Academic Year: 2022-23	**Name of Student:**
Semester: VIII	**Student ID:**
Class / Branch: BE- IT	**Roll No.:**
Subject: Cloud Computing Lab	**Date of Submission:**
Name of Instructor: Yaminee Patil	

Experiment No.:1.

Aim: To study and implement Hosted Virtualization using Virtual Box & KVM

Software Used: Virtual Box.

Theory:

Kernel-based Virtual Machine (KVM) is an open source virtualization technology built into Linux. Specifically, KVM lets you turn Linux into a hypervisor that allows a host machine to run multiple, isolated virtual environments called guests or virtual machines (VMs).

KVM is part of Linux. If you've got Linux 2.6.20 or newer, you've got KVM. KVM was first announced in 2006 and merged into the mainline Linux kernel version a year later. Because KVM is part of existing Linux code, it immediately benefits from every new Linux feature, fix, and advancement without additional engineering.

How does KVM work?

KVM converts Linux into a type-1 (bare-metal) hypervisor. All hypervisors need some operating system-level components—such as a memory manager, process scheduler, input/output (I/O) stack, device drivers, security manager, a network stack, and more—to run VMs. KVM has all these components because it's part of the Linux kernel. Every VM is implemented as a regular Linux process, scheduled by the standard Linux scheduler, with dedicated virtual hardware like anetwork card, graphics adapter, CPU(s), memory, and disks.

KVM features

KVM is part of Linux. Linux is part of KVM. Everything Linux has, KVM has too. But there are specific features that make KVM an enterprise's preferred hypervisor.

Security

KVM uses a combination of security-enhanced Linux (SELinux) and secure virtualization (sVirt) for enhanced VM security and isolation. SELinux establishes security boundaries around VMs. sVirt extends SELinux's capabilities, allowing

Mandatory Access Control (MAC) security to be applied to guest VMs and preventing manual labeling errors.

Storage

KVM is able to use any storage supported by Linux, including some local disks and network- attached storage (NAS). Multipath I/O may be used to improve storage and provide redundancy. KVM also supports shared file systems so VM images may be shared by multiple hosts. Disk

images support thin provisioning, allocating storage on demand rather than all up front.

Hardware support

KVM can use a wide variety of certified Linux-supported hardware platforms. Because hardwarevendors regularly contribute to kernel development, the latest hardware features are often rapidlyadopted in the Linux kernel.

Memory management

KVM inherits the memory management features of Linux, including non-uniform memory accessand kernel same-page merging. The memory of a VM can be swapped, backed by large volumes for better performance, and shared or backed by a disk file.

Live migration

KVM supports live migration, which is the ability to move a running VM between physical hosts with no service interruption. The VM remains powered on, network connections remain active, and applications continue to run while the VM is relocated. KVM also saves a VM's current stateso it can be stored and resumed later.

Performance and scalability

KVM inherits the performance of Linux, scaling to match demand load if the number of guest machines and requests increases. KVM allows the most demanding application workloads to be virtualized and is the basis for many enterprise virtualization setups, such as datacenters and private clouds (via OpenStack).

Scheduling and resource control

In the KVM model, a VM is a Linux process, scheduled and managed by the kernel. The Linux scheduler allows fine-grained control of the resources allocated to a Linux process and guaranteesa quality of service for a particular process. In KVM, this includes the completely fair scheduler, control groups, network name spaces, and real-time extensions.

Lower latency and higher prioritization

The Linux kernel features real-time extensions that allow VM-based apps to run at lower latency with better prioritization (compared to bare metal). The kernel also divides processes that require long computing times into smaller components, which are then scheduled and processed accordingly.

