SNS COLLEGE OF TECHNOLOGY

An Autonomous Institution

Coimbatore-35

Department of Computer Science and Engineering

23CST206–OPERATING SYSTEMS AND VIRTUALIZATION

B.E- CSE /IV SEMESTER

UNIT – V HYPERVISOR

Topic 7:Overview of Virtual Machines

Virtual Machines & OS Portability

From "Write Once, Run Anywhere" to the Cloud Era

🔎 EMPATHIZE — Hook: The Software Problem

> *“ You write a program on Windows. Your friend runs Linux. Your professor uses macOS. Does it work everywhere? ”*

Student (Windows)

Writes code in VS Code, runs .exe

Professor (Linux)

Grade on Ubuntu server

CI/CD Pipeline

Auto-builds on Docker containers

📌 DEFINE — The Core Problem

PROBLEM STATEMENT

Hardware Dependency

Programs compiled for one CPU won't run on another (x86 vs ARM)

OS API Differences

Windows APIs ≠ Linux system calls ≠ macOS frameworks

Resource Conflicts

Multiple apps compete for memory, CPU, I/O — causing crashes

Security Isolation

One buggy app can corrupt others or the OS itself

KEY DEFINITIONS

Virtual Machine (VM)

A software emulation of a complete computer, running inside a real (host) computer.

Hypervisor

The software layer that creates and manages VMs. Also called VMM (Virtual Machine Monitor).

OS Portability

The ability of an OS or application to run on different hardware architectures without modification.

Abstraction Layer

A boundary that hides hardware complexity and presents a uniform interface to software above it.

DEFINE — Portability Spectrum & Why It Matters

PORTABILITY SPECTRUM

Not Portable	Low Portability	Medium	High	Full
(ASM, C w/ OS calls)	(OS-specific C/C++)	(Interpreted Python)	(Java JVM)	(Web / WASM)

→ *Increasing Portability*

THREE APPROACHES TO PORTABILITY

Source-Level Portability

Write in C/C++ with standard APIs (POSIX). Recompile per platform. Used in: Linux kernel, SQLite.

Binary Portability (VMs)

Compile once to bytecode; VM interprets on any platform. Used in: Java, Python, .NET CLR.

Container Portability

Package app + dependencies together. OS kernel shared. Used in: Docker, Kubernetes, microservices.

IDEATE — Think–Pair–Share #1

DISCUSSION QUESTION:

"Before VMs existed, engineers installed a DIFFERENT OS for every project. What problems do you think this caused? How did it waste time and money?"

Think (2 min)

Write your own answer — consider cost, time, boot time, backups

Pair (3 min)

Compare with your neighbour — did you identify the same problems?

Share (5 min)

One pair shares with the class — instructor maps to real history

IDEATE — Types of Virtual Machines

TYPE 1 & 2

System VM (Full Virtualization)

Virtualizes an entire hardware platform

- Runs a complete OS inside another OS
- Guest OS is unmodified (Type-1 & Type-2 hypervisors)
- Hardware resources (CPU, RAM, disk) are emulated or passed through
- Examples: VMware, VirtualBox, Microsoft Hyper-V, AWS EC2

BYTECODE

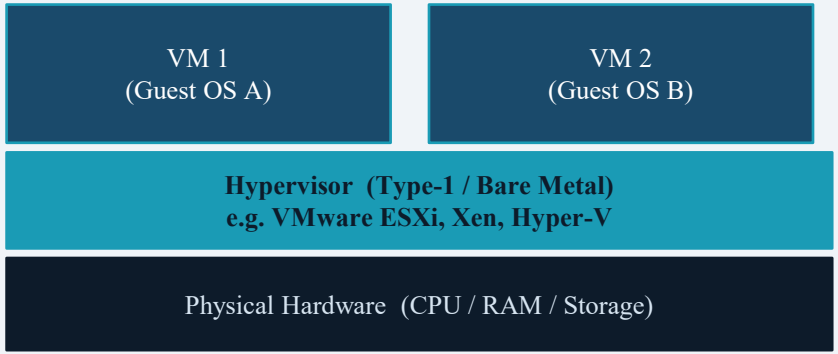
Process VM (Application VM)

Virtualizes a runtime for a specific application

- Translates high-level bytecode to native machine code at runtime
- OS-independent: compile once, run on any platform with the VM
- Classic example: Java Virtual Machine (JVM) — "Write Once, Run Anywhere"
- Examples: JVM, Python Interpreter, .NET CLR, WebAssembly

