

Cloud Computing

Prepared by Dr. Sherif Mostafa

IT381

Lecture 3



Cloud Service Models

Cloud Service Models

IaaS

PaaS

SaaS

What is KVM

How does KVM work

Understanding Cloud Service Models: IaaS, PaaS, SaaS, and Beyond

Cloud computing has revolutionized the way businesses operate, offering flexible and scalable solutions to meet diverse IT needs. Among the various cloud service models, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) are the most prominent. Each of these models serves distinct purposes and caters to different user requirements.

Understanding Cloud Service Models: IaaS, PaaS, SaaS, and Beyond

Additionally, newer models like Function as a Service (FaaS), Desktop as a Service (DaaS), Container as a Service (CaaS), and Backend as a Service (BaaS) have emerged, further expanding the cloud landscape. This chapter explores these service models in detail.

1. Infrastructure as a Service (IaaS)

IaaS provides virtualized computing resources over the internet, allowing businesses to rent IT infrastructure such as servers, storage, and networking. This model eliminates the need for physical hardware investments, enabling organizations to focus on their core operations.

Management

In an IaaS environment, users are responsible for managing the operating systems, applications, and middleware, while the service provider takes care of the physical infrastructure.

1. Infrastructure as a Service (IaaS)

Scalability and Cost-Effectiveness

IaaS offers on-demand resources, allowing businesses to scale their infrastructure up or down based on fluctuating workloads. This flexibility can lead to significant cost savings compared to maintaining on-premises infrastructure, as users only pay for the resources they consume.

Use Cases

- Hosting websites and applications
- Data storage and backup solutions
- Development and testing environments
- High-performance computing (HPC) applications

1. Infrastructure as a Service (IaaS)

Examples

- Amazon Web Services (AWS) EC2: Provides scalable computing capacity in the cloud.
- Microsoft Azure: Offers a range of virtual machines and networking options.
- Google Cloud Compute Engine: Allows users to run virtual machines on Google's infrastructure.

2. Platform as a Service (PaaS)

PaaS provides a platform that enables developers to build, deploy, and manage applications without the complexities of managing the underlying infrastructure. It includes tools and services specifically designed for application development.

Management

In a PaaS model, the service provider manages the infrastructure, operating systems, and middleware, allowing users to focus solely on application development and deployment.

2. Platform as a Service (PaaS)

Development Tools and Rapid Development

PaaS typically includes integrated development environments (IDEs), database management systems, and application hosting services. This accelerates the development process by providing pre-built components and services, facilitating collaboration among development teams.

Use Cases

- Developing web applications and APIs
- Building mobile applications
- Integrating with databases and third-party services

2. Platform as a Service (PaaS)

Examples

- Google App Engine: A fully managed platform for building and deploying applications.
- Microsoft Azure App Service: Supports various programming languages and frameworks for web and mobile applications.
- Heroku: A platform that enables developers to build, run, and operate applications entirely in the cloud.

3. Software as a Service (SaaS)

SaaS delivers software applications over the internet, allowing users to access them via a web browser without the need for local installation or maintenance. This model simplifies software delivery and usage.

Management

In a SaaS environment, the service provider manages everything, including infrastructure, application software, and security updates. Users simply access and use the software.

3. Software as a Service (SaaS)

Accessibility and Subscription Model

SaaS applications are accessible from any device with internet connectivity, making them convenient for remote work and collaboration. Typically, SaaS operates on a subscription basis, allowing users to pay for what they use, which can be more cost-effective than traditional software licensing.

Use Cases

- Customer relationship management (CRM) systems
- Enterprise resource planning (ERP) software
- Collaboration and productivity tools
- Email and communication services

3. Software as a Service (SaaS)

Examples

- Google Workspace: A suite of productivity and collaboration tools, including Gmail, Docs, and Drive.
- Microsoft 365: Offers applications like Word, Excel, and Teams as cloud-based services.
- Salesforce: A leading CRM platform that helps businesses manage customer relationships.

Additional Cloud Service Models

4. Function as a Service (FaaS)

FaaS is a serverless computing model that allows developers to run individual functions or pieces of code in response to events without managing servers. The cloud provider handles the infrastructure, scaling, and execution of functions. This model is ideal for microservices architectures and event-driven applications. Examples include AWS Lambda, Azure Functions, and Google Cloud Functions.

Additional Cloud Service Models

5. Desktop as a Service (DaaS)

DaaS provides virtual desktops hosted in the cloud, allowing users to access their desktop environments from any device. The service provider manages the infrastructure, security, and updates, making it suitable for remote work and disaster recovery. Examples include Amazon WorkSpaces, Citrix Virtual Apps and Desktops, and VMware Horizon Cloud.