Installation of KVM

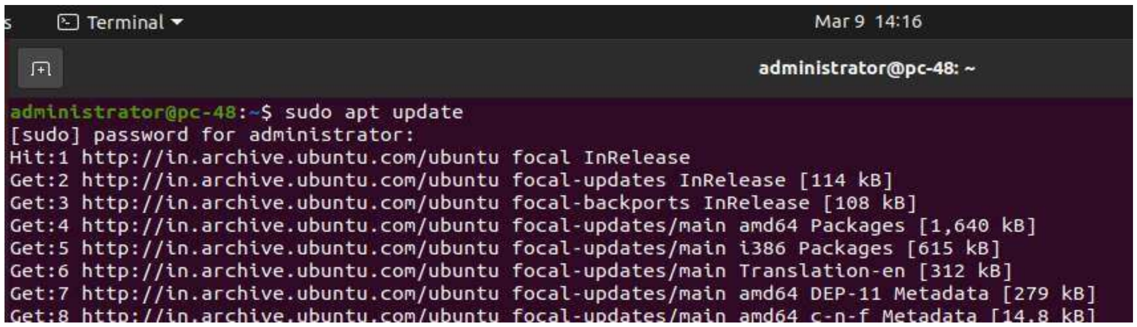
To install cpu-checker, run the following command:

sudo apt install cpu-checker

Step 1: Install KVM Packages

1. First, update the repositories:

sudo apt update

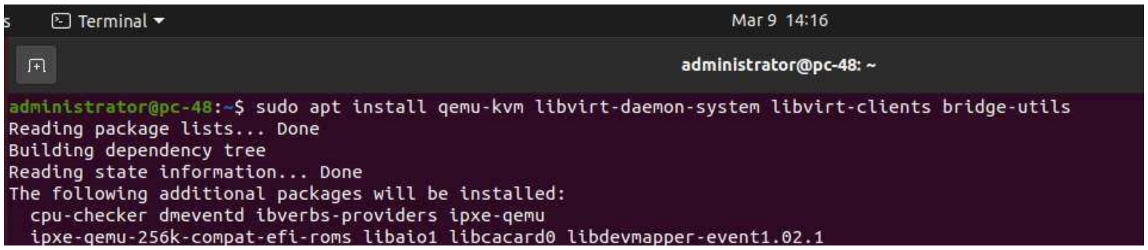


```
administrator@pc-48:~$ sudo apt update
[sudo] password for administrator:
Hit:1 http://in.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1,640 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [615 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [312 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metadata [279 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 c-n-f Metadata [14.8 kB]
```

2. Then, install essential KVM packages with the following command:

sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils

This will start the installation of four KVM packages:



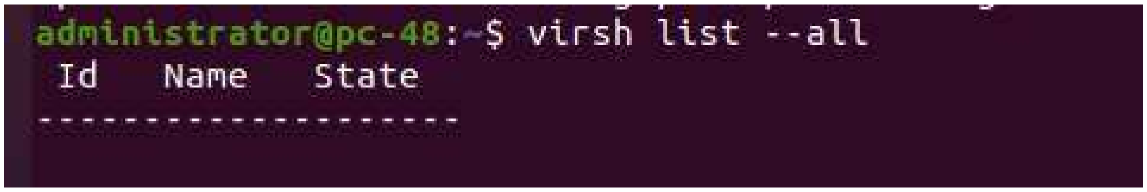
```
administrator@pc-48:~$ sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cpu-checker dmeventd ibverbs-providers ipxe-qemu
  ipxe-qemu-256k-compat-efi-roms libaio1 libcacard0 libdevmapper-event1.02.1
```

Step 3: Verify the Installation

1. Confirm the installation was successful by using the virsh command:

virsh list --all

You can expect an output as seen below:

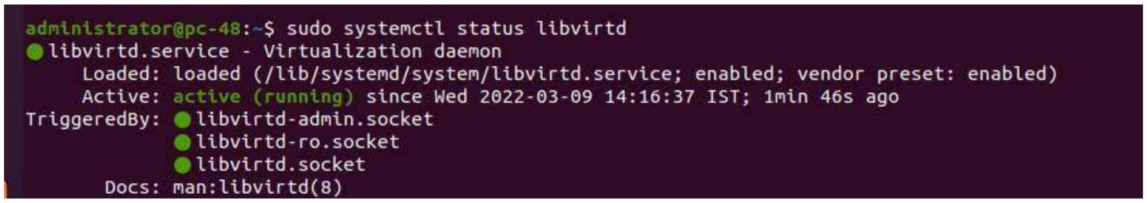


```
administrator@pc-48:~$ virsh list --all
 Id   Name   State
--------------------
```

