What is a Virtual Machine (VM)?

A **Virtual Machine (VM)** is a **software-based computer** that runs inside a real (physical) computer. It acts just like a real computer, with its own **CPU, memory, storage, and operating system**, but everything is **virtual and managed by software**.

Why Use a Virtual Machine?

✅ To run multiple operating systems (e.g., Linux or Windows)

✅ To test sofftware in a safe, isolated environment

✅ To simulate servers for DevOps or cloud projects

✅ To learn and experiment without breaking your main system.

Example:

You're using a Windows laptop, but want to practice **Linux commands**.Instead of buying a new laptop, you install **VMware** or **VirtualBox**, create a **Linux virtual machine**, and run Linux inside a window — just like opening an app.

What is a Server?

A **server** is a **powerful computer or software** that provides services, resources, or data to other computers — called **clients** — over a network. It **"serves"** data when requested. That's why it's called a *server.*

Why Are Servers Used?

✅ To host websites and applications

✅ To store and manage large amounts of data

✅ To allow multiple users to access the same service or file

✅ To ensure 24/7 access from anywhere in the world

What Does a Server Do?

Servers can provide:

Websites (Web Server)

Emails (Mail Server)

Files (File Server)

Databases (Database Server)

Applications (App Server)

Example:

When you open a website like www.google.com, your browser sends a request to Google's **web server**. The server processes your request and sends back the webpage for you to view.

What is Virtualization?

Virtualization is the process of creating a **virtual version** of something like a computer, server, storage device, or network — using software. It allows you to run **multiple virtual systems** on a single **physical machine**.

Why Use Virtualization?

✅ To run multiple operating systems on one device

✅ To save money (fewer physical servers)

✅ To isolate apps/environments (great for testing)

✅ To quickly create, modify, or delete systems

✅ To use resources more efficiently (CPU, memory, storage)

Example:

A company has 1 powerful server. Instead of buying 5 more servers, they create 5 **Virtual Machines** on the same server — each running its own OS and apps. This saves **cost**, **space**, and **energy**.

What is a Hypervisor?

A **Hypervisor** is a special type of software (or sometimes firmware) that allows you to **create and run virtual machines (VMs)** on a single physical computer.

It manages the **hardware resources** (CPU, memory, disk, etc.) and shares them across multiple VMs, keeping them **isolated and secure**.

Types of Hypervisors:

Type	Description	Use Case	Examples
Type 1	**Bare-metal Hypervisor** (Runs directly on the physical hardware)	High performance Used in data centers	VMware ESXi Microsoft Hyper-V KVM Xen
Type 2	**Hosted Hypervisor** (Runs on top of an existing operating system)	Easier to use Great for personal use/testing	VirtualBox VMware Workstation Parallels

Why is a Hypervisor Used?

✅ To run multiple operating systems on one machine
✅ To isolate environments for security/testing
✅ To save cost on hardware
✅ To make systems easier to scale and manage
✅ To enable fast provisioning of new servers (VMs)

Example :

A cloud provider like **AWS** uses **Type 1 Hypervisors** on its servers to host thousands of VMs securely and efficiently for users across the world.

You, as a learner, might use **VirtualBox** (Type 2) to run **Linux** on your **Windows laptop** for practice.

Physical Server VS Virtual Server

Difference Between Virtual Server and Physical Server as follows:

Feature	Physical Server	Virtual Server
Nature	Real, tangible hardware machine	Software-based server running inside a physical machine
Resource Usage	Uses all CPU, RAM, storage for itself	Shares CPU, RAM, storage with other servers
Operating System	One OS runs directly on hardware	Each virtual server has its own OS (can be different OSes)
Flexibility	Harder to scale or move	Easy to scale, move, clone, or delete
Cost	More expensive (hardware, maintenance, power)	Cost-effective (multiple VMs on one server)
Deployment Time	Longer (needs setup, installation)	Much faster (just create a new VM)
Performance	High performance for single-purpose loads	Slight overhead due to shared resources
Failure Impact	If it crashes, all services go down	One VM crash doesn't affect others
Use Cases	High-performance needs, databases, legacy systems	Cloud computing, testing, DevOps labs, isolated environments

How to Create Virtual Machine in AWS, Azure and GCP

AWS (Amazon Web Services) – EC2 Instance

1. Go to AWS Console
2. Search for **EC2**, click **Launch Instance**
3. Choose OS (Ubuntu, Windows, etc.)
4. Select instance type (e.g., **t2.micro** – free tier)
5. Create/select key pair (.pem file)
6. Configure settings, then **Launch**
7. Connect using SSH: ssh -i "your-key.pem" ubuntu@your-ip
8.

Azure – Virtual Machine

1. Go to Azure Portal
2. Search for **Virtual Machines**, click **Create**
3. Set name, region, OS (Ubuntu, Windows)
4. Choose size (e.g., **B1s** – free tier)
5. Set username + SSH key or password
6. Review + Create VM
7. Connect via SSH or RDP

GCP (Google Cloud Platform) – Compute Engine

1. Go to GCP Console
2. Enable **Compute Engine**, click **Create Instance**
3. Set name, zone, OS (Ubuntu, etc.)
4. Choose **e2-micro** (free tier)

5. Allow firewall rules (SSH/HTTP)

6. Click **Create**

7. Click **SSH** to open terminal in browser

Importance of Automation in DevOps

Why Automation Matters in DevOps :

Area	How Automation Helps
Speed	Speeds up code build, test, and deployment processes
Consistency	Ensures same process every time (no manual mistakes)
Efficiency	Saves time and effort by reducing repetitive tasks
Quality	Automated testing catches bugs early
Scalability	Easily scale environments and deployments
Feedback	Gives real-time insights into builds, tests, and deployments
Collaboration	Frees up developers and ops to focus on innovation instead of manual work

Examples of Automation in DevOps

🔧 Task	⚙️ Tool Used
Code Integration	GitHub Actions, Jenkins
Testing	Selenium, JUnit, PyTest
Deployment	Ansible, Spinnaker, ArgoCD
Infrastructure Provisioning	Terraform, AWS CloudFormation
Monitoring & Alerts	Prometheus, Grafana, Alertmanager

Scripting in DevOps

Scripting helps DevOps engineers **automate**, **configure**, and **manage** cloud infrastructure efficiently.

Comparison Table

Tool	Type	Language	Best For
AWS CLI	CLI Tool	Shell/Bash	Simple tasks & scripting
CDK	IaC (Imperative)	Python, TypeScript	Dev-centric, reusable infra code
CloudFormation (CFT)	IaC (Declarative)	YAML/JSON	AWS-native deployments
Terraform	IaC (Declarative)	HCL (HashiCorp)	Multi-cloud infrastructure automation
API (boto3, SDKs)	SDK / Programming	Python, Java, etc.	Advanced automation, integrations