Additional Cloud Service Models

6. Container as a Service (CaaS)

CaaS offers a container management service that allows users to deploy and manage containerized applications. The service provider manages the underlying infrastructure, while users focus on deploying and scaling their containers. This model is beneficial for microservices architectures and DevOps practices. Examples include Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), and Azure Kubernetes Service (AKS).

What is KVM?

A virtual machine is a software application that acts as an independent computer within another physical computer. It shares resources like CPU cycles, network bandwidth, and memory with the physical machine.

KVM is a Linux operating system component that provides native support for virtual machines on Linux. It has been available in Linux distributions since 2007.

What does a hypervisor do?

Virtualization, which lets you run an operating system (OS) within another OS, is possible relying on the hypervisors.

A hypervisor pools computing resources—like processing, memory, and storage—and reallocates them among VMs. A hypervisor can run multiple VMs at once, manage them, and support the creation of new ones. The physical hardware, when used as a hypervisor, is called the host, while the many VMs that use its resources are guests.

What does a hypervisor do?

Hypervisors need 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 a network card, graphics adapter, CPU(s), memory, and disks.

Why is KVM important?

Kernel-based Virtual Machine (KVM) can turn any Linux machine into a bare-metal hypervisor. This allows developers to scale computing infrastructure for different operating systems without investing in new hardware. KVM frees server administrators from manually provisioning virtualization infrastructure and allows large numbers of virtual machines to be deployed easily in cloud environments.

Why is KVM important?

With KVM, a VM is a Linux process, scheduled and managed by the kernel. VMs running with KVM benefit from the performance features of Linux, and users can take advantage of the fine-grained control provided by the Linux scheduler. KVM also brings features related to security, storage, hardware support, and live migration.

Businesses use KVM because of the following advantages

High performance

KVM is engineered to manage high-demanding applications seamlessly. All guest operating systems inherit the high performance of the host operating system (Linux). The KVM hypervisor also allows virtualization to be performed as close as possible to the server hardware, which further reduces process latency.

Businesses use KVM because of the following advantages

Security

Virtual machines running on KVM enjoy security features native to the Linux operating system, including Security-Enhanced Linux (SELinux). This ensures that all virtual environments strictly adhere to their respective security boundaries to strengthen data privacy and governance.

When you're running multiple VMs, a virtualization management tool is helpful for keeping track of them. Some VM management tools run from the command line, others provide graphical user interfaces (GUIs), and others are designed to manage VMs across large enterprise environments.

Businesses use KVM because of the following advantages

Here are a few common virtualization management solutions for KVM.

- The libvirt project provides an API for managing virtualization platforms. Within libvirt, virsh is a command-line utility for creating, starting, listing, and stopping VMs, as well as entering a virtualization shell.
- Virtual Machine Manager (known as VMM or virt-manager) provides a desktop interface for VMs and is available for major Linux distributions.
- VM administrators can choose to manage their VMs using web-based interfaces. For example, Cockpit offers a solution that lets users manage VMs from a web interface. Red Hat Enterprise Linux offers a web console plug-in for virtualization.

Businesses use KVM because of the following advantages

Security

So, KVM uses a combination of Security-Enhanced Linux (SELinux) and (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.

Businesses use KVM because of the following advantages

Stability

KVM has been widely used in business applications for more than a decade. It enjoys excellent support from a thriving open-source community. The source code that powers KVM is mature and provides a stable foundation for enterprise applications.

Cost efficiency

KVM is free and open source, which means businesses do not have to pay additional licensing fees to host virtual machines.

Businesses use KVM because of the following advantages

Flexibility

KVM provides businesses many options during installations, as it works with various hardware setups. Server administrators can efficiently allocate additional CPU, storage, or memory to a virtual machine with KVM. KVM also supports thin provisioning, which only provides the resources to the virtual machine when needed.

Storage flexibility

KVM is able to use any storage supported by Linux, including some local disks and network-attached storage (NAS). KVM also supports shared file systems so VM images may be shared by multiple hosts.

Businesses use KVM because of the following advantages

Support for multiple hardware architectures

KVM can run on a wide variety of hardware platforms. When used as part of Red Hat Enterprise Linux 9, KVM is supported with 64-bit AMD, Intel, and ARM architectures, as well as IBM z13 systems and later.

Live migration

KVM supports live migration, which is the ability to move a running VM between physical hosts with no noticeable 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 state so it can be stored and resumed later.

How does KVM work?

Kernel-based Virtual Machine (KVM) requires a Linux kernel installation on a computer powered by a CPU that supports virtualization extensions. Specifically, KVM supports all x86 CPUs, a family of computer chips capable of processing the Intel x86 instruction language.

How does KVM work?

Linux kernel is the core of the open-source operating system. A kernel is a low-level program that interacts with computer hardware. It also ensures that software applications running on the operating system receive the required computing resources. Linux distributions, such as Red Hat Enterprise Linux, Fedora, and Ubuntu, pack the Linux kernel and additional programs into a user-friendly commercial operating system.