2. Or use the systemctl command to check the status of libvirtd:

sudo systemctl status libvirtd

If everything is functioning properly, the output returns an active (running) status.



```
administrator@pc-48:~$ sudo systemctl status libvirtd
● libvirtd.service - Virtualization daemon
     Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-03-09 14:16:37 IST; 1min 46s ago
TriggeredBy: ● libvirtd-admin.socket
             ● libvirtd-ro.socket
             ● libvirtd.socket
       Docs: man:libvirtd(8)
```

3. Press Q to quit the status screen.

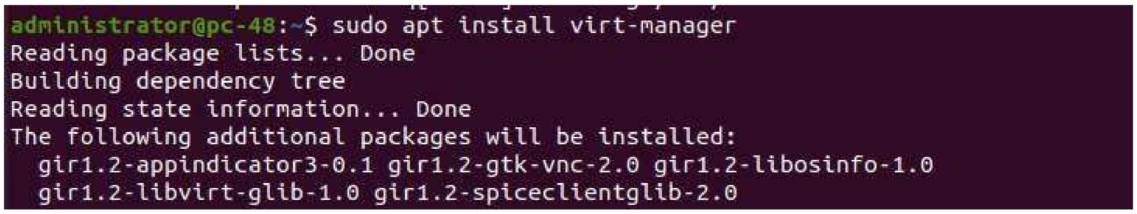
4. If the virtualization daemon is not active, activate it with the following command:

sudo systemctl enable --now libvirtd

Creating a Virtual Machine on Ubuntu 20.04

1. Before you choose one of the two methods listed below, install virt-manager, a tool forcreating and managing VMs:

sudo apt install virt-manager



```
administrator@pc-48:~$ sudo apt install virt-manager
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  gir1.2-appindicator3-0.1 gir1.2-gtk-vnc-2.0 gir1.2-libosinfo-1.0
  gir1.2-libvirt-glib-1.0 gir1.2-spiceclientglib-2.0
```

2. Type Y and press **ENTER**. Wait for the installation tofinish.

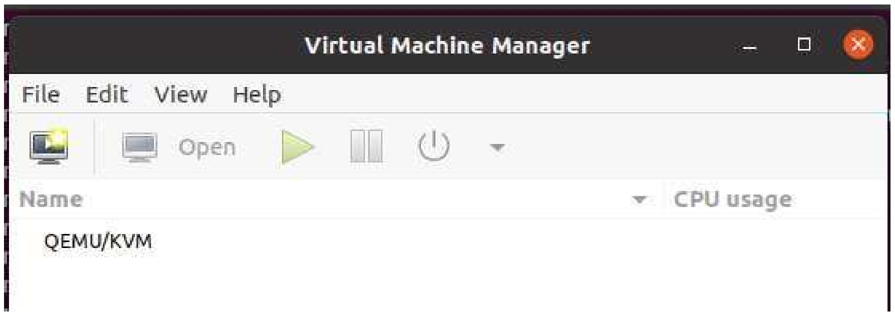
Make sure you download an ISO containing the OS you wish to install on a VM and proceed topick an installation method.

