UNIT-3

Virtualization Technology

1. What is Virtualization?

Virtualization is a technology that allows a single physical computer (server, workstation, or storage device) to run multiple virtual environments or systems. These **virtual systems** behave as if they are actual physical machines, but they share the resources of the single physical host.

This is achieved using a **software layer** known as a **hypervisor** (or virtualization layer), which creates and manages these **virtual machines (VMs)**.

Key Components of Virtualization

1. Hypervisor

The hypervisor is the core of virtualization. It manages the physical hardware and enables the creation and operation of VMs.

There are two main types of hypervisors:

- **Type 1 Hypervisor (Bare-metal):**
  - Runs directly on the physical hardware.
  - Examples: VMware ESXi, Microsoft Hyper-V, Xen, KVM.
  - Offers better performance and is used in data centers or enterprise environments.
- **Type 2 Hypervisor (Hosted):**
  - Runs on top of an existing operating system.
  - Examples: VMware Workstation, Oracle VirtualBox, Parallels Desktop.
  - Easier to set up, used mostly in development or testing environments.

2. Virtual Machines (VMs)

A **VM** is a software emulation of a physical computer. Each VM contains:

- A virtual CPU
- Virtual RAM
- Virtual storage (disk)
- A virtual network interface
- Its own **operating system** and applications

To the operating system and apps inside the VM, it appears they are running on real hardware.

How Virtualization Works

1. The physical computer has a **hypervisor** installed on it.
2. The hypervisor allocates **hardware resources** (CPU, RAM, disk space, etc.) to each **VM**.
3. Multiple VMs run **independently and simultaneously** on the same physical machine.
4. Each VM can run a **different operating system** (Windows, Linux, etc.).
5. The hypervisor manages the execution of each VM and ensures that they do not interfere with each other.

Types of Virtualization

1. **Hardware Virtualization**
   - Most common type, involves creating VMs.
   - Enables multiple OSes to run on a single hardware platform.
2. **Storage Virtualization**
   - Abstracts physical storage from multiple devices into one virtual storage pool.
   - Helps in better resource allocation and backup.
3. **Network Virtualization**
   - Combines hardware (like routers, switches) and software network resources into a single software-based administrative entity.
   - Enables creation of virtual networks.
4. **Desktop Virtualization**
   - Enables running desktop environments on a central server.
   - Users access their virtual desktops over a network.
5. **Application Virtualization**
   - Applications run on a central server but appear as if they are running locally on a user's device.

2. Key Benefits of Virtualization

1. Resource Optimization

Efficient Use of Physical Resources

- Traditionally, physical servers often run **one application per server**, leading to **low hardware utilization**.
- With virtualization, multiple **virtual machines (VMs)** can run on a single physical server, each hosting different applications or services.

Example:

Instead of having 10 physical servers each running one app at 10% capacity, you can consolidate those into **one or two servers**, each running 5 VMs at 50–70% capacity.

Result:

- Reduced hardware needs
- Lower energy consumption

- Smaller physical footprint (space, cooling, etc.)

2. Isolation

Independent Environments

- Each VM runs its **own operating system and applications**, completely separated from others.
- If one VM **crashes**, is **infected by malware**, or **experiences a failure**, it does **not impact** the other VMs or the host.

Ideal for Testing & Development

- Developers can create **test environments**, try new software, or simulate failures without risking the actual system.
- Once testing is done, VMs can be easily **deleted** or **restored**.

3. Scalability

Rapid Expansion

- Virtual machines can be:
    - **Cloned** (duplicated quickly),
    - **Scaled up** (more resources like RAM or CPU),
    - Or **migrated** to another host server.

Key in Cloud Environments

- Cloud providers (like AWS, Azure, Google Cloud) use virtualization to offer **elastic resources**:
    - Need 10 more servers for an app launch? Spin up 10 VMs in minutes.

Benefit:

- Businesses can **adapt instantly** to changing demands—no need to wait for physical hardware procurement or setup.

4. Flexibility and Portability

Move across Hardware & Locations

- VMs are stored as **files** (VM images), making them **easy to move** between physical machines or data centers.

Use Case:

- Need to move from one data center to another? Just **copy the VM image**, deploy it on a new host, and you're ready to go.

Backup & Disaster Recovery

- Backing up a VM is as simple as copying a file.
- In case of system failure, you can restore a VM quickly from a backup image—much faster than reinstalling an OS and apps manually.

5. Cost Savings

Reduced Hardware Investment

- Fewer physical machines needed = lower **hardware purchase** costs.
- Less equipment also means **lower energy consumption** and **reduced maintenance** needs (fewer parts, repairs, or replacements).

Licensing & Software Costs

- Many **open-source virtualization platforms** are free (e.g., KVM, Xen, VirtualBox), lowering software costs.
- Operating systems and applications can also be managed more efficiently in virtual environments, reducing licensing overhead.