How does KVM work?

Linux kernel

is the core of the open-source operating system. A kernel is a low-level program that interacts with computer hardware. It also ensures that software applications running on the operating system receive the required computing resources. Linux distributions, such as Red Hat Enterprise Linux, Fedora, and Ubuntu, pack the Linux kernel and additional programs into a user-friendly commercial operating system.

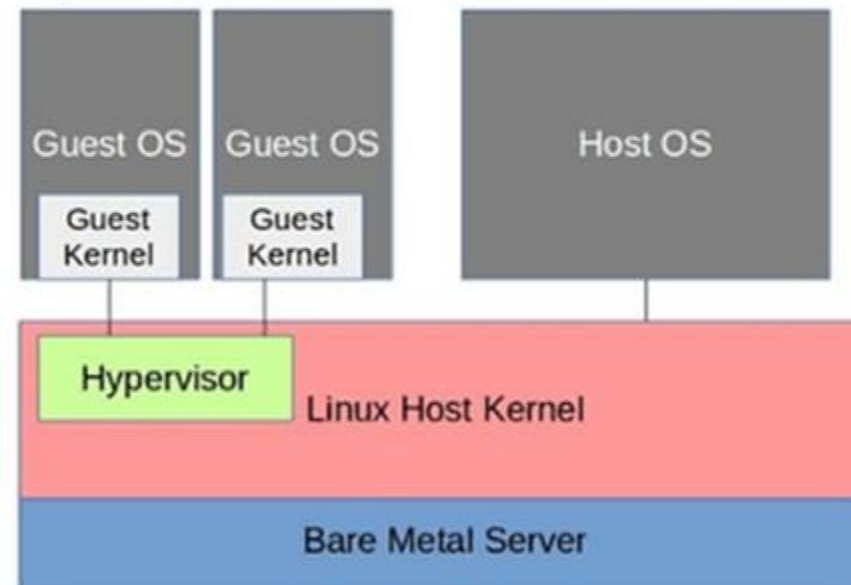
How to enable KVM

Once you have installed the Linux kernel, you need to install the following additional software components on the Linux machine:

- A host kernel module
- A processor-specific module
- An emulator
- A range of other Linux packages for expanding KVM's capabilities and performance

How to enable KVM

Once loaded, the server administrator creates a virtual machine via the command line tool or graphical user interface. KVM then launches the virtual machine as an individual Linux process. The hypervisor allocates every virtual machine with virtual memory, storage, network, CPU, and resources.



KVM Hypervisor architecture

What is the difference between KVM and VMware?

VMware is the software company that produces VMware ESXi, a commercially licensed virtualization solution. VMware hypervisors are used for enterprise applications, with virtual machines capable of handling heavy workloads.

Kernel-based Virtual Machine (KVM) and VMware ESXi both provide virtualization infrastructure to deploy type 1 hypervisors on the Linux kernel. However, KVM is an open-source feature while VMware ESXi is available via commercial licenses.

Organizations using VMware's virtualization components enjoy professional support from its technical team. Meanwhile, KVM users rely on a vast open-source community to address potential issues.

Why migrate to a KVM-based virtualization platform?

One of the benefits of VMs is flexibility—and portability. That portability makes it possible to move a VM from one virtualization provider to another to take advantage of lower costs, expanded capabilities, more control, or other advantages.

Developers consider KVM a mature technology, and enterprises trust it to handle VMs efficiently at scale. A migration to a KVM-based platform, such as Red Hat's virtualization portfolio, is a way to preserve existing VM investments while choosing a solution with an open-source foundation. Red Hat's technologies also make it possible to manage a unified environment for VMs, containers, and serverless apps.

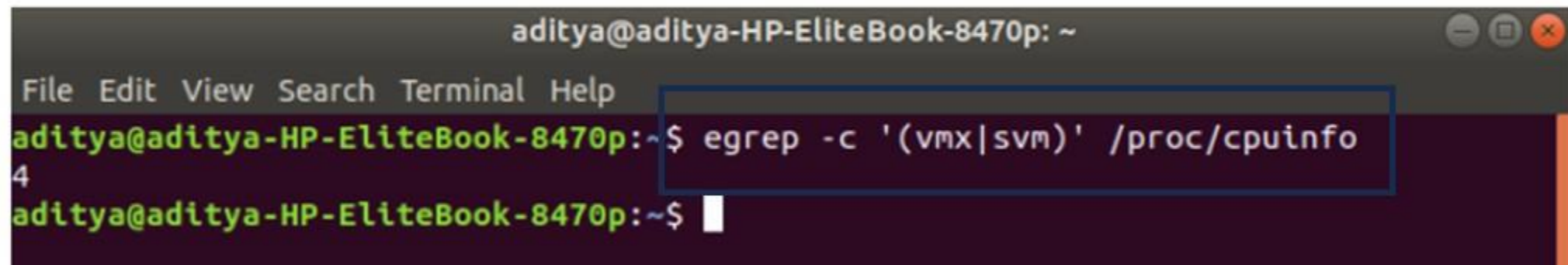
Steps for creating Virtual Machine in Linux using KVM

