploying vNIOS through xml file with Bridge Networking

1. Login to the ubuntu KVM host as a root user and navigate to /var/lib/libvirt/images directory
2. Create a vNIOS.xml file and add the following code to it. Change the values marked in red.

```
<domain type='kvm' id='2'>
  <name>vnios_instance_name</name>
  <memory unit='KiB'> vnios_memory_kb </memory>
  <currentMemory unit='KiB'>vnios_memory_kb</currentMemory>
  <vcpu placement='static'>number_of_vcpus</vcpu>
  <resource>
    <partition>/machine</partition>
  </resource>
  <os>
    <type arch='x86_64' machine='pc-i440fx-bionic'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <vmport state='off'/>
  </features>
  <cpu mode='custom' match='exact' check='full'>
    <model fallback='forbid'>Broadwell</model>
    <feature policy='require' name='vme'/>
    <feature policy='require' name='f16c'/>
    <feature policy='require' name='rdrand'/>
    <feature policy='require' name='hypervisor'/>
    <feature policy='require' name='arat'/>
    <feature policy='disable' name='erms'/>
    <feature policy='require' name='xsaveopt'/>
    <feature policy='require' name='abm'/>
  </cpu>
  <clock offset='utc'>
    <timer name='rtc' tickpolicy='catchup'/>
    <timer name='pit' tickpolicy='delay'/>
    <timer name='hpet' present='no'/>
  </clock>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <pm>
    <suspend-to-mem enabled='no'/>
```

```
    <suspend-to-disk enabled='no'/>
  </pm>
  <devices>
    <emulator>/usr/bin/kvm-spice</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='qcow2'/>
      <source file='absolute_path_of_vnios_qcow2_image'/>
      <backingStore/>
      <target dev='vda' bus='virtio'/>
      <alias name='virtio-disk0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x08'
function='0x0'/>
    </disk>
    <controller type='usb' index='0' model='ich9-ehci1'>
      <alias name='usb'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
function='0x7'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci1'>
      <alias name='usb'/>
      <master startport='0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
function='0x0' multifunction='on'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci2'>
      <alias name='usb'/>
      <master startport='2'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
function='0x1'/>
    </controller>
    <controller type='usb' index='0' model='ich9-uhci3'>
      <alias name='usb'/>
      <master startport='4'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
function='0x2'/>
    </controller>
    <controller type='pci' index='0' model='pci-root'>
      <alias name='pci.0'/>
    </controller>
    <controller type='virtio-serial' index='0'>
      <alias name='virtio-serial0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x07'
function='0x0'/>
    </controller>
    <interface type='bridge'>
      <source network='LAN1' bridge='infoblox-lan1'/>
      <model type='virtio'/>
    </interface>
    <serial type='pty'>
      <source path='/dev/pts/1'/>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
      <alias name='serial0'/>
    </serial>
    <console type='pty' tty='/dev/pts/1'>
      <source path='/dev/pts/1'/>
      <target type='serial' port='0'/>
      <alias name='serial0'/>
```

```
    </console>
    <channel type='unix'>
                                                <source       mode='bind'
path='/var/lib/libvirt/qemu/channel/target/domain-2-GM-1/org.qemu.gues
t_agent.0'/>
                  <target   type='virtio'   name='org.qemu.guest_agent.0'
state='disconnected'/>
      <alias name='channel0'/>
      <address type='virtio-serial' controller='0' bus='0' port='1'/>
    </channel>
    <channel type='spicevmc'>
                     <target   type='virtio'   name='com.redhat.spice.0'
state='disconnected'/>
      <alias name='channel1'/>
      <address type='virtio-serial' controller='0' bus='0' port='2'/>
    </channel>
    <input type='tablet' bus='usb'>
      <alias name='input0'/>
      <address type='usb' bus='0' port='1'/>
    </input>
    <input type='mouse' bus='ps2'>
      <alias name='input1'/>
    </input>
    <input type='keyboard' bus='ps2'>
      <alias name='input2'/>
    </input>
               <graphics    type='spice'    port='5900'    autoport='yes'
listen='127.0.0.1'>
      <listen type='address' address='127.0.0.1'/>
      <image compression='off'/>
    </graphics>
    <sound model='ich6'>
      <alias name='sound0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x05'
function='0x0'/>
    </sound>
    <video>
           <model  type='qxl'  ram='65536'  vram='65536'  vgamem='16384'
heads='1' primary='yes'/>
      <alias name='video0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x02'
function='0x0'/>
    </video>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir0'/>
      <address type='usb' bus='0' port='2'/>
    </redirdev>
    <redirdev bus='usb' type='spicevmc'>
      <alias name='redir1'/>
      <address type='usb' bus='0' port='3'/>
    </redirdev>
    <memballoon model='virtio'>
      <alias name='balloon0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x09'
function='0x0'/>
    </memballoon>
  </devices>
  <seclabel type='dynamic' model='apparmor' relabel='yes'>
    <label>libvirt-9e92c826-7207-4fc4-bf6f-cf115aa02a24</label>
```

```
<imagelabel>libvirt-9e92c826-7207-4fc4-bf6f-cf115aa02a24</imagelabel>
  </seclabel>
  <seclabel type='dynamic' model='dac' relabel='yes'>
    <label>+64055:+130</label>
    <imagelabel>+64055:+130</imagelabel>
  </seclabel>
</domain>
```

3. Deploy vNIOS instance from above mentioned xml file by running **`virsh create vNIOS.xml`** command.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh create vNIOS.xml
Domain Grid-Master created from vNIOS.xml
```

4. Verify that vNIOS instance has been created and is running by running **`virsh list --all`** command.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh list --all
 Id    Name                           State
----------------------------------------------------
 9     Grid-Master                    running
```

5. To login to the console of vNIOS, use **`virsh console`** ***`instance_id`*** command.
   Note: After running the command `virsh console` `instance_id`, hit enter key multiple times to get the console prompt.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh console 9
Connected to domain Grid-Master
Escape character is ^]



Disconnect NOW if you have not been expressly authorized to use this system.
login:
```

6. To exit the console press “ctrl” and “]” key simultaneously.
7. To delete vNIOS instance use **`virsh destroy`** ***`instance_id`*** command.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh destroy 9
Domain 9 destroyed

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox#
```

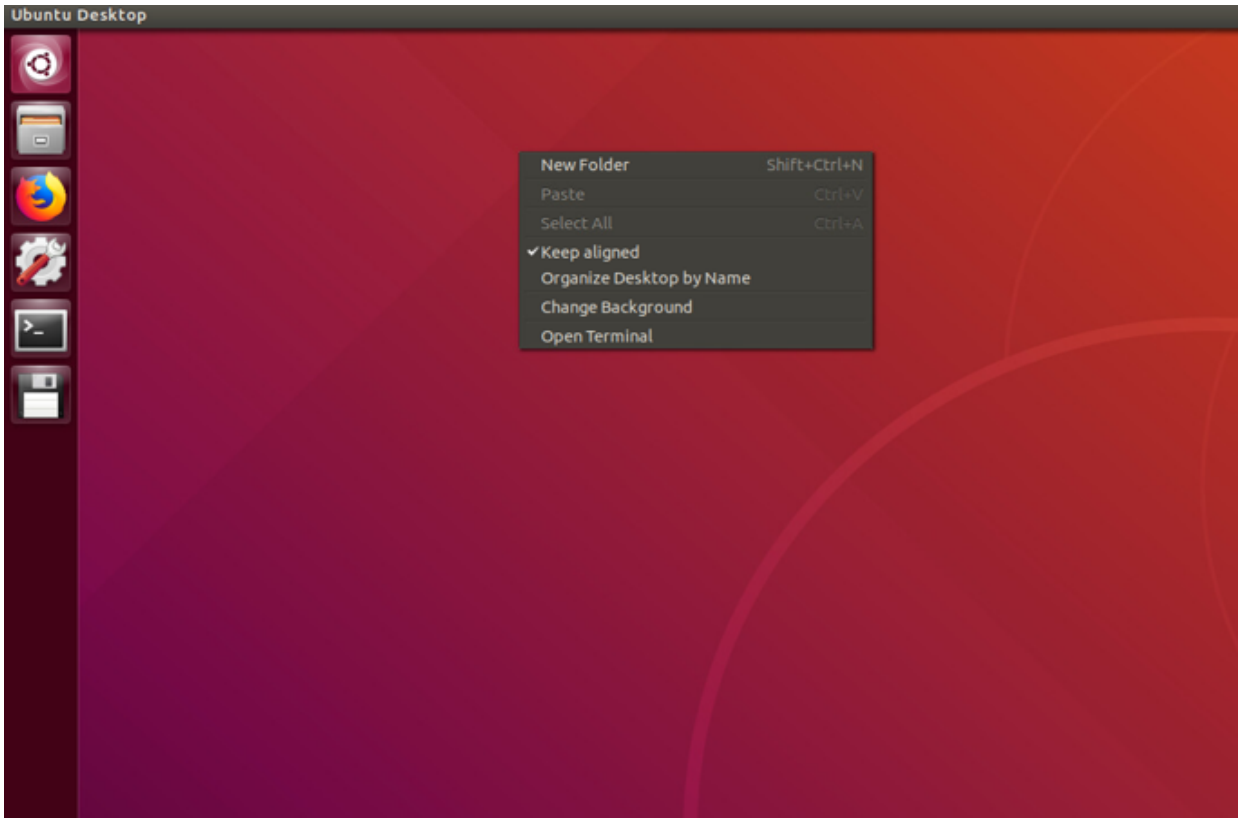
Deploying vNIOS instance using virt-manager (GUI based approach) with Bridge Networking

The virt-manager application is a desktop user interface for managing virtual machines through libvirt. It primarily targets KVM VMs, but also manages Xen and LXC (linux containers).