Method 1: Virt Manager GUI

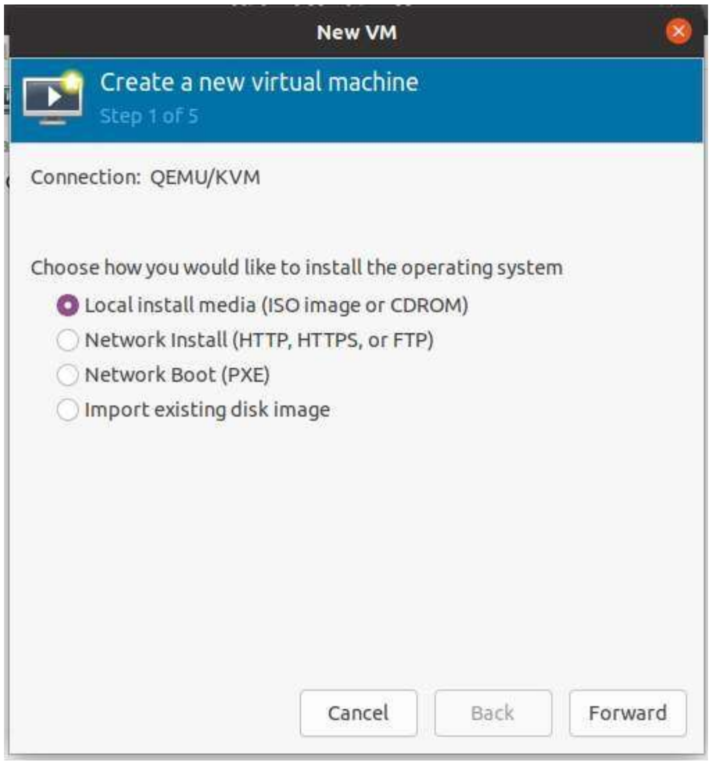
1. Start virt-manager with:

