Ex No: 5a **Installation of KVM and creation of Virtual instances**

28/07/26

Aim:

To install KVM and create virtual instances for setting up a virtualized environment on a Linux system.

Theory:

What is KVM?

- **KVM (Kernel-based Virtual Machine)** is a virtualization module in the Linux kernel.
- It enables Linux to function as a **hypervisor**, allowing multiple **virtual machines (VMs)** to run simultaneously on a single physical host.
- KVM uses **Intel VT-x** or **AMD-V** for hardware acceleration.

Components Involved:

1. **KVM Module**: Kernel-level module providing virtualization support.
2. **QEMU**: A machine emulator that handles I/O virtualization.
3. **libvirt**: A management layer to control and automate KVM.
4. **Virt-Manager**: A graphical user interface to manage VMs easily.

Working Principle of KVM:

- KVM converts the Linux kernel into a type-1 (bare-metal) hypervisor.
- Each virtual machine is treated as a **regular Linux process**, with dedicated virtualized CPU, RAM, storage, and network.
- **QEMU** works with KVM to emulate peripheral devices.
- **libvirt** provides APIs and tools to manage these VMs.
- **Virt-Manager** offers a GUI frontend for user-friendly VM management.

Benefits of KVM:

- Open-source and part of the Linux kernel.
- Supports **full virtualization**.
- Efficient use of hardware resources.
- Supports snapshots, live migration, and advanced networking.

Virtualization in KVM:

- Uses **hardware virtualization extensions** (VT-x/AMD-V) for performance.
- If hardware support is absent, QEMU can emulate virtualization in software mode (slower).

Procedure: **22IT011**

Virtualization Technology

1. Check Hardware Virtualization Support.

```
tce@tce-VirtualBox:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
0
tce@tce-VirtualBox:~$
```

   Output ≥ 1 indicates support. If 0, CPU doesn't support hardware virtualization.

2. Check 64-bit CPU and machine hardware architecture of your system.

```
tce@tce-VirtualBox:~$ egrep -c 'lm' /proc/cpuinfo
1
tce@tce-VirtualBox:~$ uname -m
x86_64
tce@tce-VirtualBox:~$
```

3. Run *uname -a* , which shows kernel name, hostname, kernel version, build date, architecture, and OS type in one line.

```
tce@tce-VirtualBox:~$ uname -a
Linux tce-VirtualBox 5.4.0-150-generic #167~18.04.1-Ubuntu SMP Wed May 24 00:51
:42 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
tce@tce-VirtualBox:~$
```

4. To list the KVM kernel module files present in your kernel, run:

```
tce@tce-VirtualBox:~$ ls /lib/modules/5.4.0-84-generic/kernel/arch/x86/kvm/kvm*

/lib/modules/5.4.0-84-generic/kernel/arch/x86/kvm/kvm-amd.ko
/lib/modules/5.4.0-84-generic/kernel/arch/x86/kvm/kvm-intel.ko
/lib/modules/5.4.0-84-generic/kernel/arch/x86/kvm/kvm.ko
tce@tce-VirtualBox:~$
```

KVM Installation

1. Install required packages : *qemu-kvm, libvirt-bin, bridge-utils, virt-manager and qemu-system.*

```
tce@tce-VirtualBox:~$ sudo apt update
Hit:1 https://dl.google.com/linux/chrome/deb stable InRelease
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [102 kB]
```

```
tce@tce-VirtualBox:~$ sudo apt install qemu-kvm libvirt-daemon-system libvirt-c
lients bridge-utils virt-manager qemu-system
Reading package lists... Done
Building dependency tree
Reading state information... Done
bridge-utils is already the newest version (1.5-15ubuntu1).
bridge-utils set to manually installed.
```

```
tce@tce-VirtualBox:~$ sudo apt install qemu-kvm libvirt-bin bridge-utils virt-m
anager qemu-system
[sudo] password for tce:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

2. Configure the libvirt daemon configuration file for remote access & socket permissions in KVM/libvirt.