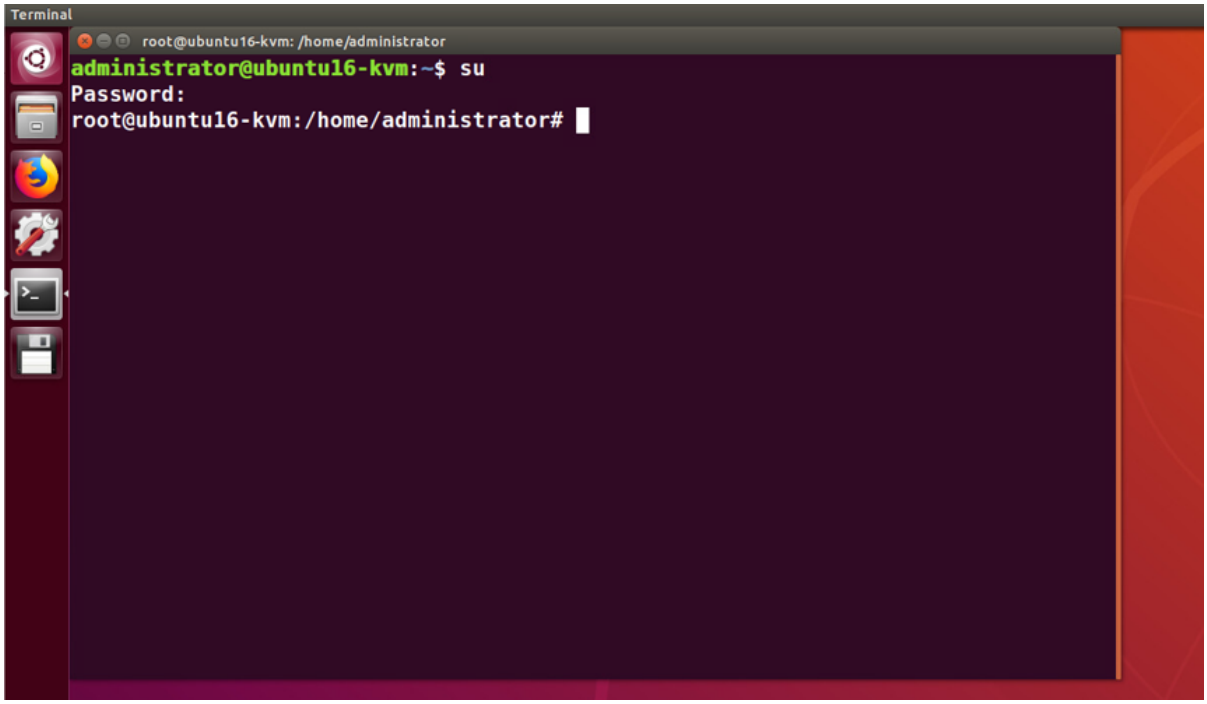
It presents a summary view of running domains, their live performance & resource utilization statistics. Wizards enable the creation of new domains, and configuration & adjustment of a domain's resource allocation & virtual hardware.

An embedded VNC and SPICE client viewer presents a full graphical console to the guest domain.

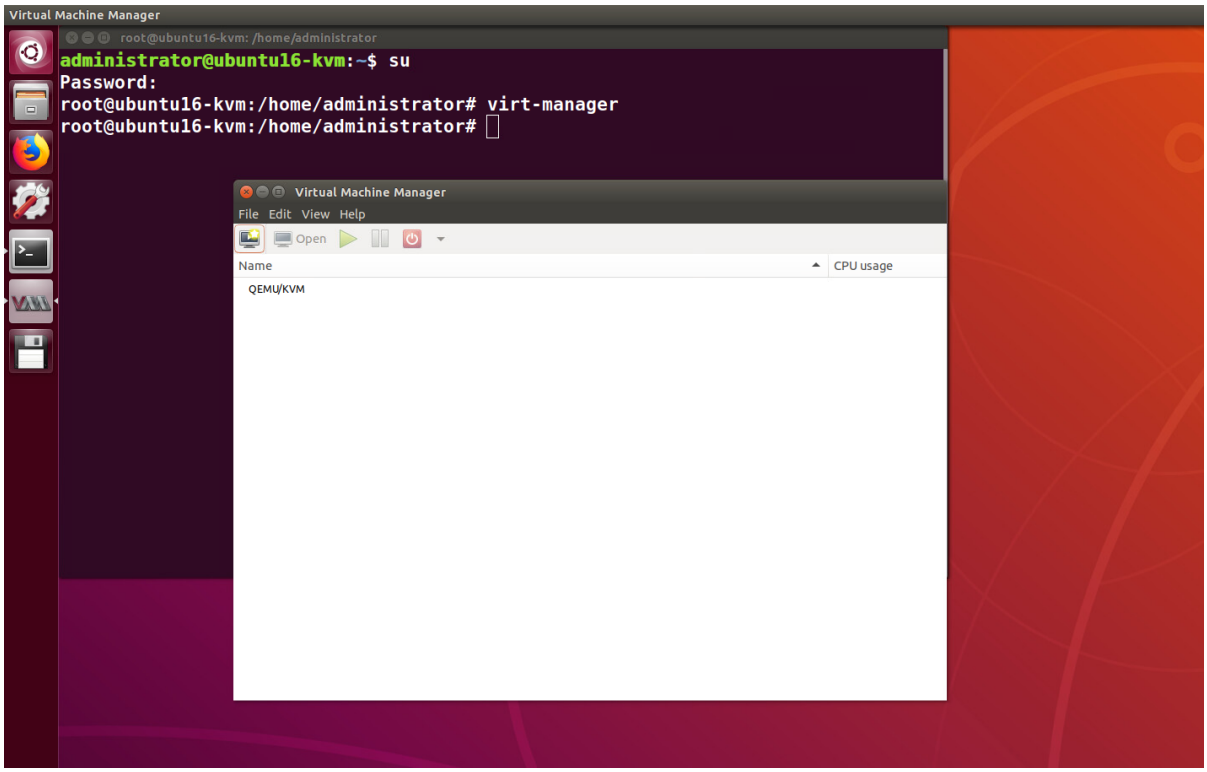
1. Login to ubuntu desktop host, right click on the desktop and select **Open Terminal**



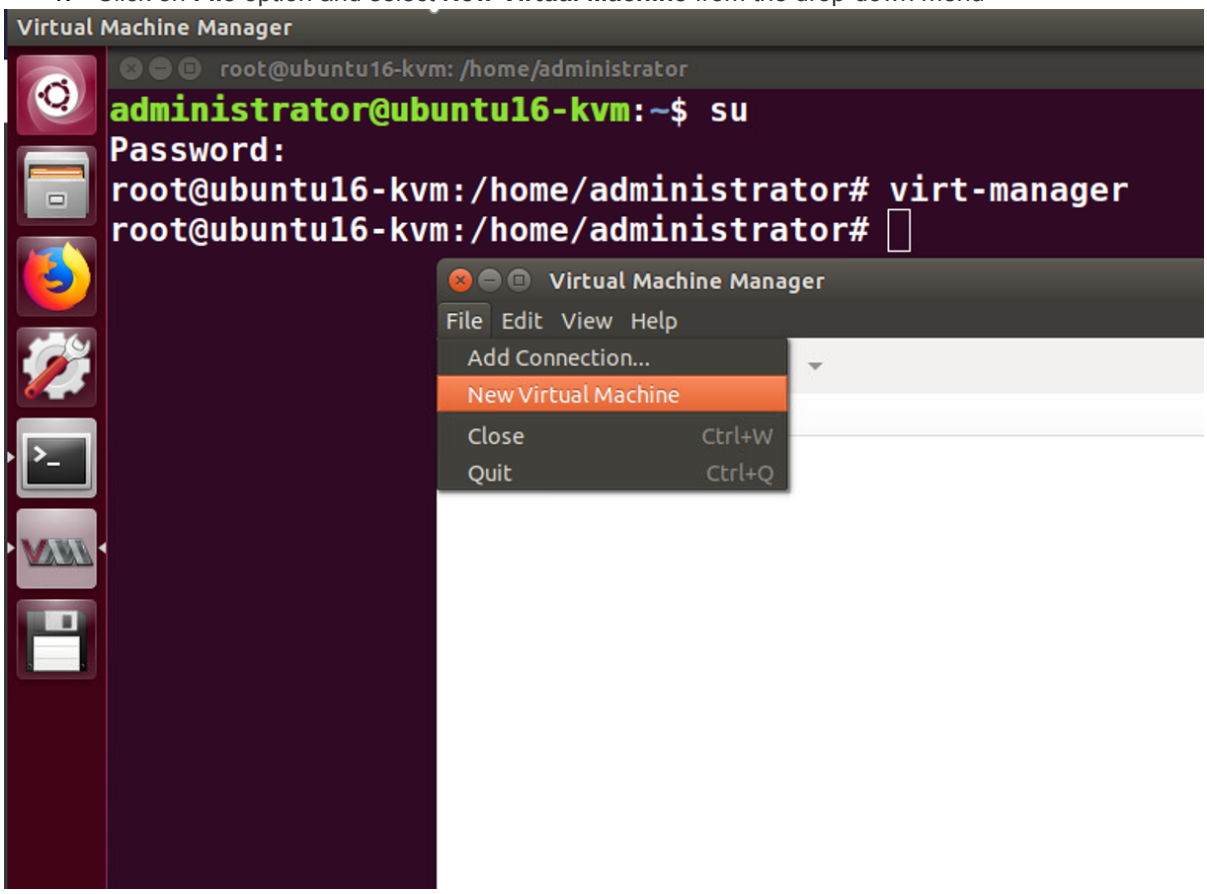
2. In the terminal session switch to root user using `su` command.