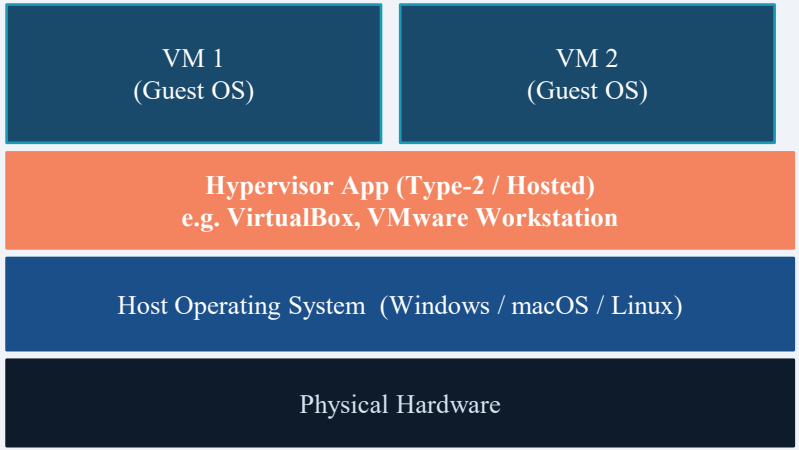
PROTOTYPE — VM Architecture: Type-1 vs Type-2 Hypervisor

TYPE-1: Bare Metal Hypervisor



✓ Faster (no Host OS overhead)
✓ More secure isolation
✓ Used in data centers & cloud

TYPE-2: Hosted Hypervisor



✓ Easy to install on laptops
✓ Good for development/testing
✗ Slower due to Host OS layer

PROTOTYPE — How a VM Executes: Step-by-Step

1. **Guest OS Issues Instruction**
   Guest OS runs on virtual CPU. It executes privileged instruction (e.g., disk I/O, memory alloc).
2. **VM Trap to Hypervisor**
   Privileged instruction triggers a trap. Control switches from Guest OS to the Hypervisor.
3. **Hypervisor Intercepts**
   Hypervisor catches the trap, identifies the requested operation, checks permissions.
4. **Maps to Host Resources**
   Hypervisor translates virtual resource request to real hardware call (CPU scheduler, disk driver).
5. **Result Returned to Guest**
   Operation executes on physical hardware. Result is returned to Guest OS as if it ran natively.

KEY MECHANISMS ENABLING THIS:

Trap-and-Emulate

Privileged instructions trap to hypervisor. Used in full virtualization.

Hardware-Assisted Virt.

Intel VT-x / AMD-V: CPU has a special VMX mode to run VMs more efficiently.

Binary Translation

Hypervisor rewrites non-virtualizable instructions at runtime (older x86 CPUs).

Memory Ballooning

VMs lend unused RAM back to the host dynamically, improving utilization.

PROTOTYPE — JVM: The Classic Portability Solution

COMPILATION PIPELINE:

ONE BYTECODE → RUNS ON ANY PLATFORM:

🔑 The JVM is a PROCESS VM. Java programs interact with the JVM API, not directly with the OS or hardware. The JVM handles translation to each platform's native code.

COMPARATIVE ANALYSIS:

Approach	Performance	Isolation	Portability	Setup Cost	Use Case
Bare Metal (No VM)	★★★★★	None	★☆☆☆☆	Low	Gaming, Embedded
Type-1 Hypervisor (ESXi)	★★★★☆	Strong	★★★★☆	Med	Cloud Servers, DC
Type-2 Hypervisor (VBox)	★★★☆☆	Medium	★★★☆☆	Low	Dev, Testing
Process VM (JVM/Python)	★★★☆☆	Low	★★★★★	Low	Cross-platform Apps
Container (Docker)	★★★★☆	Medium	★★★★☆	Med	Microservices, CI/CD

⚠️ VM LIMITATIONS

- Overhead: VMs use extra CPU/RAM for the hypervisor layer
- Startup time: VMs boot like a real OS (minutes)
- Licensing: Many hypervisors cost money at enterprise scale

✅ VM BENEFITS

- Full isolation: one VM crash doesn't affect others
- Snapshots: save/restore VM state instantly
- Live Migration: move running VMs between physical hosts

INDUSTRY SCENARIO: Amazon Web Services (AWS EC2)

Cloud Infrastructure

AWS, Azure, GCP all built on massive Type-1 hypervisor farms. Millions of VMs run simultaneously on shared hardware — enabling the "rent a server" model.

Dev/Test Environments

Teams spin up fresh VMs per feature branch. When done, snapshot & delete. No more 'works on my machine' — CI/CD pipelines run on identical VM images.

Security Sandboxing

Antivirus software, browsers, and OS security features use VM isolation to run untrusted code. Windows WSL2 runs Linux inside a lightweight VM on Windows.