```
tce@tce-VirtualBox:~$ sudo nano /etc/libvirt/libvirtd.conf
tce@tce-VirtualBox:~$
```

```
tce@tce-VirtualBox:~$ grep ^[^#] /etc/libvirt/libvirtd.conf
listen_addr = "0.0.0.0"
unix_sock_group = "libvirt"
unix_sock_ro_perms = "0777"
unix_sock_rw_perms = "0777"
unix_sock_dir = "/var/run/libvirt"
auth_unix_ro = "none"
auth_unix_rw = "none"
tce@tce-VirtualBox:~$
```

After configuring it, restart libvirtd dervice.

```
tce@tce-VirtualBox:~$ sudo systemctl restart libvirtd
tce@tce-VirtualBox:~$
```

3. Run *virsh list* - Displays no of Virtual machine running.

```
tce@tce-VirtualBox:~$ virsh list
 Id    Name                           State
----------------------------------------------------
```

4. Connecting to Virtualization interactive terminal.

```
tce@tce-VirtualBox:~$ virsh
Welcome to virsh, the virtualization interactive terminal.

Type:  'help' for help with commands
       'quit' to quit

virsh #
```

5. Getting Version of libvirt & QEMU.

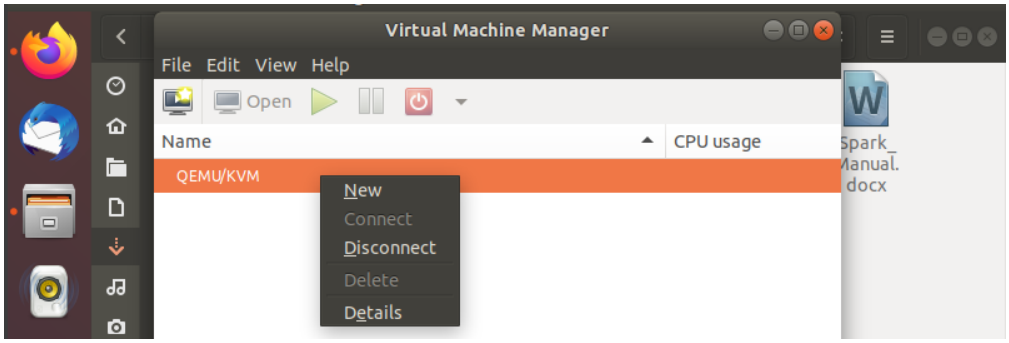
```
virsh # version
Compiled against library: libvirt 4.0.0
Using library: libvirt 4.0.0
Using API: QEMU 4.0.0
Running hypervisor: QEMU 2.11.1
```

6. Getting Node information

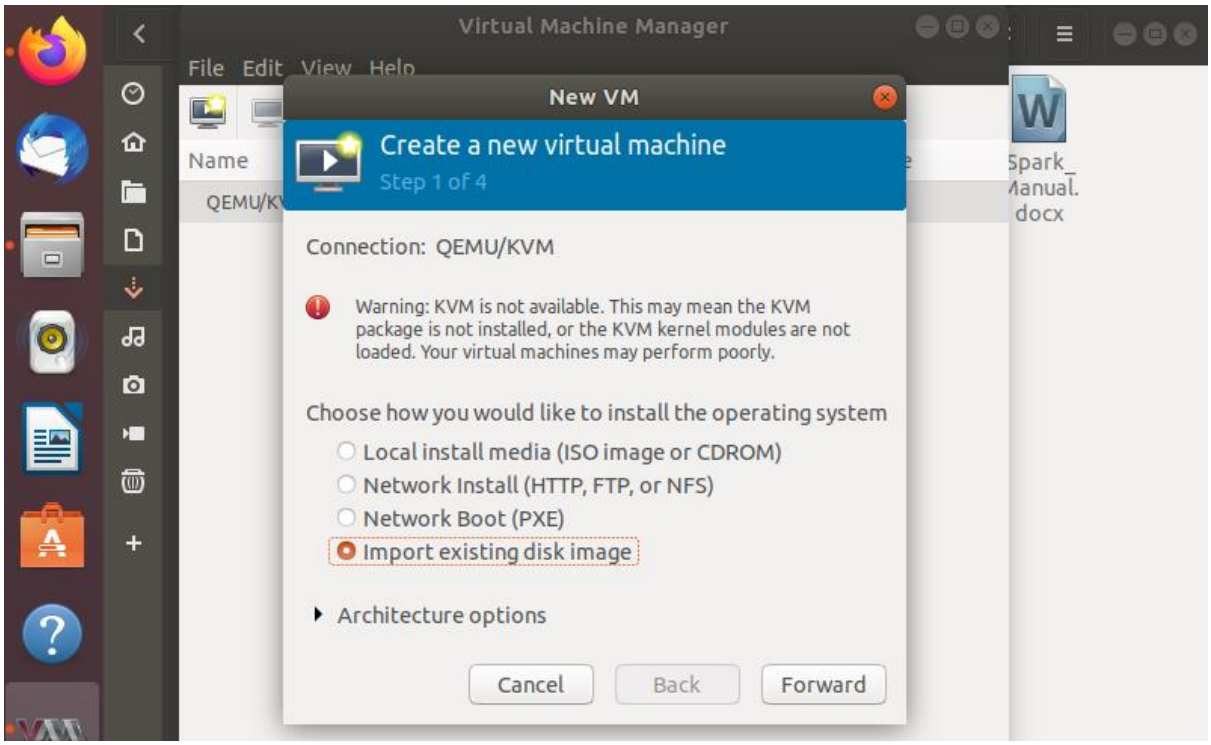