sudo virt-manager

2. In the first window, click the computer icon in the upper-left corner.



3. In the dialogue box that opens, select the option to install the VM using an ISO image. Thenclick **Forward**.

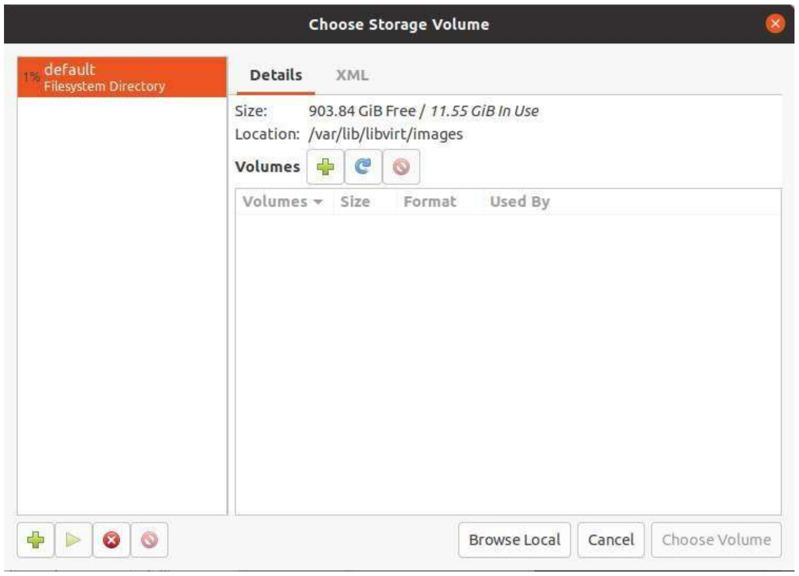


Connection: QEMU/KVM

Choose how you would like to install the operating system

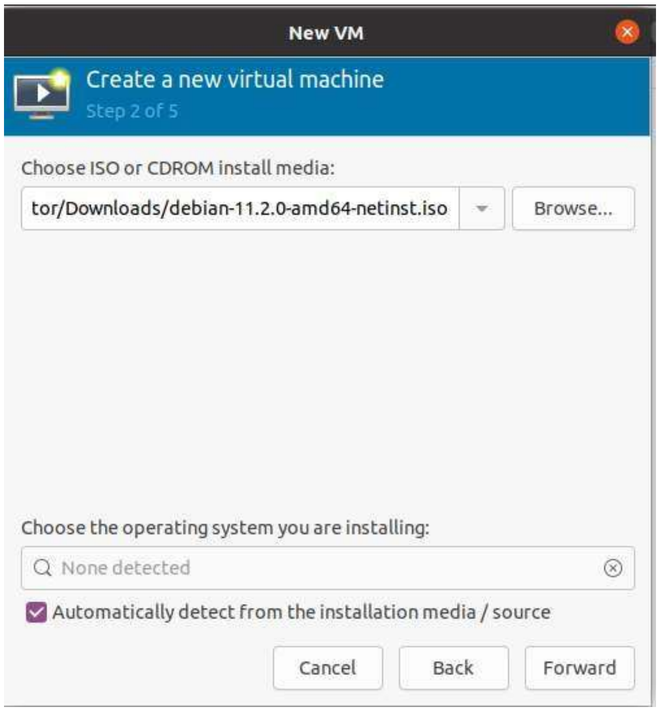
- [✓ Local install media (ISO image or CDROM)]
- Network Install (HTTP, HTTPS, or FTP)
- Network Boot (PXE)
- Import existing disk image

4. In the next dialogue, click **Browse Local** and navigate to the path where you stored the ISOyou wish to install.



Size: 903.84 GiB Free / 11.55 GiB In Use
Location: /var/lib/libvirt/images

5. The ISO you chose in the previous window populates the field in Step 2. Proceed to Step 3 byclicking **Forward**.



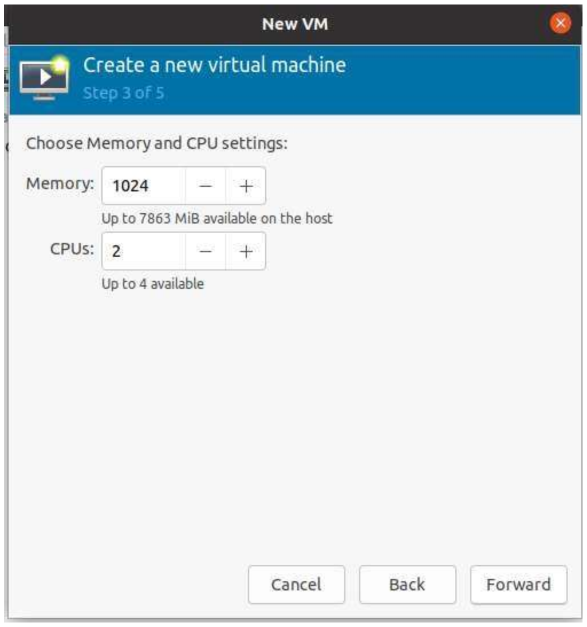
Choose ISO or CDROM install media: tor/Downloads/debian-11.2.0-amd64-netinst.iso

Choose the operating system you are installing: None detected

☑ Automatically detect from the installation media / source

6. Enter the amount of RAM and the number of CPUs you wish to allocate to the

VM andproceed to the next step.



Choose Memory and CPU settings:

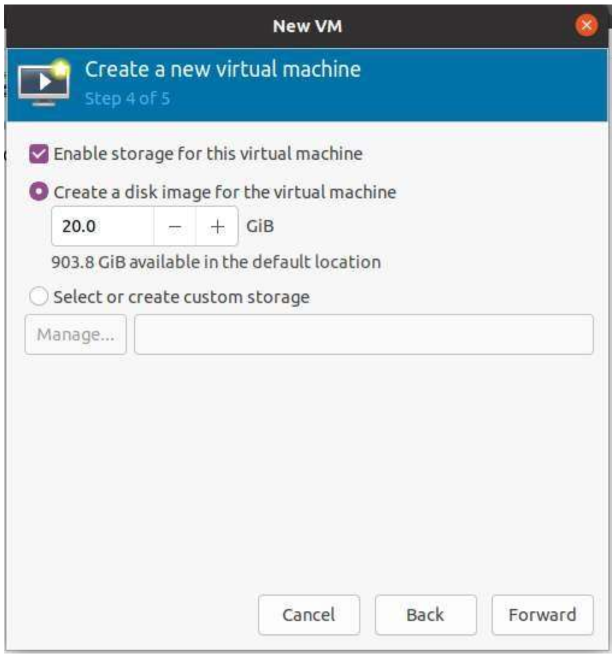
Memory: 1024

Up to 7863 MiB available on the host

CPUs: 2

Up to 4 available

7. Allocate hard disk space to the VM. Click **Forward** to go to the last step.



[✓ Enable storage for this virtual machine]