We are going to use ubuntu OS throughout this steps,

- Checking if virtualization is enabled or not
- Installing KVM on Ubuntu
- Adding a user to KVM
- Creating Virtual Machine

Steps for creating Virtual Machine in Linux using KVM

To create a virtual machine first we need to ensure that virtualization is enabled on our system. It is mandatory to create virtual machines. There are multiple ways to check if virtualization is enabled,

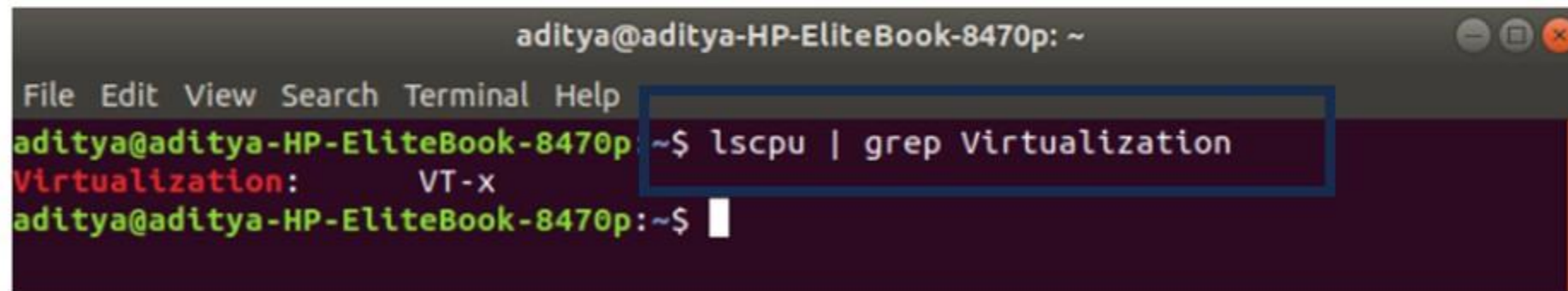
A terminal window titled 'aditya@aditya-HP-EliteBook-8470p: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'aditya@aditya-HP-EliteBook-8470p:~\$'. The command 'egrep -c '(vmx|svm)' /proc/cpuinfo' is entered and highlighted with a blue box. The output '4' is shown on the next line. The prompt 'aditya@aditya-HP-EliteBook-8470p:~\$' is shown again with a cursor.

```
aditya@aditya-HP-EliteBook-8470p: ~  
File Edit View Search Terminal Help  
aditya@aditya-HP-EliteBook-8470p:~$ egrep -c '(vmx|svm)' /proc/cpuinfo  
4  
aditya@aditya-HP-EliteBook-8470p:~$
```

The above command `/proc/cpuinfo` gives information about the processor. The output of the command will be a number. The output number 1 or more than that represents that virtualization is enabled, output 0 says you need to enable virtualization on your system.

Steps for creating Virtual Machine in Linux using KVM

This command is used to check which type of virtualization your processor supports. If the system contains a CPU with Intel VT support, the above command will provide the following output.



```
aditya@aditya-HP-EliteBook-8470p: ~  
File Edit View Search Terminal Help  
aditya@aditya-HP-EliteBook-8470p:~$ lscpu | grep Virtualization  
Virtualization:      VT-x  
aditya@aditya-HP-EliteBook-8470p:~$
```

Installing KVM on Ubuntu

Now we know our system is capable of creating and running virtual machines, it's time to install tools which will create our virtual machines. To install KVM and other KVM dependencies such as virt-manager, bridge-utils enter command:

```
$ sudo apt -y install bridge-utils cpu-checker  
libvirt-clients libvirt-daemon qemu qemu-kvm
```

Installing KVM on Ubuntu

```
$ sudo apt -y install bridge-utils cpu-checker  
libvirt-clients libvirt-daemon qemu qemu-kvm
```

- bridge-utils: The bridge-utils package contains a utility needed to create and manage bridge devices. This is useful in setting up networks for a hosted virtual machine (VM).
- cpu-checker: Outputs the specifications of CPU