```
virsh # nodeinfo
CPU model:           x86_64
CPU(s):              2
CPU frequency:       1996 MHz
CPU socket(s):       1
Core(s) per socket:  2
Thread(s) per core:  1
NUMA cell(s):        1
Memory size:         3927860 KiB
```

Importing existing VM

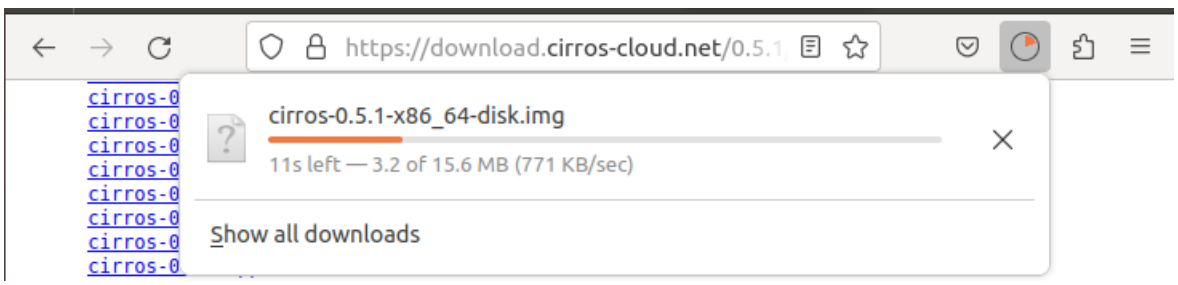
1. Go to virtual machine manager (virt-manager) in your linux system. Then, right click QEMU/KVM and select *New* to create new VM.



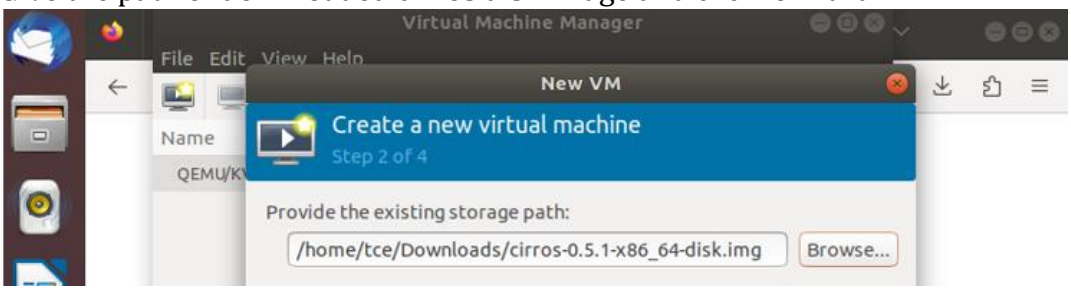
2. Select *import existing disk image* and click forward.



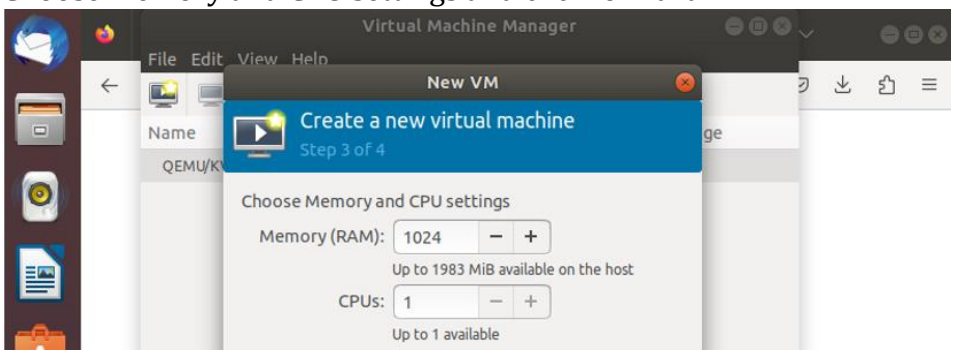