Summary Table

Benefit	Description	Business Impact
Resource Optimization	Better hardware utilization, less waste	Saves space, power, and money
Isolation	VMs are independent and secure	Minimizes risk, enhances testing
Scalability	Easily scale systems up/down as needed	Supports business agility
Flexibility & Portability	Move, back up, or restore systems quickly	Simplifies management and recovery
Cost Savings	Fewer servers, less power, open-source tools	Significant long-term cost reductions

3. Implementation Levels of Virtualization

Virtualization is a technology that allows you to create multiple simulated environments or dedicated resources from a single, physical hardware system. It separates the hardware from the software, allowing multiple operating systems (OS) or applications to run independently on the same physical machine.

Implementation levels of virtualization refer to the layers at which virtualization can be applied in a computing system. Each level abstracts different resources and serves different purposes. These levels are:

1. Instruction Set Architecture (ISA) Level Virtualization

Description:

- This is the **lowest level** of virtualization.
- It virtualizes the **Instruction Set Architecture**, which is the interface between software (usually the OS) and the hardware.

How it works:

- It provides a different ISA to the guest system than the host system.
- An emulator is used to simulate a hardware architecture that is different from the host's.

Example:

- Running x86 applications on ARM architecture using emulators (e.g., **QEMU**).

Pros:

- Can run software designed for a completely different hardware platform.
- Provides high portability.

Cons:

- Very **slow performance** due to instruction-level translation/emulation.

2. Hardware Level Virtualization (System-Level Virtualization)

Description:

- Also known as **full virtualization** or **bare-metal virtualization**.
- Virtualization is implemented **directly on the hardware**.
- A software layer called the **hypervisor (VMM - Virtual Machine Monitor)** interacts directly with the physical hardware and manages guest OSs.

Types:

- **Type 1 Hypervisors (Bare-metal):** Installed directly on hardware.
  - E.g., VMware ESXi, Microsoft Hyper-V, Xen, KVM
- **Type 2 Hypervisors (Hosted):** Run on a host OS.
  - E.g., VMware Workstation, VirtualBox

Pros:

- Near-native performance.
- Strong isolation between VMs.
- High scalability and efficiency.

Cons:

- Complex setup (especially Type 1).
- Requires compatible hardware for optimal performance.

3. Operating System Level Virtualization

Description:

- Virtualization is implemented at the **OS kernel level**.
- The OS allows multiple isolated user-space instances (called containers) to run on a single kernel.

How it works:

- Containers share the **same kernel** but have their own file system, network stack, and process space.
- Uses features like **chroot**, **cgroups**, and **namespaces**.

Example:

- **Docker**, **LXC (Linux Containers)**, **OpenVZ**

Pros:

- Lightweight and fast.
- Low overhead since there's no need to virtualize the entire OS.
- High density of containers per host.

Cons:

- All containers must use the **same OS kernel**.
- Less isolation compared to VMs (though this is improving).

4. Library Level Virtualization

Description:

- Virtualization occurs at the **application library level**.
- Applications use a **common set of libraries** that are virtualized or emulated.

How it works:

- A library or runtime acts as a bridge between the app and the OS, allowing portability and compatibility.
- Used in cross-platform software development.

Example:

- **Wine** (runs Windows apps on Linux by translating Windows API calls).
- Java Virtual Machine (JVM) for Java programs.

Pros:

- Enables applications to run on platforms they weren't originally designed for.
- Reduces compatibility issues.

Cons:

- Performance overhead due to translation/emulation.
- Not all applications are supported.

5. Application Level Virtualization

Description:

- Virtualizes individual applications rather than the entire OS or hardware.
- The application runs in a **sandboxed environment**, isolated from the host OS.

How it works:

- The application is packaged with all necessary files and dependencies into a virtual container.
- No changes needed on the host OS.

Example:

- **Microsoft App-V**, **VMware ThinApp**, **Cameyo**

Pros:

- Simplifies application deployment.
- No conflicts between applications.
- Easy rollback and updates.

Cons:

- Some applications may not function correctly in a virtualized state.
- Licensing issues may arise for virtualized apps.

Summary Table:

Level	Main Focus	Examples	Performance	Isolation	Flexibility
ISA Level	Hardware architecture emulation	QEMU	Low	High	High
Hardware Level	Full system virtualization	VMware ESXi, Hyper-V, KVM	High	High	Medium
OS Level	Container-based isolation	Docker, LXC	Very High	Medium-High	Medium
Library Level	API emulation	Wine, JVM	Medium-Low	Low	High
Application Level	App sandboxing	App-V, ThinApp	High	Medium	Medium

What is a Hypervisor?