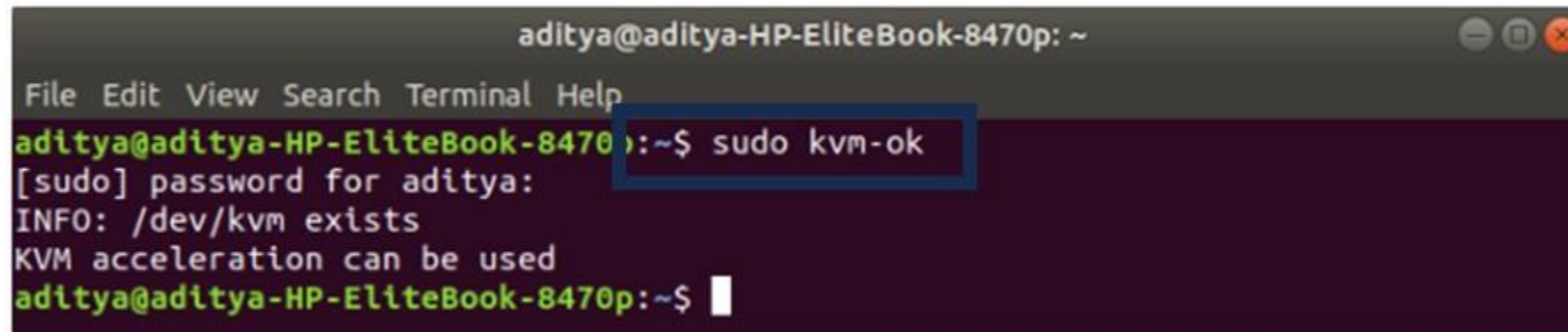
Installing KVM on Ubuntu

```
$ sudo apt -y install bridge-utils cpu-checker  
libvirt-clients libvirt-daemon qemu qemu-kvm
```

- libvirt-clients: a toolkit to manage virtualization platforms/clients and hypervisors
- qemu: A program that can run the operating system of the machine on different machines
- qemu-kvm: Runs process using KVM module

Installing KVM on Ubuntu

All dependencies are installed now run command its time to check if KVM ins installed successfully:

A terminal window titled 'aditya@aditya-HP-EliteBook-8470p: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'sudo kvm-ok' being executed. The output indicates that KVM acceleration can be used. The command prompt is highlighted with a blue box.

```
aditya@aditya-HP-EliteBook-8470p: ~  
File Edit View Search Terminal Help  
aditya@aditya-HP-EliteBook-8470p:~$ sudo kvm-ok  
[sudo] password for aditya:  
INFO: /dev/kvm exists  
KVM acceleration can be used  
aditya@aditya-HP-EliteBook-8470p:~$
```

Also, we need to confirm if the virtualization daemon – libvirt-daemon – is running, to do so enter the command.

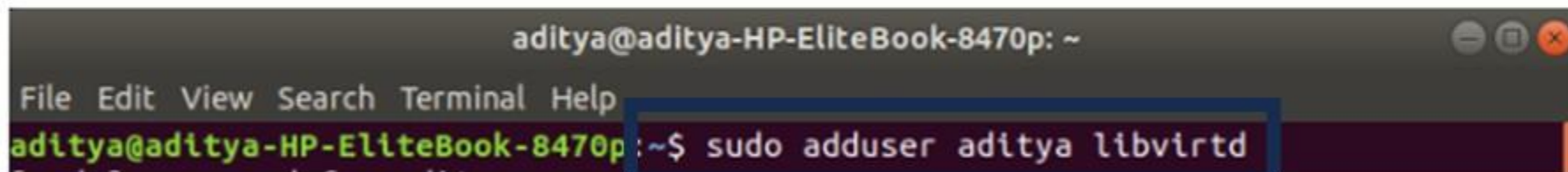
```
$ sudo systemctl status libvirt
```

Adding a user to KVM

In this part, we are going to create a user for KVM. To prevent root user from using KVM and root user is only available when root user is a part of libvirt/libvirtd group.

- To add a new user to KVM, use the following command,

```
sudo adduser [username] libvirtd
```

A screenshot of a Linux terminal window. The title bar shows 'aditya@aditya-HP-EliteBook-8470p: ~'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The prompt is 'aditya@aditya-HP-EliteBook-8470p: ~\$' and the command 'sudo adduser aditya libvirtd' is being entered. The command is highlighted with a blue selection box.

```
aditya@aditya-HP-EliteBook-8470p: ~$ sudo adduser aditya libvirtd
```

Creating Virtual Machine

There are two ways to create a virtual machine

- Using the command line
- Using the graphical interface

Create a Virtual Machine via Command Line

virt-install is a command which is used to create virtual machines in Linux, following is the command which creates a VM.

```
sudo virt-install --name=ubuntu-guest --os-variant=ubuntu20.04 --vcpu=2 --  
ram=2048 --graphics none --location=[local path to iso file] --network bridge:vibr0
```

The above command creates a Ubuntu virtual machine with version 20.04 and the name ubuntu-guest.