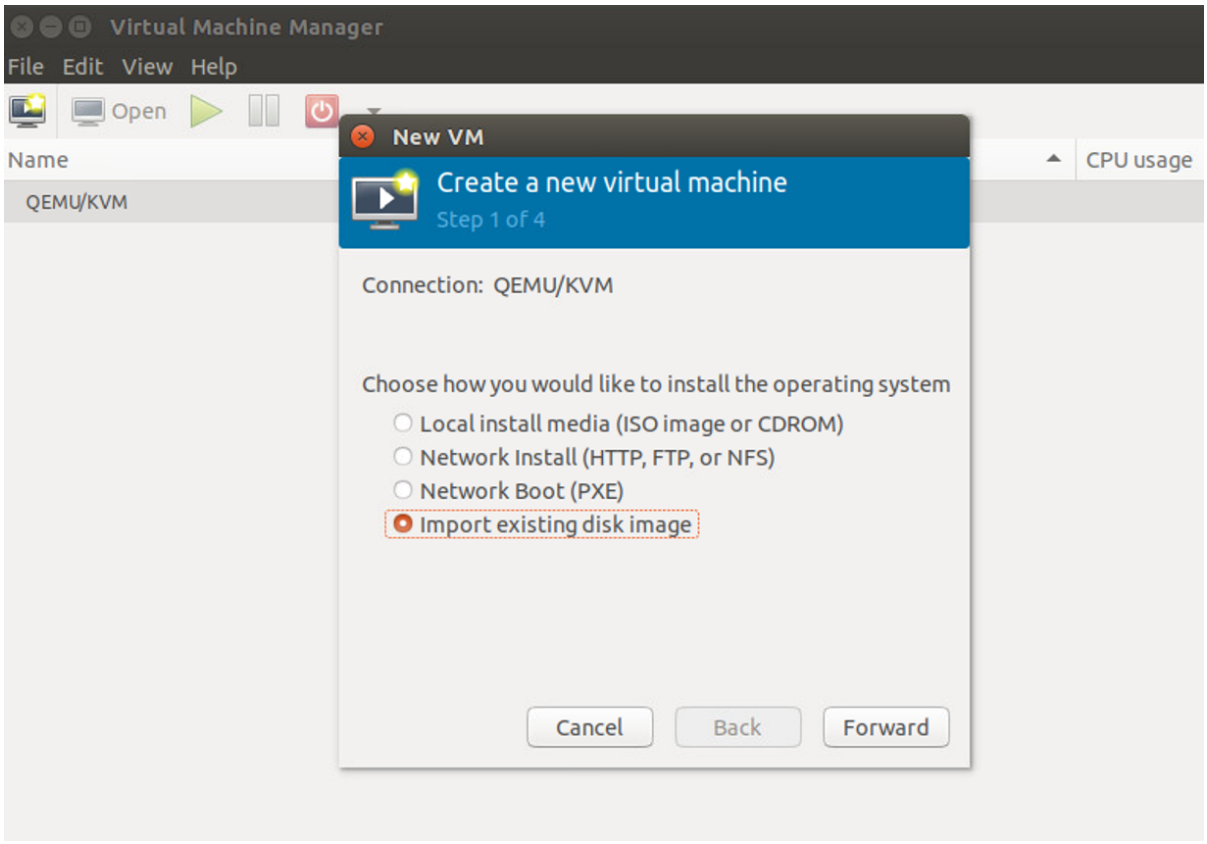
3. Execute **`virt-manager`** command to open Virt-Manager GUI



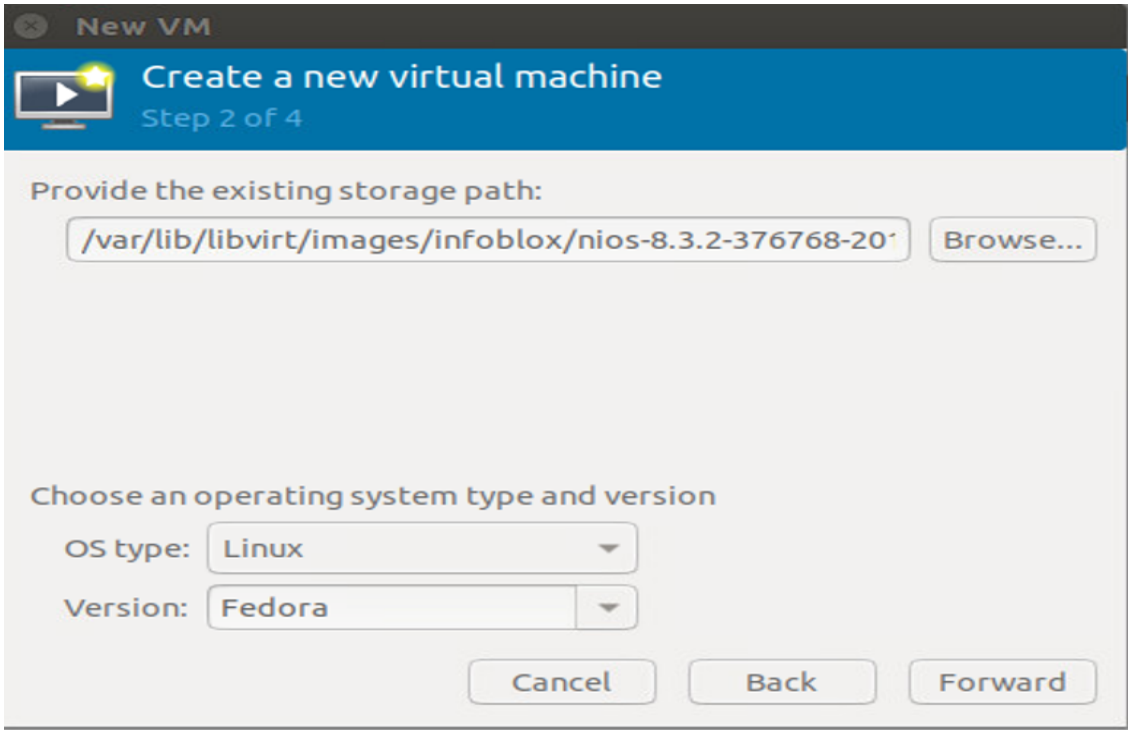
4. Click on **File** option and select **New Virtual Machine** from the drop-down menu



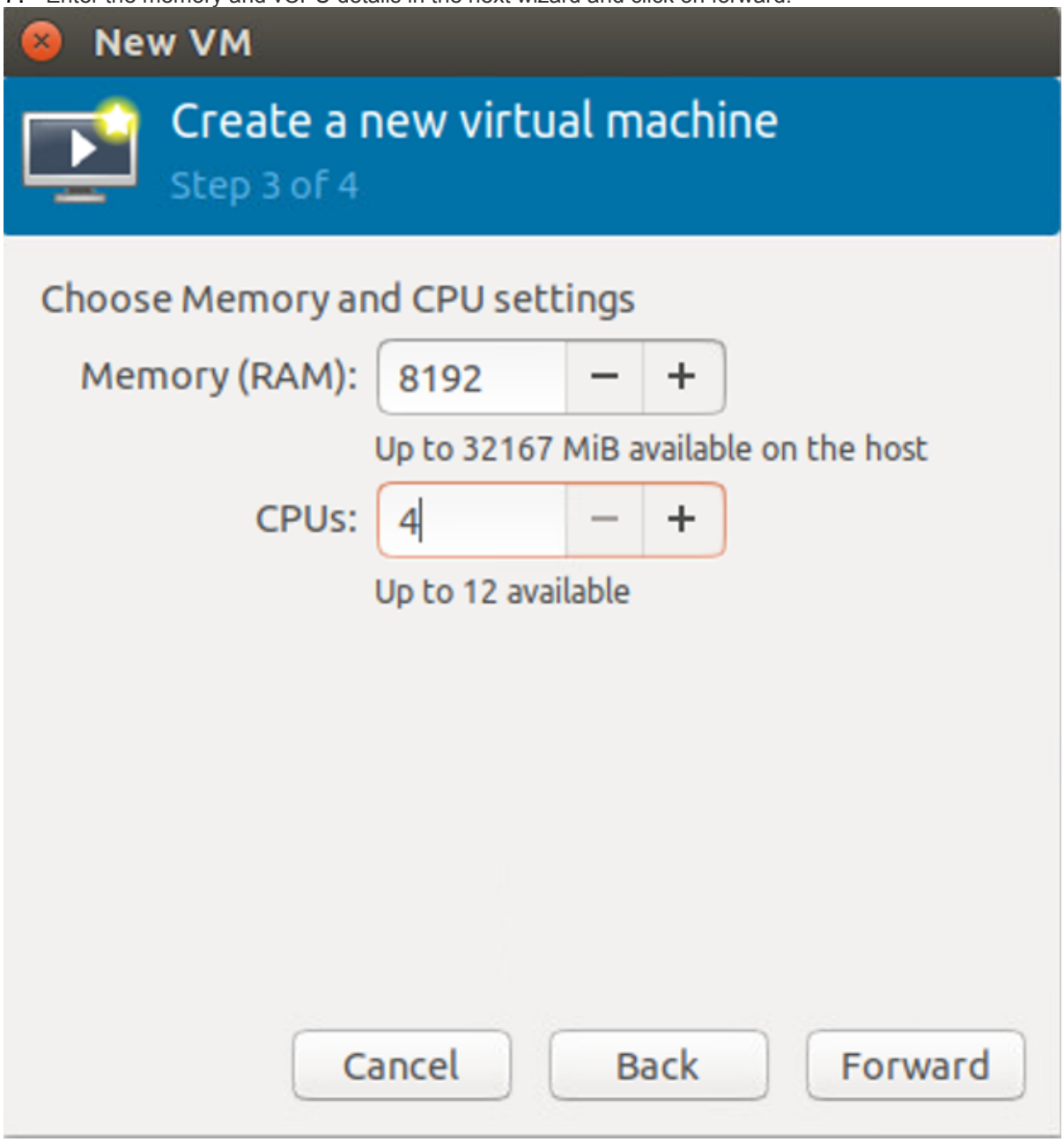
5. Select **Import existing disk image** in the next wizard and click on forward.