A **hypervisor**, also called a **Virtual Machine Monitor (VMM)**, is a piece of software or firmware that creates and manages virtual machines by abstracting physical hardware.

Each VM behaves like a real computer with its own CPU, memory, storage, and network interfaces, while the hypervisor manages and distributes these resources.

Types of Hypervisors

Type	Description	Examples
Type 1 (Bare-Metal)	Installed directly on physical hardware. No host OS. Offers better performance and efficiency.	VMware ESXi, Microsoft Hyper-V, Xen, KVM
Type 2 (Hosted)	Runs on top of a host OS. Easier to install and use for desktop virtualization.	VMware Workstation, Oracle VirtualBox, Parallels Desktop

Type 1 Hypervisors (Bare-Metal)

1. VMware ESXi

- **Vendor:** VMware
- **Deployment:** Installed directly on server hardware.
- **Features:**
    - Enterprise-grade performance and stability.
    - Centralized management via **vCenter Server**.
    - Supports live migration (**vMotion**), snapshots, and high availability.
- **Use Case:** Data centers, enterprise environments.

---

2. Microsoft Hyper-V

- **Vendor:** Microsoft
- **Deployment:** Built into Windows Server and available on Windows 10/11 Pro/Enterprise.
- **Features:**
    - Tight integration with Windows.
    - Supports nested virtualization, checkpoints, and dynamic memory.
    - Managed via **Hyper-V Manager** or **System Center**.
- **Use Case:** Windows-based enterprise environments.

---

3. Xen Hypervisor

- **Vendor:** Open-source (managed by the Xen Project)
- **Deployment:** Runs directly on hardware; supports **para-virtualization** and **full virtualization**.
- **Features:**
    - Lightweight and secure.
    - Used by AWS for EC2 instances.
    - Dom0 (privileged domain) and DomU (guest domains) architecture.
- **Use Case:** Cloud computing, Linux-based environments.

---

4. KVM (Kernel-based Virtual Machine)

- **Vendor:** Open-source (part of Linux kernel)
- **Deployment:** Converts the Linux OS into a hypervisor.
- **Features:**
    - Integrated into most Linux distributions (Ubuntu, CentOS, RHEL).
    - Works with management tools like **libvirt**, **virt-manager**, and **OpenStack**.
    - Supports hardware acceleration via Intel VT-x and AMD-V.
- **Use Case:** Linux environments, open-source cloud platforms.

Type 2 Hypervisors (Hosted)

1. VMware Workstation / VMware Fusion

- **VMware Workstation:** For Windows/Linux
- **VMware Fusion:** For macOS
- **Features:**
  - User-friendly GUI for managing VMs.
  - High-performance VMs for development and testing.
  - Snapshots, cloning, and remote VM access.
- **Use Case:** Software development, desktop virtualization.

---

2. Oracle VirtualBox

- **Vendor:** Oracle (Open-source)
- **Deployment:** Runs on Windows, macOS, Linux.
- **Features:**
  - Easy to use.
  - Supports multiple OSs as guest systems.
  - Shared folders, snapshots, USB pass-through.
- **Use Case:** Personal use, education, cross-platform testing.

---

3. Parallels Desktop

- **Vendor:** Parallels (now owned by Corel)
- **Deployment:** macOS only
- **Features:**
  - Seamlessly runs Windows alongside macOS.
  - Optimized for Mac hardware.
  - Supports DirectX and OpenGL for Windows apps.
- **Use Case:** Running Windows on Macs, productivity tools.

Other Virtualization Tools and Mechanisms

Containers vs Hypervisors

- **Containers (Docker, LXC):** Lightweight virtualization at the OS level.
- **Hypervisors:** Full virtualization of OS + hardware.

Feature	**Hypervisors (VMs)**	**Containers**
Boot Time	Slower	Fast
Resource Usage	High	Low
Isolation Level	High	Medium
OS Flexibility	Any OS	Same kernel only
Examples	VMware, KVM	Docker, LXC

Comparison of Popular Hypervisors

Feature	**VMware ESXi**	**Hyper-V**	**KVM**	**Virtual Box**	**Xen**
Type	Type 1	Type 1	Type 1	Type 2	Type 1
License	Commercial	Free & Paid	Open-source	Free	Open-source
Performance	Excellent	Very Good	Excellent	Moderate	Excellent
Ease of Use	Moderate	Easy	Moderate	Very Easy	Moderate
Platform Support	x86-64	x86-64	Linux only	Cross-platform	Linux only
Used In	Enterprises	Enterprises	Linux, Cloud	Personal/Dev	Cloud, Hosting

1. VMware

What is VMware?

VMware is a **commercial virtualization platform** developed by **VMware Inc.** It offers a suite of virtualization products for both **desktop and server environments**. It's known for its **high performance, strong enterprise support**, and **easy-to-use GUI tools**.