Name: specify the name of the virtual machine being created

vcpu: Number of virtual CPUs to configure for the guest.

Ram: Memory to allocate for guest instance in megabytes. According to your machine, you can specify the given memory of the VM.

Create a Virtual Machine via Command Line

```
sudo virt-install --name=ubuntu-guest --os-variant=ubuntu20.04 --vcpu=2 --  
ram=2048 --graphics none --location=[local path to iso file] --network bridge:vibr0
```

Graphics spice: If no graphics option is specified, “virt-install” will default to ‘--graphics vnc’ if the display environment variable is set, otherwise ‘--graphics none’ is used.

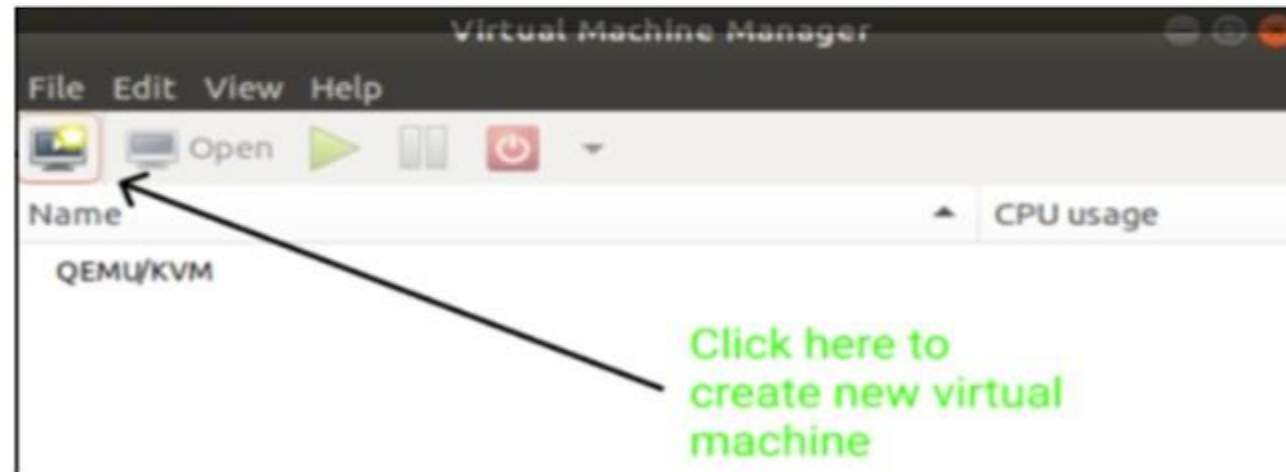
Location: location of iso file on which virtual machine will be built. It can be the path to an ISO image, or to a CDROM device. It can also be a URL from which to fetch/access a minimal boot ISO image.

Network-bridge: Connect the guest to the host network, Connect to a bridge device in the host called “BRIDGE”.

Create a Virtual machine using a graphical interface

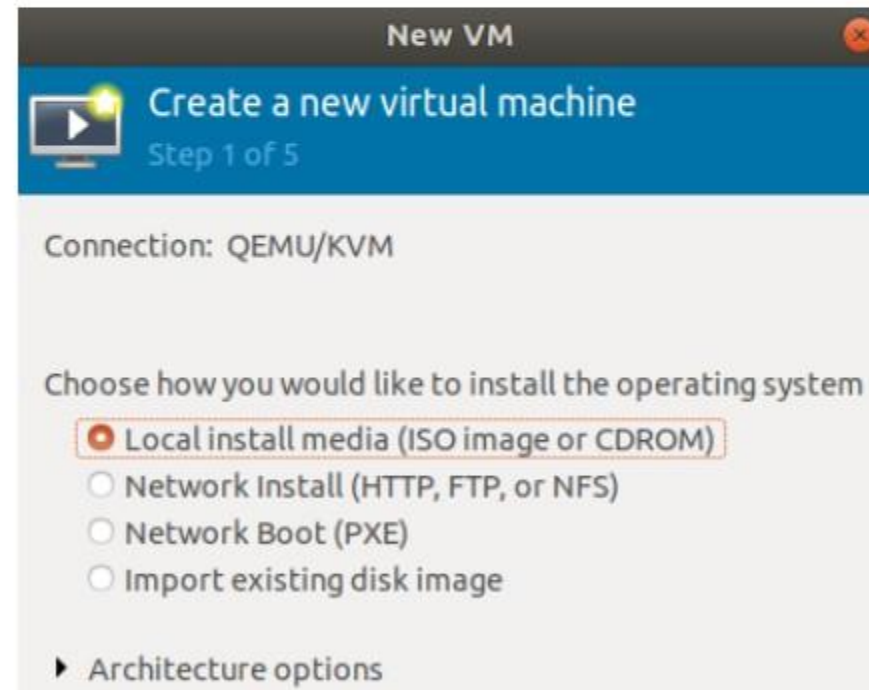
If you are not much familiar with the command line don't worry, there's another way to create a virtual machine using a tool called virt-manager you can easily create virtual machines. Steps to create VM using virt-manager,