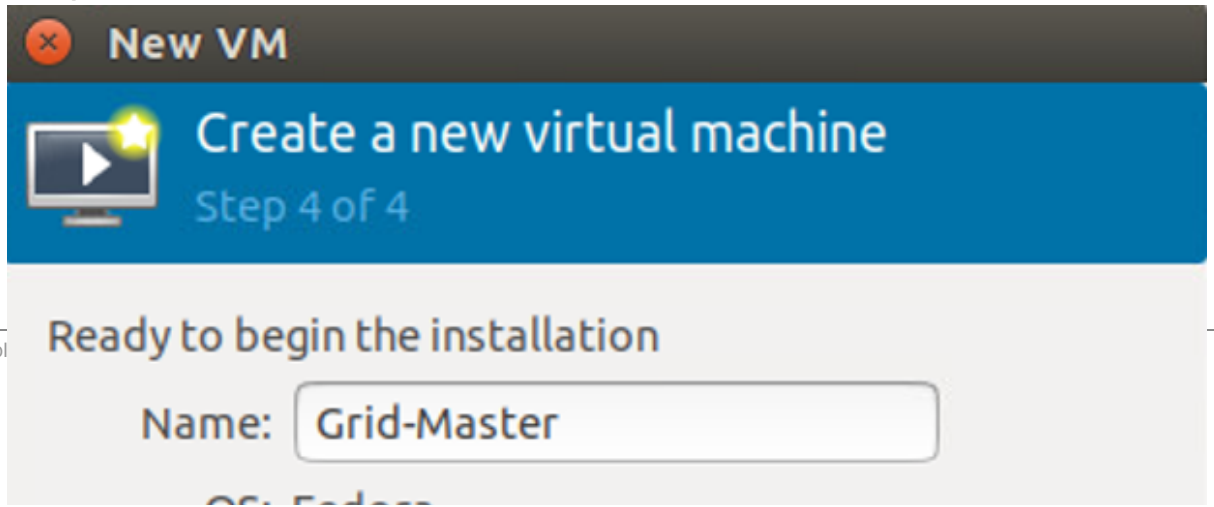
6. Click on **Browse** to select the vNIOS qcow2 image. Select **Linux** from the **OS type** drop-down box and **Fedora** from the **Version** drop-down box. Click on forward.



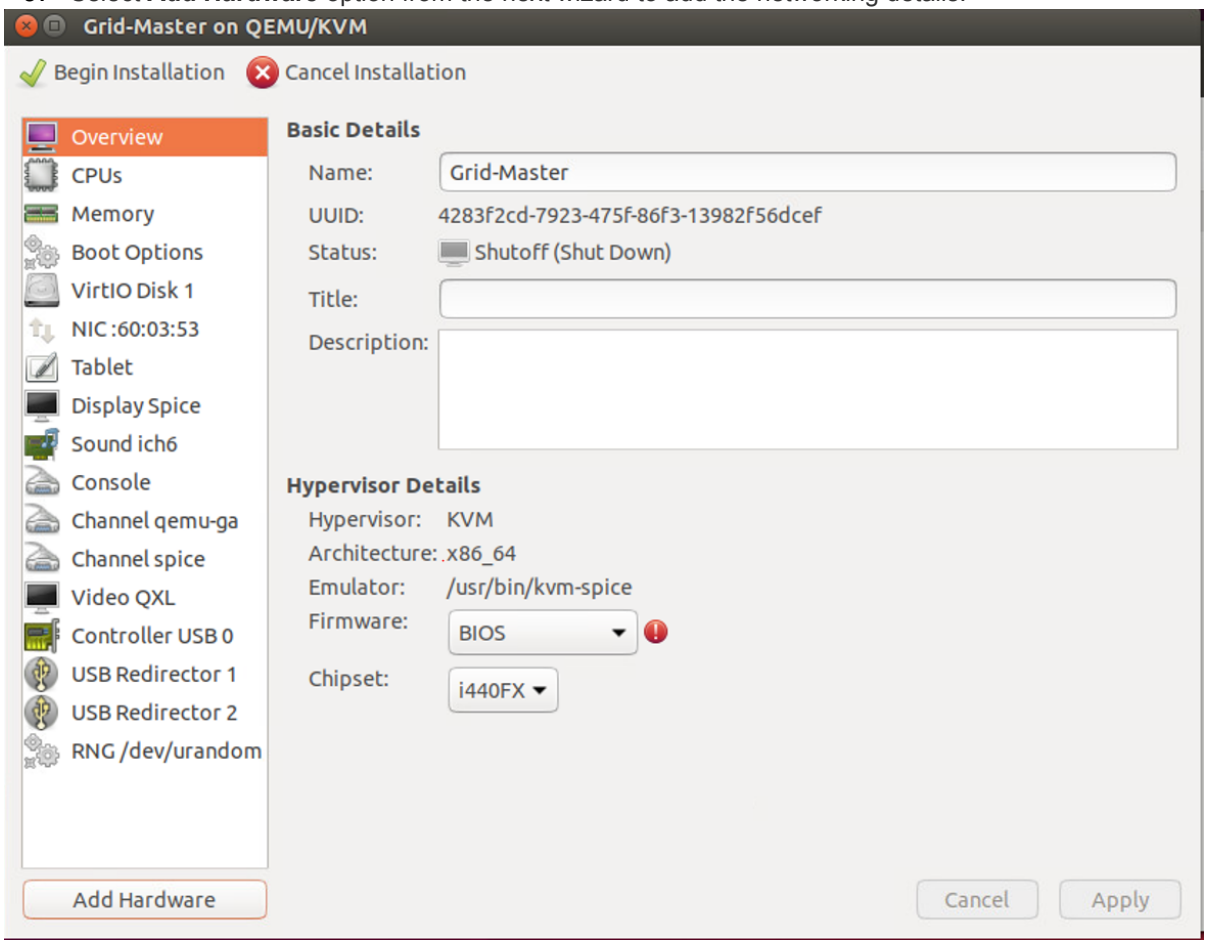
7. Enter the memory and vCPU details in the next wizard and click on forward.



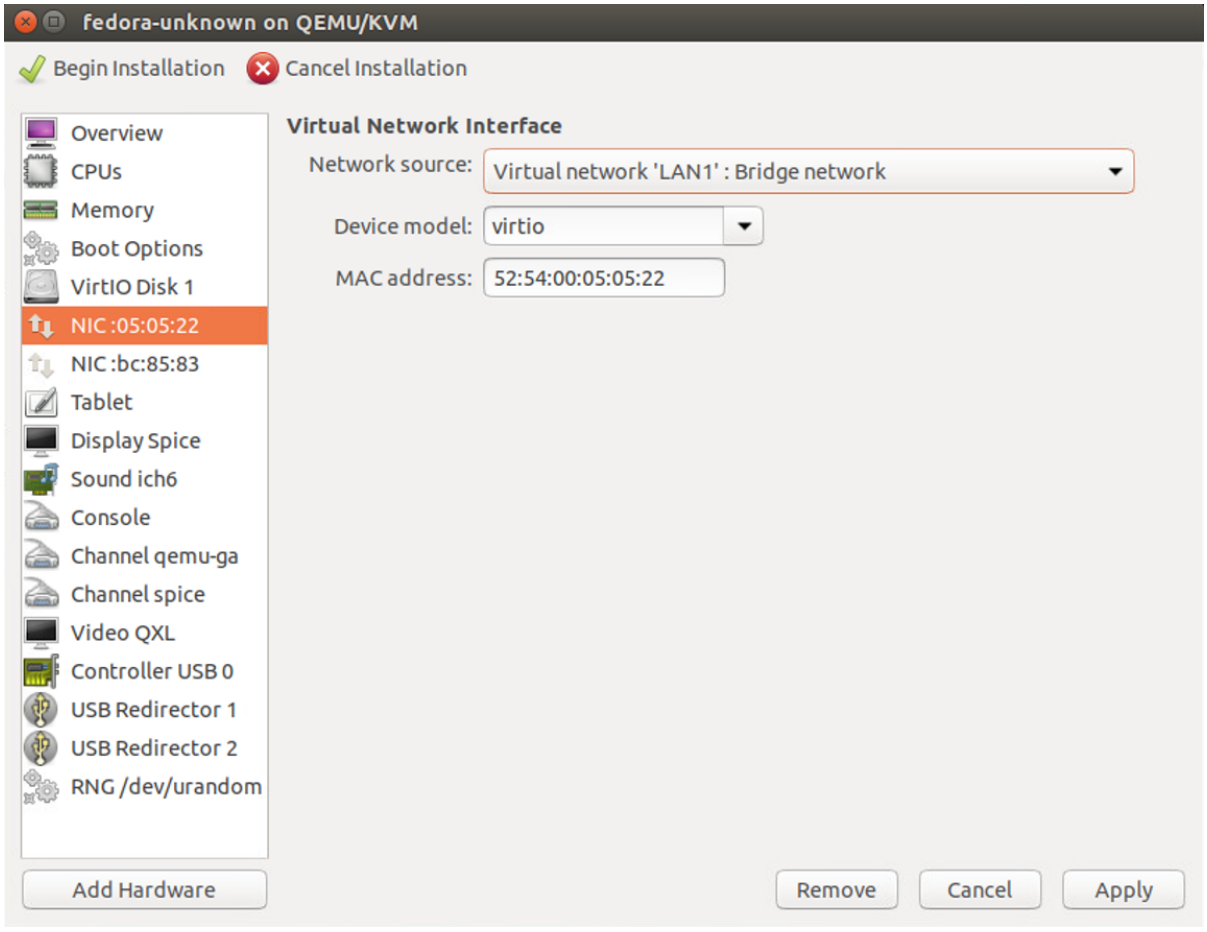
8. Enter the name of vNIOS instance and check the **Customize configuration before install option**. Click on Finish