3. Download cirros disk image from the link: https://download.cirros-cloud.net/0.5.1/



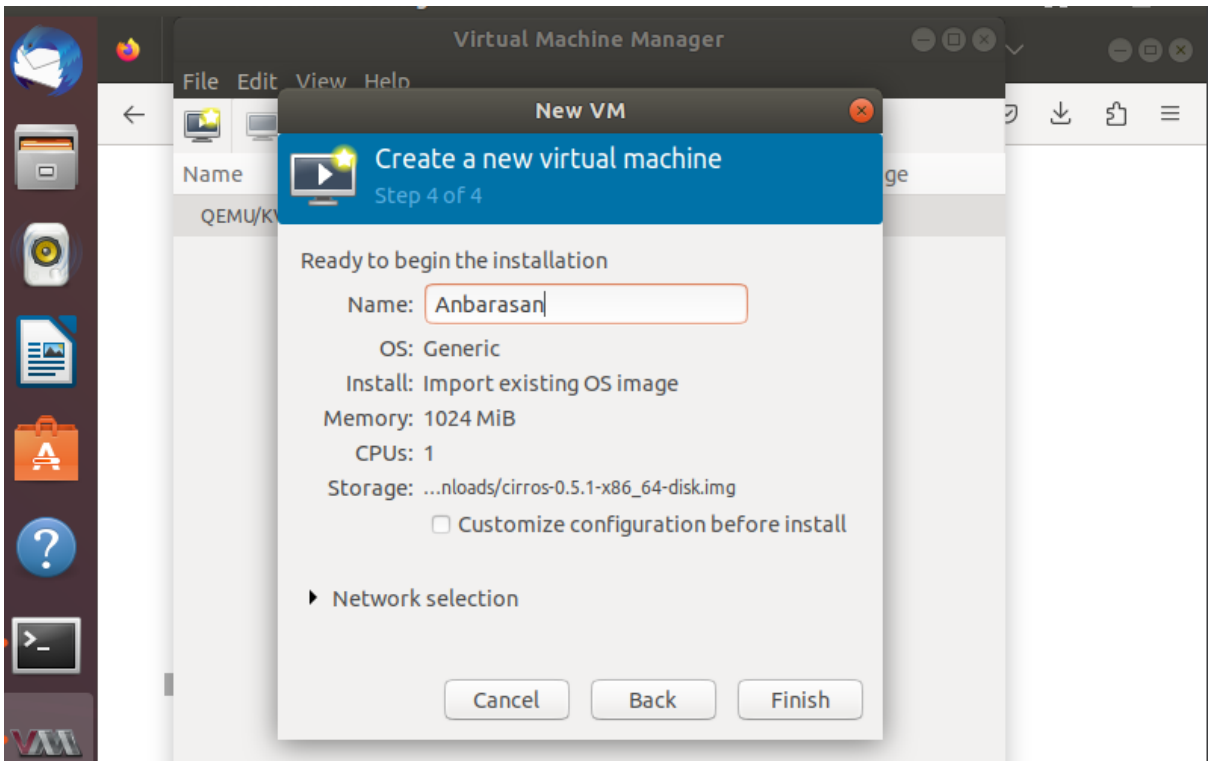
4. Give the path of downloaded cirros disk image and click forward.

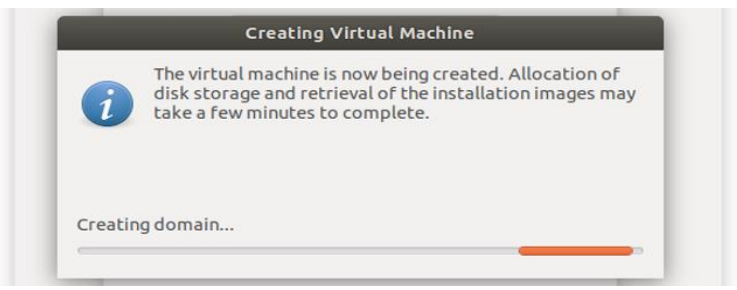


5. Choose memory and CPU settings and click forward.

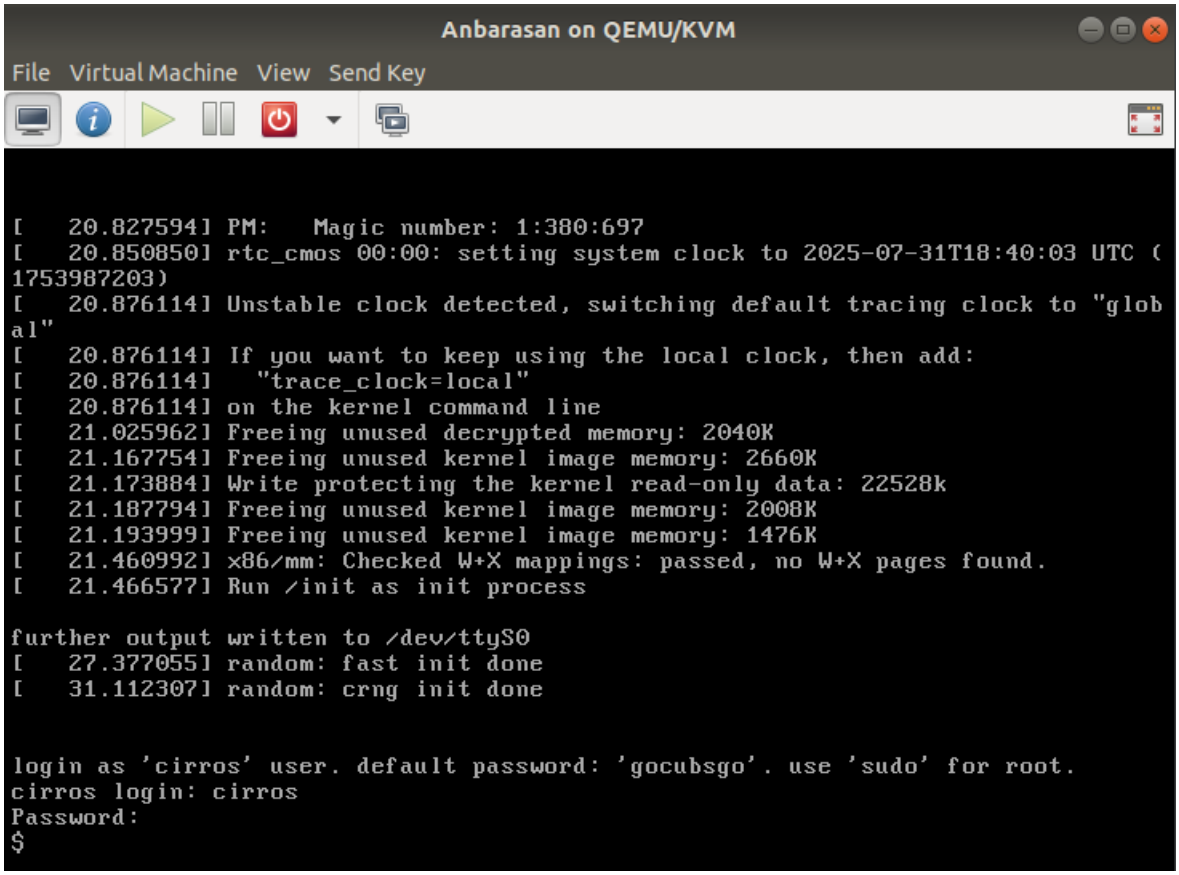


6. Name your VM and click Finish.





7. The VM is booted successfully and the login was also successful.



Result:

Thus, the KVM was successfully installed on the Linux system, and virtual instances were created and configured using Virt-Manager, demonstrating the setup of a virtualized environment.