Types of VMware Products:

- **VMware Workstation / Fusion** – Desktop virtualization for Windows/macOS.
- **VMware ESXi (formerly ESX)** – A bare-metal hypervisor for servers.
- **VMware vSphere** – A suite that includes ESXi and **vCenter Server** for management.
- **VMware vCloud / NSX / vSAN** – Cloud, networking, and storage virtualization.

Architecture:

- **Type 1 Hypervisor** (ESXi): Runs directly on the hardware without a host OS.
- **VMkernel**: A lightweight OS developed by VMware to manage VM operations.
- Offers tight integration with **vCenter** for management, HA, DRS, and vMotion.

Key Features:

- Live migration (vMotion)
- High Availability (HA)
- Distributed Resource Scheduler (DRS)
- vSAN for storage virtualization
- Enterprise-level support
- Proprietary drivers and features

Use Cases:

- Enterprise data centers
- Mission-critical applications
- Cloud environments (via VMware Cloud)

Pros:

- User-friendly GUI tools
- Strong support and stability
- Advanced features like vMotion, DRS

Cons:

- **License costs are high**
- Closed-source, proprietary
- Hardware compatibility list (HCL) must be followed strictly

---

2. KVM (Kernel-based Virtual Machine)

What is KVM?

KVM is an **open-source Type 1 hypervisor** built into the **Linux kernel**. It turns the Linux OS into a full-fledged hypervisor and supports both **Linux and Windows** guest VMs.

Developed initially by **Qumranet**, and acquired by **Red Hat**, KVM is widely used in **cloud infrastructure**, especially in **OpenStack**.

Architecture:

- **Type 1 Hypervisor**, but technically implemented as a **kernel module** (`kvm.ko`)
- Each VM is a **normal Linux process**
- Uses **QEMU** for hardware emulation and device management
- Supports **virtio** drivers for better I/O performance

Key Features:

- Integrated with the Linux kernel (low overhead)
- Uses hardware virtualization (Intel VT-x / AMD-V)
- Works with libvirt and tools like `virt-manager`, `virsh`
- Live migration with `virsh` and other tools
- Backed by Red Hat and the Linux community

Use Cases:

- Public/private clouds (e.g., OpenStack)
- Hosting providers
- Development/test environments

Pros:

- Free and open-source
- High performance, low overhead
- Scalable and secure
- Works with many Linux distros

Cons:

- More complex to set up than VMware
- Limited GUI tools (unless using third-party or Red Hat tools)
- Performance may depend on tuning

---

3. Xen (Xen Project Hypervisor)

What is Xen?

Xen is an **open-source, Type 1 hypervisor** originally developed at the University of Cambridge. Now maintained by the **Xen Project** (a Linux Foundation project), it's used in many commercial products like **Citrix Hypervisor (formerly XenServer)** and **AWS EC2**.

Architecture:

- **Type 1 bare-metal hypervisor**
- Uses a microkernel design
- Special management VM called **Domain 0 (Dom0)**:
    - A privileged Linux VM that manages other VMs
    - Loads drivers and handles I/O for guest VMs
- Guest VMs are called **DomUs** (unprivileged domains)

Virtualization Modes:

- **Paravirtualization (PV)**: Modified guest OS; better performance without hardware support.
- **Hardware Virtualization (HVM)**: Uses Intel VT/AMD-V; unmodified OS support.
- **PVH**: Hybrid model (better performance and compatibility)

Key Features:

- Strong isolation between VMs
- Paravirtualization support
- High availability
- Supports live migration
- Runs on x86, ARM

Use Cases:

- Cloud services (e.g., **AWS EC2 uses a modified Xen**)
- Virtual desktops (Citrix)
- Environments needing strong security/isolation

Pros:

- Very secure (used in sensitive environments)
- Flexible virtualization modes
- Open-source and highly customizable

Cons:

- Steeper learning curve
- Less community momentum than KVM
- Dom0 adds complexity
- Not as modern or widely supported as KVM

Comparison Summary Table:

Feature	VMware (ESXi)	KVM	Xen
Type	Type 1 (bare-metal)	Type 1 (via Linux kernel)	Type 1 (microkernel)
License	Proprietary	Open Source (GPL)	Open Source (GPL)

Feature	VMware (ESXi)	KVM	Xen
Host OS	None (VMkernel)	Linux	None (uses Dom0 Linux)
Management Tool	vCenter, vSphere	libvirt, virt-manager	XenCenter, XL, XAPI
Performance	Excellent	Excellent	Good (PV) / Better (HVM/PVH)
Ease of Use	Very easy	Moderate	Complex
Cloud Use	VMware Cloud	OpenStack, CloudStack	AWS, Citrix Cloud
Security	High	High	Very High
Live Migration	Yes (vMotion)	Yes	Yes
Cost	High	Free	Free