9. Select **Add Hardware** option from the next wizard to add the networking details.

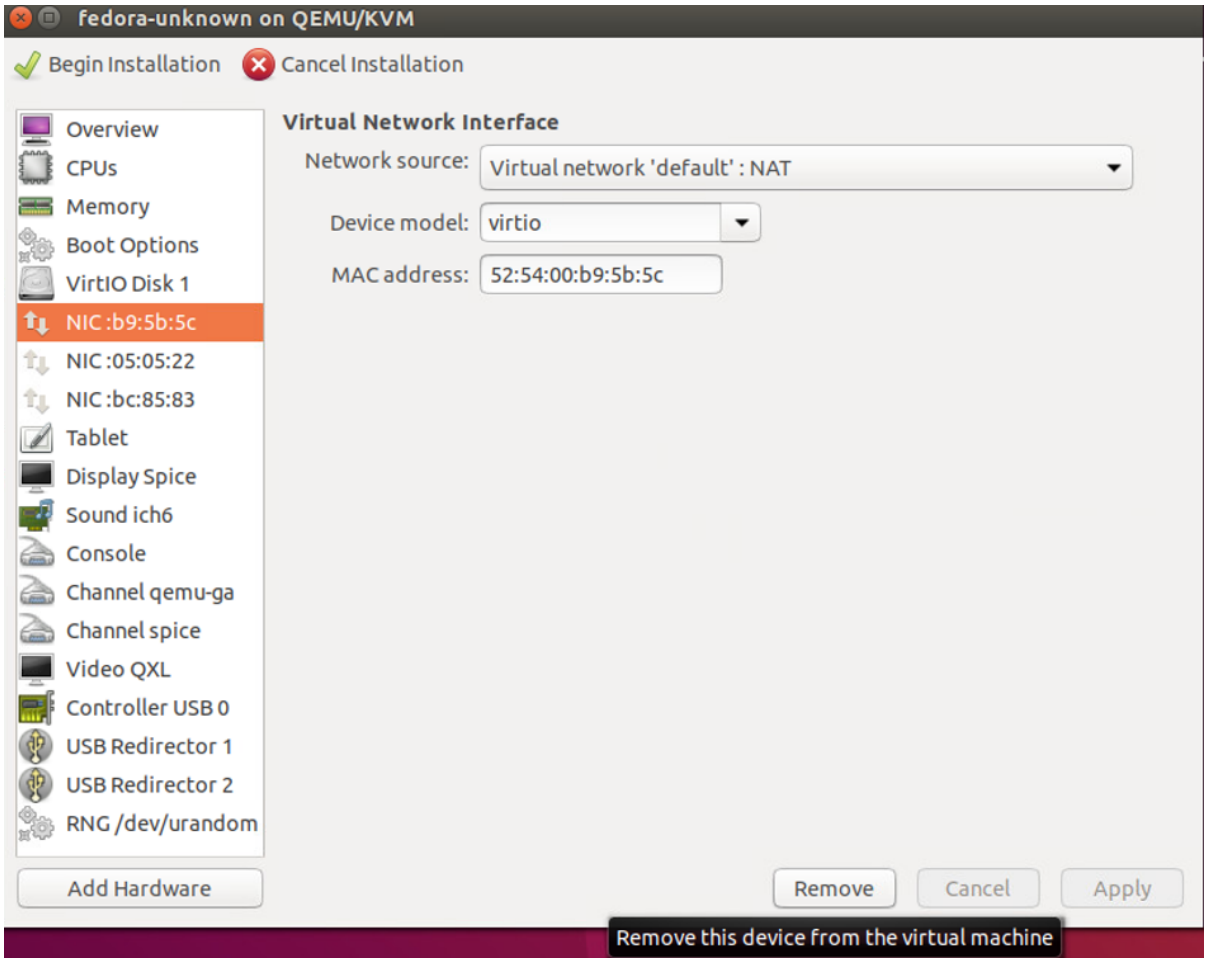


10. Click on Network from Add New Virtual Hardware wizard. From the **Network source** drop-down box select **Virtual network 'LAN1': Bridge network**. In the **Device mode** option select **virtio**. Click on **Finish**



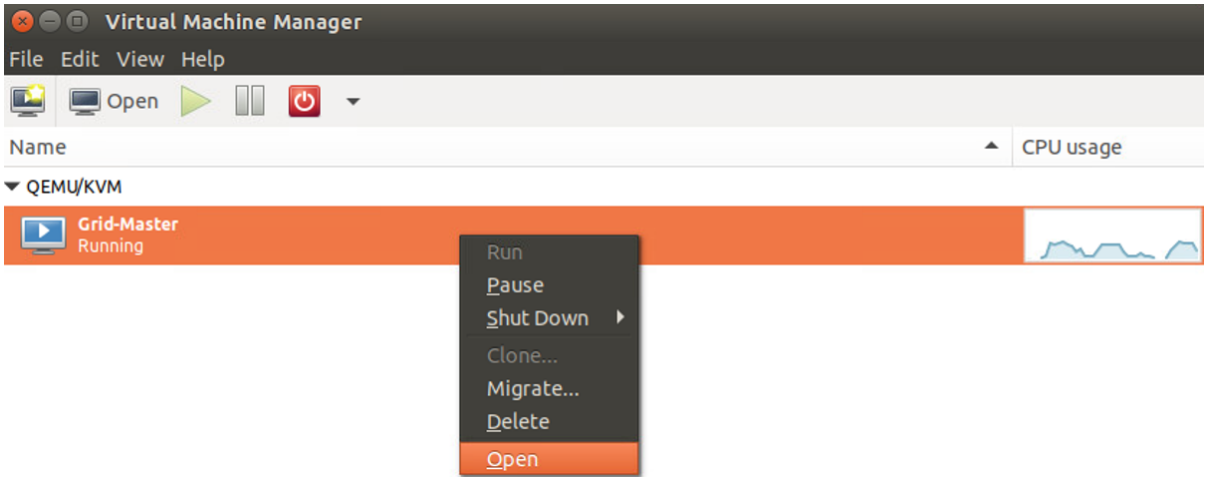
11. Repeat the same step to add **Virtual network 'LAN1': MGMT Bridge network**.

12. After adding Bridge interfaces, remove the first interface which gets added by default.



13. Click on **Begin Installation** to start vNIOS deployment.

14. Once the instance is up and running, right click on the instance and select **Open** to access the console.



Deploying vNIOS instance using virt-install utility with Bridge Networking

virt-install is a command line tool for creating new KVM, Xen, or Linux container guests using the "libvirt" hypervisor management library.

Given suitable command line arguments, "virt-install" can run completely unattended, with the guest 'kickstarting' itself too.

1. Login to the ubuntu KVM host as root user.
2. Run the following virt-install command to deploy vNIOS instance.
3. **virt-install --name *vnios-instance-name* --ram=*ram-in-mb* --vcpus=*number-of-vcpus* --disk path=*absolute-path-of-vnios-qcow2-image*,size=250 --boot hd --import --network network:MGMT --network network:LAN1 --os-type=linux**

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virt-install --name Grid-Master --ram=8192 --vcpus=8 --disk path=/var/lib/
libvirt/images/infoblox/nios-8.3.2-376768-2018-11-02-02-41-25-ddi.qcow2,size=250 --boot hd --import --network network:MGMT --n
etwork network:LAN1 --os-type=linux --os-variant=rhel7
WARNING  Graphics requested but DISPLAY is not set. Not running virt-viewer.
WARNING  No console to launch for the guest, defaulting to --wait -1

Starting install...
Domain installation still in progress. Waiting for installation to complete.
```

--os-variant=rhel7

4. Run **virsh list --all** command to verify that the vNIOS is deployed and is in running state.
5. To login to the console of the vNIOS instance use **virsh console *instance-id***
6. To exit the console press “ctrl” and “]” key simultaneously.

Deploying vNIOS with Macvtap Networking

vNIOS can be deployed on KVM using either of the following approaches.

Deploying vNIOS through xml file with Macvtap Networking

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh console 21
Connected to domain Grid-Master
Escape character is ^]

Good Bye


Disconnect NOW if you have not been expressly authorized to use this system.
login:
```