Step 1: Launch virt-manager



Create a Virtual machine using a graphical interface

Step 2: Choose installation media it can be an iso file of OS, you can install from the network or can be a disk copy

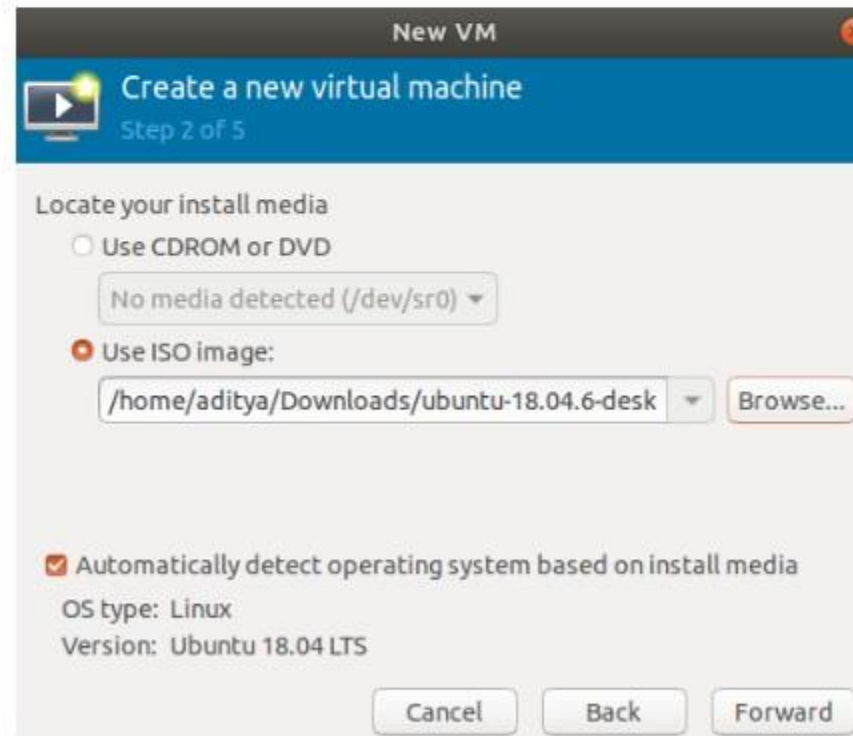


Chapter

03

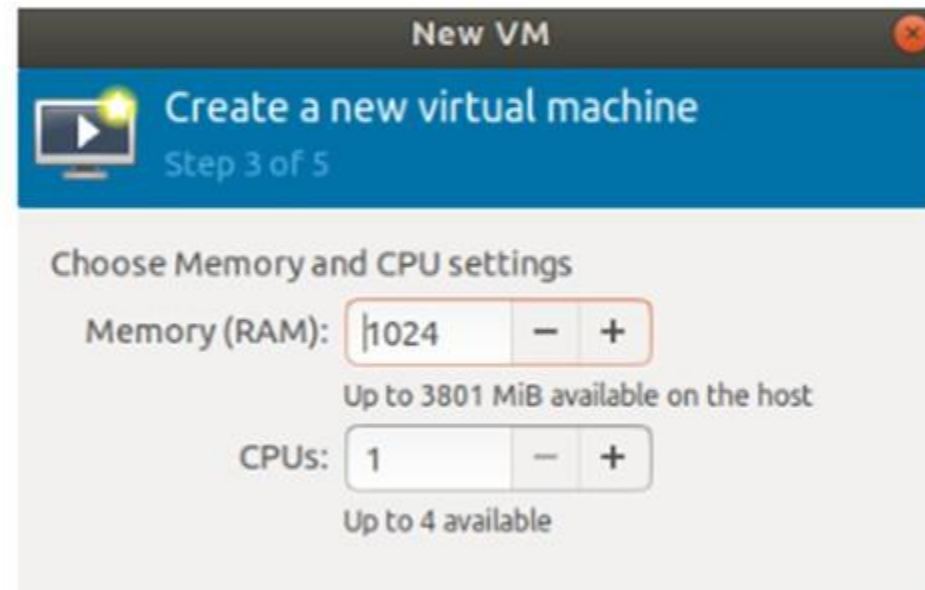
Create a Virtual machine using a graphical interface