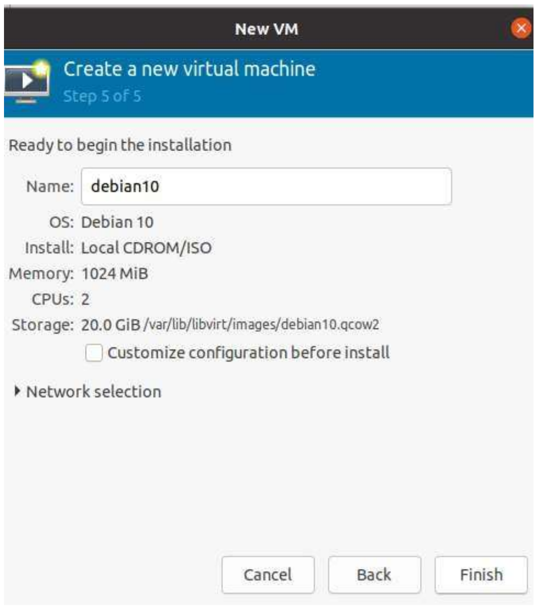
[✓ Create a disk image for the virtual machine]

20.0 GiB

903.8 GiB available in the default location

Select or create custom storage

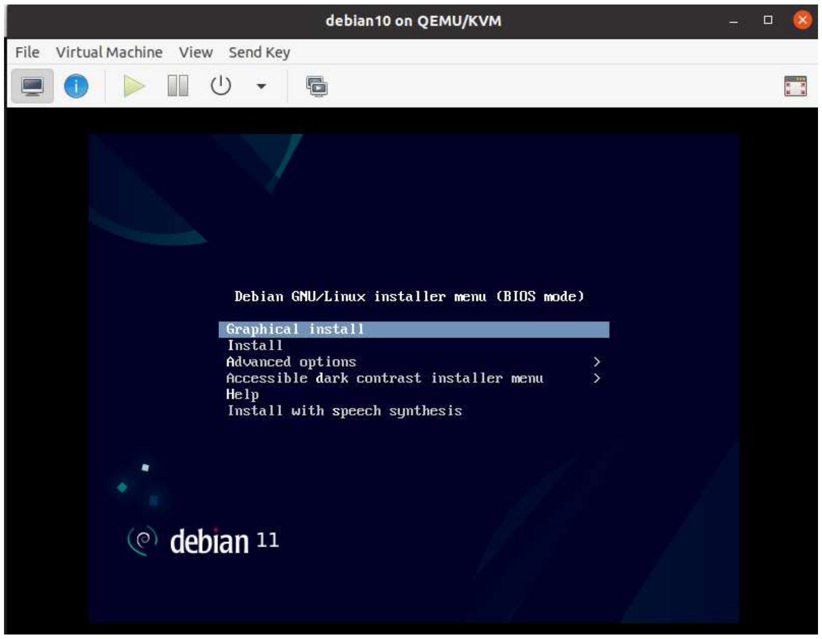
8. Specify the name for your VM and click **Finish** to complete the setup.



Ready to begin the installation

Name: debian10
OS: Debian 10
Install: Local CDROM/ISO
Memory: 1024 MiB
CPUs: 2
Storage: 20.0 GiB /var/lib/libvirt/images/debian10.qcow2
☐ Customize configuration before install

9. The VM starts automatically, prompting you to start installing the OS that's on the ISO file.



```
Debian GNU/Linux installer menu (BIOS mode)

Graphical install
Install
Advanced options                                >
Accessible dark contrast installer menu         >
Help
Install with speech synthesis
```

Conclusion: Thus, we have successfully implemented Hosted Virtualization using KVM