1. Login to the ubuntu KVM host as a root and navigate to **/var/lib/libvirt/images** directory
2. Create a **vNIOS-macvtap.xml** file and add the following code to it. Change the values marked in red

```
      <domain type='kvm' id='22'>
        <name>vnios_instance_name</name>
        <memory unit='KiB'>memory_in_kb</memory>
        <currentMemory unit='KiB'>memory_in_kbm</currentMemory>
        <vcpu placement='static'>number_of_vcpus</vcpu>
        <resource>
          <partition>/machine</partition>
        </resource>
        <os>
<type arch='x86_64' machine='pc-i440fx-bionic'>hvm</type>
          <boot dev='hd'/>
        </os>
        <features>
          <acpi/>
          <apic/>
          <vmport state='off'/>
        </features>
        <cpu mode='custom' match='exact' check='full'>
          <model fallback='forbid'>Broadwell</model>
          <feature policy='require' name='vme'/>
          <feature policy='require' name='f16c'/>
          <feature policy='require' name='rdrand'/>
          <feature policy='require' name='hypervisor'/>
          <feature policy='require' name='arat'/>
          <feature policy='disable' name='erms'/>
          <feature policy='require' name='xsaveopt'/>
          <feature policy='require' name='abm'/>
        </cpu>
        <clock offset='utc'>
          <timer name='rtc' tickpolicy='catchup'/>
          <timer name='pit' tickpolicy='delay'/>
          <timer name='hpet' present='no'/>
        </clock>
        <on_poweroff>destroy</on_poweroff>
        <on_reboot>restart</on_reboot>
        <on_crash>destroy</on_crash>
        <pm>
          <suspend-to-mem enabled='no'/>
          <suspend-to-disk enabled='no'/>
        </pm>
        <devices>
          <emulator>/usr/bin/kvm-spice</emulator>
          <disk type='file' device='disk'>
            <driver name='qemu' type='qcow2'/>
            <source file='absolute_path_vnios_qcow2_image'/>
            <backingStore/>
            <target dev='hda' bus='ide'/>
            <alias name='ide0-0-0'/>
                  <address  type='drive'  controller='0'  bus='0'  target='0'
      unit='0'/>
          </disk>
          <controller type='usb' index='0' model='ich9-ehci1'>
            <alias name='usb'/>
                <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
      function='0x7'/>
          </controller>
          <controller type='usb' index='0' model='ich9-uhci1'>
            <alias name='usb'/>
            <master startport='0'/>
```

```
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
      function='0x0' multifunction='on'/>
        </controller>
        <controller type='usb' index='0' model='ich9-uhci2'>
          <alias name='usb'/>
          <master startport='2'/>
            <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
      function='0x1'/>
        </controller>
        <controller type='usb' index='0' model='ich9-uhci3'>
          <alias name='usb'/>
          <master startport='4'/>
            <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x06'
      function='0x2'/>
        </controller>
        <controller type='pci' index='0' model='pci-root'>
          <alias name='pci.0'/>
        </controller>
        <controller type='ide' index='0'>
          <alias name='ide'/>
            <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x01'
      function='0x1'/>
        </controller>
        <controller type='virtio-serial' index='0'>
           <alias name='virtio-serial0'/>
              <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x07'
      function='0x0'/>
         </controller>
         <interface type='direct'>
                    <source  network='LAN1_MACVCTAP'  dev='infoblox-lan1'
      mode='bridge'/>
           <model type='virtio'/>
         </interface>
         <interface type='direct'>
                     <source  network='MGMT_MACVTAP'  dev='infoblox-mgmt'
      mode='bridge'/>
           <model type='virtio'/>
         </interface>
         <serial type='pty'>
           <source path='/dev/pts/2'/>
           <target type='isa-serial' port='0'>
             <model name='isa-serial'/>
           </target>
           <alias name='serial0'/>
         </serial>
         <console type='pty' tty='/dev/pts/2'>
           <source path='/dev/pts/2'/>
           <target type='serial' port='0'/>
           <alias name='serial0'/>
        </console>
        <channel type='spicevmc'>
                      <target   type='virtio'   name='com.redhat.spice.0'
state='disconnected'/>
          <alias name='channel0'/>
          <address type='virtio-serial' controller='0' bus='0' port='1'/>
        </channel>
        <input type='mouse' bus='ps2'>
          <alias name='input0'/>
        </input>
```

```
     <input type='keyboard' bus='ps2'>
       <alias name='input1'/>
     </input>
                <graphics   type='spice'   port='5900'   autoport='yes'
listen='127.0.0.1'>
       <listen type='address' address='127.0.0.1'/>
       <image compression='off'/>
     </graphics>
     <sound model='ich6'>
       <alias name='sound0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x05'
function='0x0'/>
     </sound>
     <video>
           <model  type='qxl'  ram='65536'  vram='65536'  vgamem='16384'
heads='1' primary='yes'/>
       <alias name='video0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x02'
function='0x0'/>
     </video>
     <redirdev bus='usb' type='spicevmc'>
       <alias name='redir0'/>
       <address type='usb' bus='0' port='1'/>
     </redirdev>
     <redirdev bus='usb' type='spicevmc'>
       <alias name='redir1'/>
       <address type='usb' bus='0' port='2'/>
     </redirdev>
     <memballoon model='virtio'>
       <alias name='balloon0'/>
          <address  type='pci'  domain='0x0000'  bus='0x00'  slot='0x08'
function='0x0'/>
     </memballoon>
   </devices>
   <seclabel type='dynamic' model='apparmor' relabel='yes'>
     <label>libvirt-588c6acb-3fcd-49a3-ad84-a59905c55f7d</label>

<imagelabel>libvirt-588c6acb-3fcd-49a3-ad84-a59905c55f7d</imagelabel>
   </seclabel>
   <seclabel type='dynamic' model='dac' relabel='yes'>
     <label>+64055:+130</label>
     <imagelabel>+64055:+130</imagelabel>
   </seclabel>
</domain>
```

3. Deploy vNIOS instance from above mentioned xml file by running **`virsh create vNIOS-macvtap.xml`** command.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh create vNIOS-macvtap.xml
Domain Grid-Master-Macvtap-Interfaces created from vNIOS-macvtap.xml
```

4. Verify that vNIOS instance has been created and is running by running **`virsh list --all`** command.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh list --all
 Id    Name                           State
----------------------------------------------------
 24    Grid-Master-Macvtap-Interfaces running
```

5. To login to the console of vNIOS, use `virsh console` ***instance_id*** command.
   Note: After running the command `virsh console` ***instance_id, hit enter key multiple times to get the console prompt.***

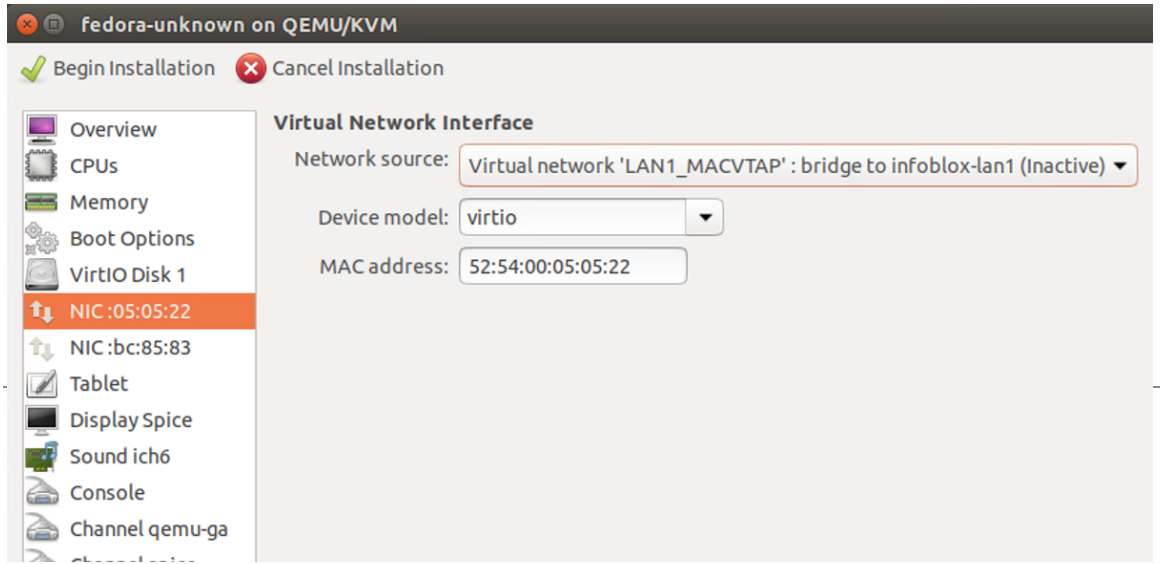
```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh console 24
Connected to domain Grid-Master-Macvtap-Interfaces
Escape character is ^]


Good Bye


Disconnect NOW if you have not been expressly authorized to use this system.
login:
```

6. To exit the console press “ctrl” and “]” key simultaneously.
7. To delete vNIOS instance use **virsh destroy** ***instance_id*** command.

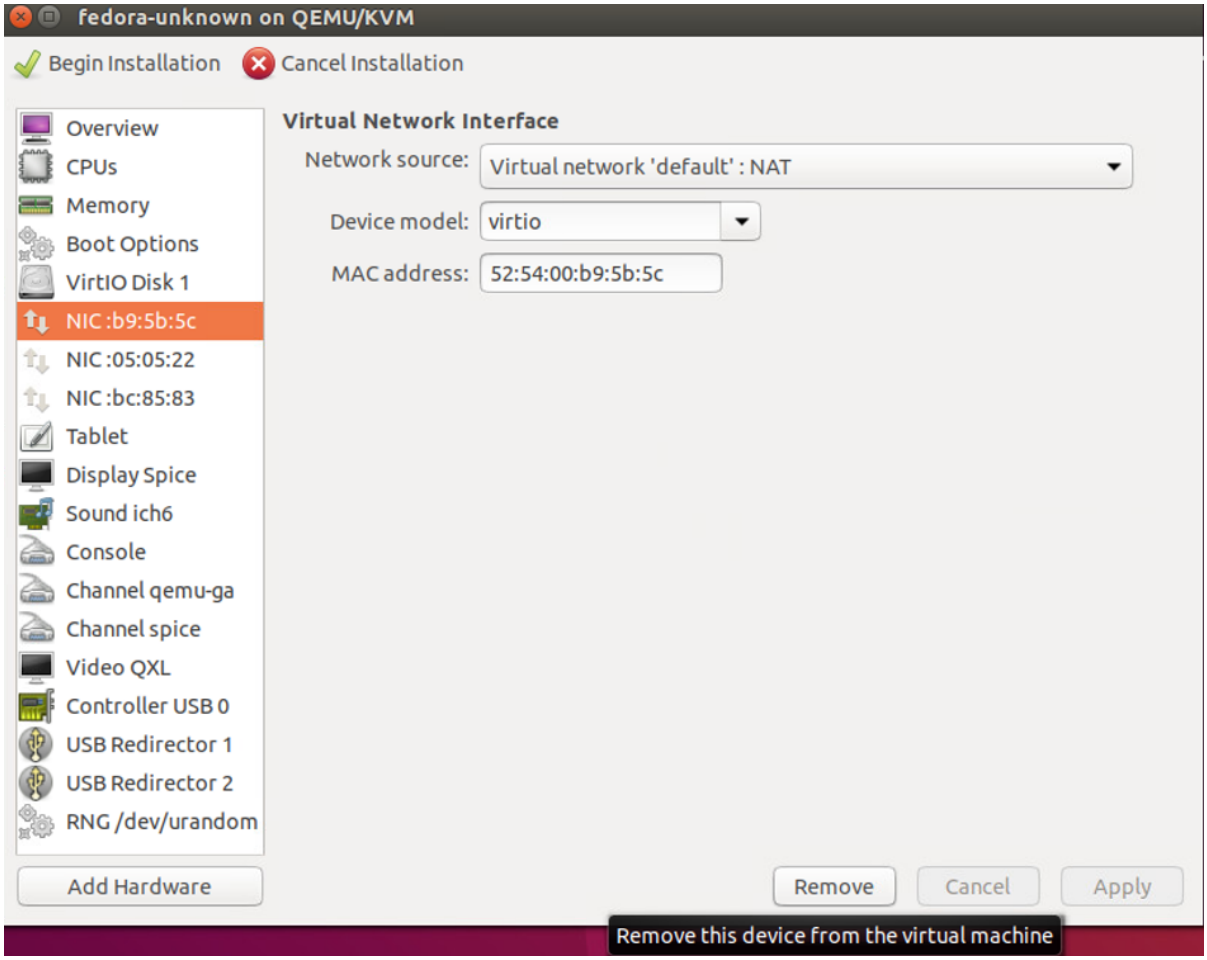
```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh destroy 24
Domain 24 destroyed

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox#
```

Deploying vNIOS instance using virt-manager (GUI based approach) with Macvtap networking

1. Follow the same sequence of steps as mentioned in **Deploying vNIOS instance using virt-manager (GUI based approach) with Bridge Networking.**
2. While selecting network interfaces for the vNIOS, select **LAN1_MACVTAP** and **MGMT_MACVTAP** interfaces respectively one by one with Device mode as virtio.