Step 3: Enter the path/URL of your file click next



Create a Virtual machine using a graphical interface

Step 4: Enter memory and CPU requirements



New VM

Create a new virtual machine
Step 3 of 5

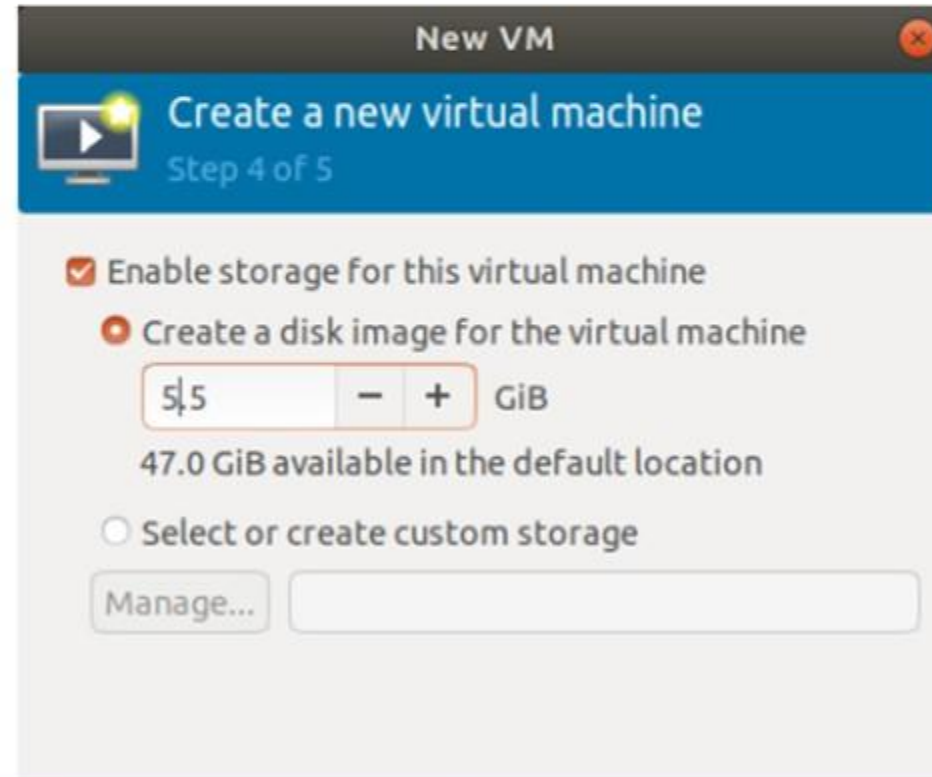
Choose Memory and CPU settings

Memory (RAM): 1024 - +
Up to 3801 MiB available on the host

CPUs: 1 - +
Up to 4 available

Create a Virtual machine using a graphical interface

Step 5: Enter required disk space



New VM

Create a new virtual machine
Step 4 of 5

☒ Enable storage for this virtual machine

☒ Create a disk image for the virtual machine

515 - + GiB

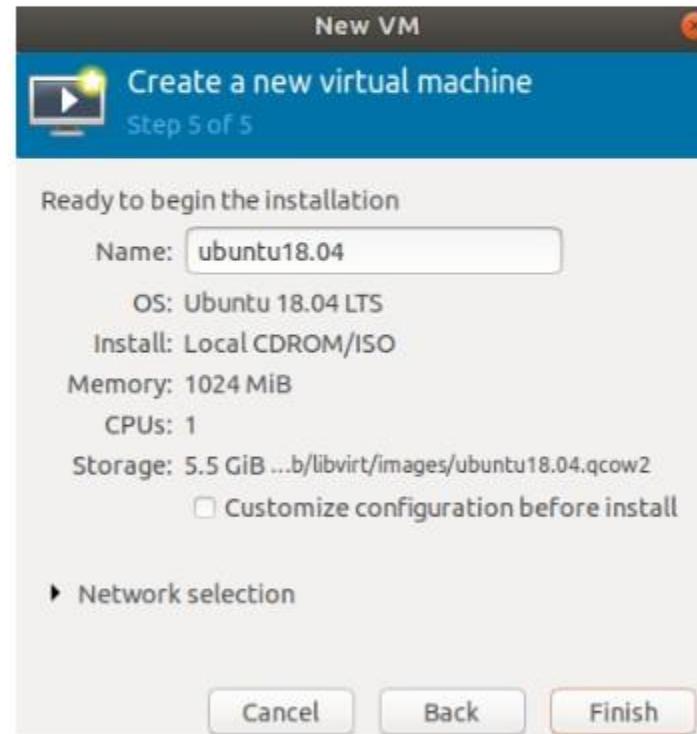
47.0 GiB available in the default location

☐ Select or create custom storage

Manage...

Create a Virtual machine using a graphical interface

Step 6: Enter the name of VM and check specifications and hit the finish



Now the installations should be completed, and the virtual machine should start running.



Thank You