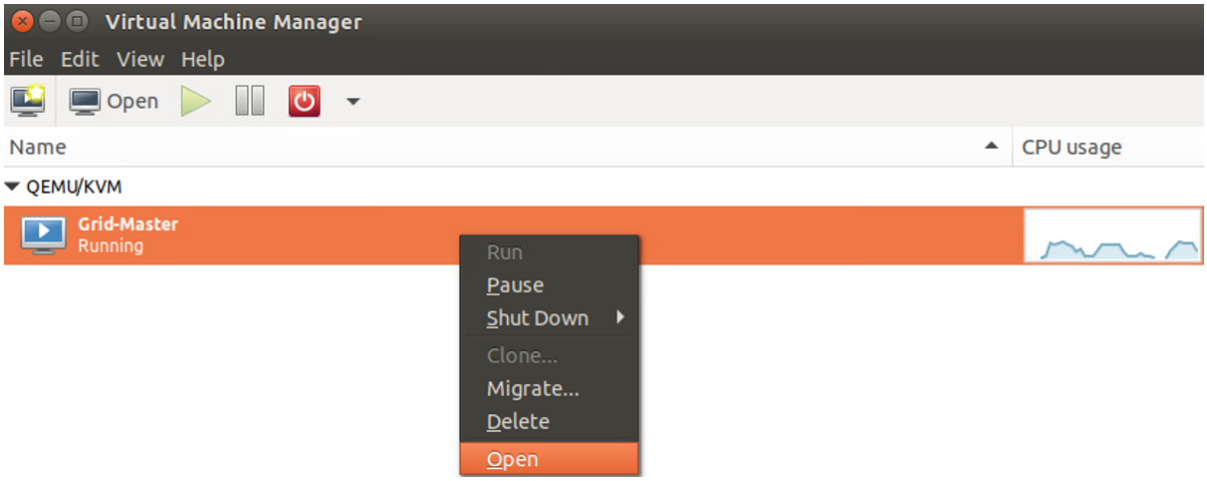


3. After adding Macvtap interfaces, remove the first interfaces which gets added by default.



4. Click on **Begin Installation** to start vNIOS deployment.

5. Once the instance is up and running, right click on the instance and select **Open** to access the console.



Deploying vNIOS instance using virt-install utility with Macvtap Networking

1. Login to the ubuntu host as root user.

2. Run the following virt-install command to deploy vNIOS instance.

```
virt-install --name vnios-instance-name --ram=ram-in-mb
--vcpus=number-of-vcpus --disk
path=absolute-path-of-vnios-qcow2-image,size=250 --boot hd --import
--network network:MGMT_MACVTAP --network network:LAN1_MACVTAP --os-type=linux
```

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virt-install --name Grid-Master --ram=8192 --vcpus=4 --disk path=/var/lib/
libvirt/images/infoblox/nios-8.3.2-376768-2018-11-02-02-41-25-ddi.qcow2,size=250 --boot hd --import --network network:MGMT_MAC
VTAP --network network:LAN1_MACVTAP --os-type=linux --os-variant=rhel7
WARNING  Graphics requested but DISPLAY is not set. Not running virt-viewer.
WARNING  No console to launch for the guest, defaulting to --wait -1

Starting install...
Domain installation still in progress. Waiting for installation to complete.
```

```
--os-variant=rhel7
```

3. Run `virsh list --all` command to verify that the vNIOS is deployed and is in running state.
4. To login to the console of vNIOS, use `virsh console instance_id` command. ***Note: After running the command `virsh console instance_id`, hit enter key multiple times to get the console prompt***.
5. To exit the console press “ctrl” and “]” key simultaneously.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh list --all
 Id    Name                           State
----------------------------------------------------
 27    Grid-Master                    running
```

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh console 27
Connected to domain Grid-Master
Escape character is ^]


Good Bye


Disconnect NOW if you have not been expressly authorized to use this system.
login:
```

6. To delete vNIOS instance use `virsh destroy instance_id` command.

```
root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox# virsh destroy 27
Domain 27 destroyed

root@ubuntu16-kvm:/var/lib/libvirt/images/infoblox#
```

vNIOS deployment on KVM with SR-IOV interfaces

The single root I/O virtualization (SR-IOV) interface is an extension to the PCI Express (PCIe) specification. SR-IOV allows a device, such as a network adapter, to separate access to its resources among various PCIe hardware functions. These functions consist of the following types:

- A PCIe Physical Function (PF). This function is the primary function of the device and advertises the device's SR-IOV capabilities.

- One or more PCIe Virtual Functions (VFs). Each VF is associated with the device's PF. A VF shares one or more physical resources of the device, such as a memory and a network port, with the PF and other VFs on the device. Each VF is associated with a KVM instance in a virtualized environment.

Each PF and VF is assigned a unique PCI Express Requester ID (RID) that allows an I/O memory management unit (IOMMU) to differentiate between different traffic streams and apply memory and interrupt translations between the PF and VFs. This allows traffic streams to be delivered directly to the appropriate KVM instance. As a result, nonprivileged data traffic flows from the PF to VF without affecting other VFs.

SR-IOV enables network traffic to bypass the software switch layer of the KVM virtualization stack. Because the VF is assigned to an instance, the network traffic flows directly between the VF and instance. As a result, the I/O overhead in the software emulation layer is diminished and achieves network performance that is nearly the same performance as in nonvirtualized environments.

Enabling SR-IOV virtual functions in KVM

Enabling SR-IOV virtual function on KVM

1. Login to the ubuntu KVM host as a root user.
2. Run **`ip a`** command to get the list of interfaces and make a note of SR-IOV physical ports.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq master virbr0 state UP group default qlen 1000
    link/ether 18:66:da:fb:7c:c4 brd ff:ff:ff:ff:ff:ff
3: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 18:66:da:fb:7c:c5 brd ff:ff:ff:ff:ff:ff
4: eno3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 18:66:da:fb:7c:c6 brd ff:ff:ff:ff:ff:ff
5: eno4: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether 18:66:da:fb:7c:c7 brd ff:ff:ff:ff:ff:ff
6: enp6s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether a0:36:9f:b9:ff:1c brd ff:ff:ff:ff:ff:ff
    inet 10.196.206.233/24 brd 10.196.206.255 scope global enp6s0f0
       valid_lft forever preferred_lft forever
    inet6 fe80::a236:9fff:feb9:ff1c/64 scope link
       valid_lft forever preferred_lft forever
7: enp6s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether a0:36:9f:b9:ff:1e brd ff:ff:ff:ff:ff:ff
    inet 10.196.205.100/24 brd 10.196.205.255 scope global enp6s0f1
       valid_lft forever preferred_lft forever
    inet6 fe80::a236:9fff:feb9:ff1e/64 scope link
       valid_lft forever preferred_lft forever
```

Note: SR-IOV interfaces always have a long name.

3. Navigate to **`/sys/class/net/name`** directory. List the contents of the directory by running **`ls`** command. You will find directories corresponding to the interfaces. Change directory to one of the SR-IOV interface.

```
root@ubuntu16-kvm:~# cd /sys/class/net/
root@ubuntu16-kvm:/sys/class/net# ls
eno1  eno2  eno3  eno4  enp6s0f0  enp6s0f1  lo  virbr0  virbr0-nic
root@ubuntu16-kvm:/sys/class/net#
```

4. Edit the sriov_numfs file, which is present in device directory, using a vi editor **vi /device/sriov_numfs**

```
root@ubuntu16-kvm:/sys/class/net# cd enp6s0f0
root@ubuntu16-kvm:/sys/class/net/enp6s0f0# ls
addr_assign_type  carrier_changes     dev_port          ifalias    name_assign_type  phys_switch_id  statistics
address           carrier_down_count  dormant           ifindex    netdev_group      power           subsystem
addr_len          carrier_up_count    duplex            iflink     operstate         proto_down      tx_queue_len
broadcast         device              flags             link_mode  phys_port_id      queues          type
carrier           dev_id              gro_flush_timeout mtu        phys_port_name    speed           uevent
root@ubuntu16-kvm:/sys/class/net/enp6s0f0#
```

```
root@ubuntu16-kvm:/sys/class/net/enp6s0f0/device# vi sriov_numvfs
```

5. Enter the number of **vfs**, you would like to activate. Save and close the file.

```
4
~
~
```

Repeat the same steps for second SR-IOV interface and populate the number of **vfs** you would like to activate.

6. To validate that SR-IOV **vfs** are active and available, run **lspci |grep -i ethernet** command. You should see **8 vfs.**

```
root@ubuntu16-kvm:/sys/class/net/enp6s0f1/device# lspci |grep -i ethernet
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
01:00.1 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
02:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
02:00.1 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
06:00.0 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)
06:00.1 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)
06:10.0 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.1 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.2 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.3 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.4 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.5 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.6 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.7 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
```

SR-IOV vfs

7. Run **ip link show** command to verify that **vfs** are showing up under corresponding SR-IOV physical ports.

8. To create persistent vfs use the following command.

```
echo "echo '7' > /sys/class/net/sriov_interface_name/device/sriov_numvfs" >>
/etc/rc.local
```

```
root@ubuntu16-kvm:/sys/class/net/enp6s0f1/device# ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qle
    link/ether 18:66:da:fb:7c:c4 brd ff:ff:ff:ff:ff:ff
3: eno2: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT group default
    link/ether 18:66:da:fb:7c:c5 brd ff:ff:ff:ff:ff:ff
4: eno3: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT group default
    link/ether 18:66:da:fb:7c:c6 brd ff:ff:ff:ff:ff:ff
5: eno4: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN mode DEFAULT group default
    link/ether 18:66:da:fb:7c:c7 brd ff:ff:ff:ff:ff:ff
6: enp6s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
    link/ether a0:36:9f:b9:ff:1c brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 2a:28:6c:78:7d:a8, spoof checking on, link-state auto, trust off
    vf 1 MAC b6:98:08:27:a4:8f, spoof checking on, link-state auto, trust off
    vf 2 MAC 62:0a:4a:53:c0:29, spoof checking on, link-state auto, trust off
    vf 3 MAC 1e:d6:7f:e6:58:09, spoof checking on, link-state auto, trust off
7: enp6s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default
    link/ether a0:36:9f:b9:ff:1e brd ff:ff:ff:ff:ff:ff
    vf 0 MAC 2a:fd:04:1b:33:df, spoof checking on, link-state auto, trust off
    vf 1 MAC d6:c0:6f:c3:7c:4a, spoof checking on, link-state auto, trust off
    vf 2 MAC 72:72:a3:c2:8d:eb, spoof checking on, link-state auto, trust off
    vf 3 MAC 7a:51:4c:a1:b3:7b, spoof checking on, link-state auto, trust off
```

Deploying vNIOS with SR-IOV vfs

1. Run `lspci| grep -i ethernet` command and make a note of pcie identifier of the virtual functions.

```
root@ubuntu16-kvm:/sys/class/net/enp6s0f1/device# lspci |grep -i ethernet
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
01:00.1 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
02:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
02:00.1 Ethernet controller: Broadcom Corporation NetXtreme BCM5720 Gigabit Ethernet PCIe
06:00.0 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)
06:00.1 Ethernet controller: Intel Corporation Ethernet Controller 10-Gigabit X540-AT2 (rev 01)
06:10.0 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.1 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.2 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.3 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.4 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.5 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.6 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
06:10.7 Ethernet controller: Intel Corporation X540 Ethernet Controller Virtual Function (rev 01)
```

pcie identifier

Note: pcie identifiers are numbered alternatively for each of the SR-IOV physical port. In the above screen shot 6:10.0, 6:10.2... correspond to first physical SR-IOV port and 6:10.1,6:10.3... correspond to second physical SR-IOV port.

2. Run `virsh nodedev-list --cap pci` command to get the complete pcie identifier of the virtual function and make a note of it. Use the pci identifiers obtained from the previous command to get full pcie identifier.

```
pci_0000_06_10_0
pci_0000_06_10_1
pci_0000_06_10_2
pci_0000_06_10_3
pci_0000_06_10_4
pci_0000_06_10_5
pci_0000_06_10_6
pci_0000_06_10_7
```

3. Use the following command to deploy vnios with SR-IOV interfaces. We are taking one **vf** from each physical SR-IOV port.

```
virt-install --name=vnios_instance_name --disk
path=absolute_path_of_vnios_image, size=250 --vcpu=number_of_vcpus
--ram=ram_in_mb --boot hd --os-type=linux --os-variant=rhel7 --nonetworks
--host-device=pci_0000_06_10_0 --host-device=pci_0000_06_10_1
```

```
root@ubuntu16-kvm:~# virt-install --name=Grid-Master-SR-IOV --disk path=/var/lib/lib
virt/images/nios-8.3.2-376768-2018-11-02-02-41-25-ddi.qcow2,size=259 --vcpu=4 --ram=
12258 --boot hd --os-type=linux --os-variant=rhel7 --nonetwork --host-device=pci_000
0_06_10_0 --host-device=pci_0000_06_10_1
WARNING  Graphics requested but DISPLAY is not set. Not running virt-viewer.
WARNING  No console to launch for the guest, defaulting to --wait -1

Starting install...
Creating domain...                                          |    0 B  00:00:03
Domain installation still in progress. Waiting for installation to complete.
```

4. Validate that vNIOS is in running state by running **virsh list --all** command

```
root@ubuntu16-kvm:~# virsh list --all
 Id    Name                           State
----------------------------------------------------
 10    Grid-Master-SR-IOV             running
```

5. Login to the console of the vNIOS by running **virsh console** ***instance-id***

```
root@ubuntu16-kvm:~# virsh console 10
Connected to domain Grid-Master-SR-IOV
Escape character is ^]



Disconnect NOW if you have not been expressly authorized to use this
login:
```

6. Once logged in to the console run **`show interface all`** command. Make a note of the MAC

```
Infoblox > show interface all
LAN1:
          IP Address:  10.196.206.100   MAC Address: 26:97:48:91:8B:8F
          Mask:        255.255.255.0    Broadcast:   10.196.206.255
          MTU:         1500             Metric:      1
          IPv6 Link:          fe80::2497:48ff:fe91:8b8f/64
          IPv6 Status:        Enabled
          Negotiation: Disabled
          Speed:       1000M            Duplex:      Full
          Status:      UP BROADCAST RUNNING MULTICAST

          Statistics Information
            Received
                  packets:  1796         bytes:    360583 (352.1 KiB)
                  errors:   0            dropped:  0
                  overruns: 0            frame:    0
            Transmitted
                  packets:  48           bytes:    2016 (1.9 KiB)
                  errors:   0            dropped:  0
                  overruns: 0            carrier:  0
            Collisions: 0                Txqueuelen:  10000
MGMT:
          IP Address:  10.196.201.100   MAC Address: 66:B1:89:F1:3B:A0
          Mask:        255.255.255.0    Broadcast:   10.196.201.255
          MTU:         1500             Metric:      1
          IPv6 Link:          fe80::64b1:89ff:fef1:3ba0/64
          IPv6 Status:        Enabled
```

address of the **`Lan1`** and **`Mgmt`** interface.

7. Exit out from the console by pressing “ctrl” and “]” key simultaneously.

8. Run **`ip link show`** command and verify that MAC obtained from the previous step matches with the MAC address of **`vfs.`** We have used one **`vf`** from each SR-IOV physical port.

```
enp6s0f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
 link/ether a0:36:9f:b9:ff:1c brd ff:ff:ff:ff:ff:ff
 vf 0 MAC 66:b1:89:f1:3b:a0, spoof checking on, link-state auto, trust off
 vf 1 MAC 56:3a:02:9f:2e:1e, spoof checking on, link-state auto, trust off
 vf 2 MAC b6:e6:4f:41:63:48, spoof checking on, link-state auto, trust off
 vf 3 MAC 6a:b9:7b:a3:f7:82, spoof checking on, link-state auto, trust off
 vf 4 MAC 86:a8:a7:c5:e9:db, spoof checking on, link-state auto, trust off
 vf 5 MAC 4e:00:21:58:fe:3e, spoof checking on, link-state auto, trust off
 vf 6 MAC ea:3b:ce:a0:f9:a4, spoof checking on, link-state auto, trust off
 vf 7 MAC 22:53:f3:bb:70:40, spoof checking on, link-state auto, trust off
enp6s0f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT group default qlen 1000
 link/ether a0:36:9f:b9:ff:1e brd ff:ff:ff:ff:ff:ff
 vf 0 MAC 26:97:48:91:8b:8f, spoof checking on, link-state auto, trust off
 vf 1 MAC 56:29:e2:91:b4:63, spoof checking on, link-state auto, trust off
 vf 2 MAC 72:72:a3:c2:8d:eb, spoof checking on, link-state auto, trust off
 vf 3 MAC 7a:51:4c:a1:b3:7b, spoof checking on, link-state auto, trust off
 vf 4 MAC e2:62:c4:e9:16:80, spoof checking on, link-state auto, trust off
 vf 5 MAC 2e:fc:53:45:33:1e, spoof checking on, link-state auto, trust off
 vf 6 MAC f6:6f:6b:68:32:6e, spoof checking on, link-state auto, trust off
 vf 7 MAC ee:b0:72:6f:2f:31, spoof checking on, link-state auto, trust off
```

9. To delete the vnios instance execute **`virsh destroy`** ***`instance-id`*** command.

Some Useful information

1. In case, when an active console session exists for a domain, you will not be able to start a fresh console session, instead you will get the following prompt

```
root@ubuntu16-kvm:~# virsh console 28
Connected to domain fedora-unknown
Escape character is ^]
error: operation failed: Active console session exists for this domain
```

   Use **`service libvirtd restart`** command to kill all the active console sessions.

```
root@ubuntu16-kvm:~# service libvirtd restart
root@ubuntu16-kvm:~#
```

2. Following SR-IOV interfaces are tested and validated by Infoblox on KVM
   - Intel 82599 (10G)
   - Intel 10GX540

3. Following non-SR-IOV interfaces are tested and validated by Infoblox on KVM.
   - Intel I350
   - Broadcom BCM5719

4. For deploying Infoblox Reporting appliance use the following command
   **`virt-install --name=Infoblox-Reporting --ram=`*`ram_in_mb`*
   `--vcpus=`*`number_of_vcpus`* **`--disk`
   `path=`*`absolute_path_of_reporting_disk1_qcow2_image`*`,size=250 --disk`**
   `path=`*`absolute_path_of_reporting_disk2_qcow2_image`*`,size=250 --boot`**
   `hd --import --network network:Infoblox-Reporting-Mgmt --network`
   `network:Infoblox-Reporting-Lan1 --os-type=linux --os-variant=rhel7`

5. For troubleshooting purpose, instance logs can be viewed from /**`var/log/libvirt/qemu`** directory.

Infoblox is the leader in modern, cloud-first networking and security services. Through extensive integrations, its solutions empower organizations to realize the full advantages of cloud networking today, while maximizing their existing infrastructure investments. Infoblox has over 12,000 customers, including 70 percent of the Fortune 500.

Corporate Headquarters | 2390 Mission College Boulevard, Ste. 501 | Santa Clara, CA | 95054
+1.408.986.4000 | info@infoblox.com | www.infoblox.com