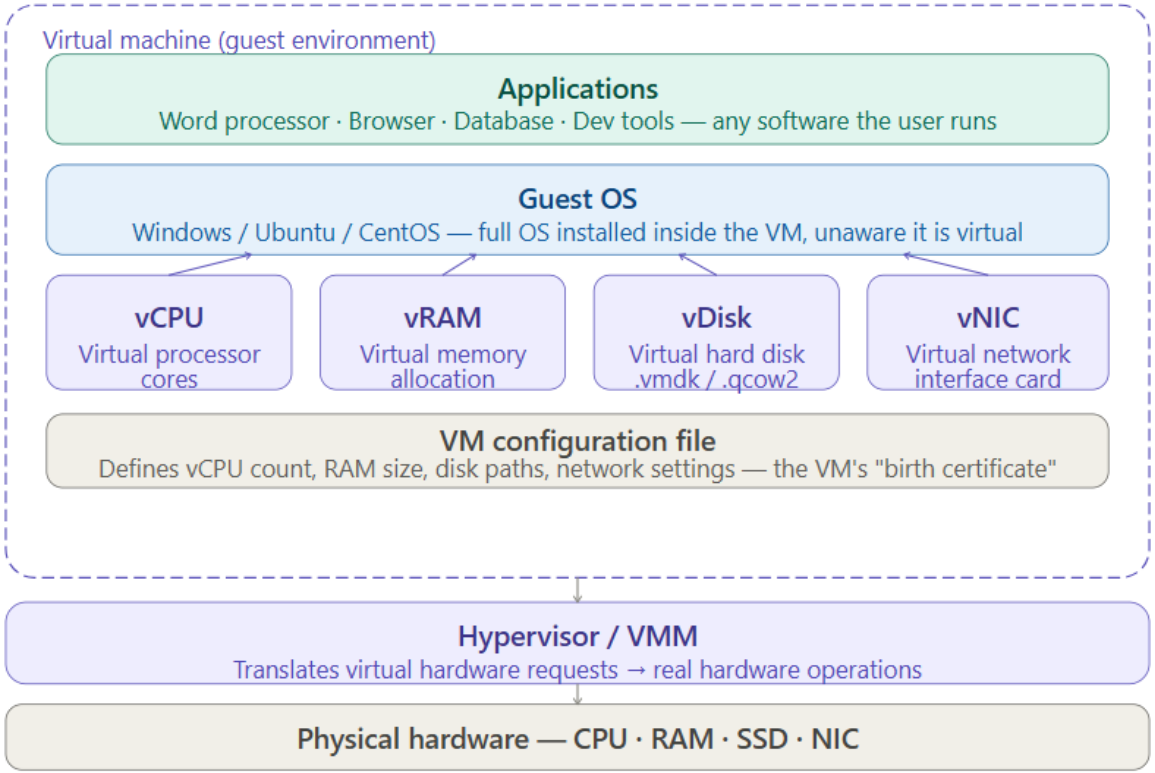
Legacy Preservation

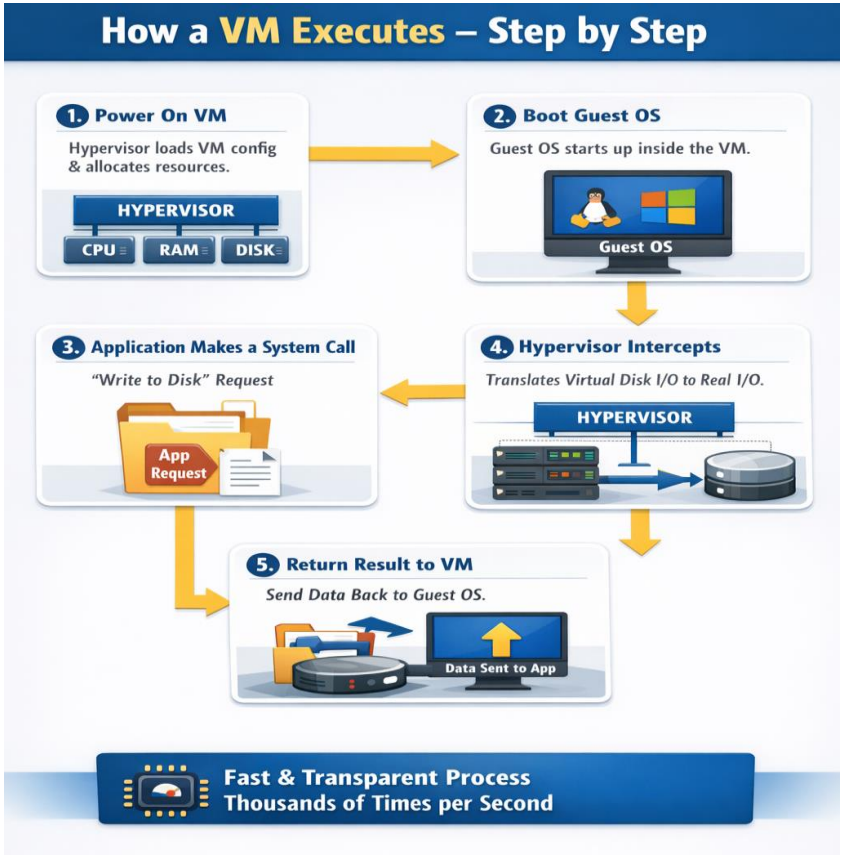
Old enterprise software from the 1990s still runs inside VMs on modern hardware. VMs preserve entire OS environments, avoiding costly rewrites of critical systems.

SUMMARY — From Problem to Solution

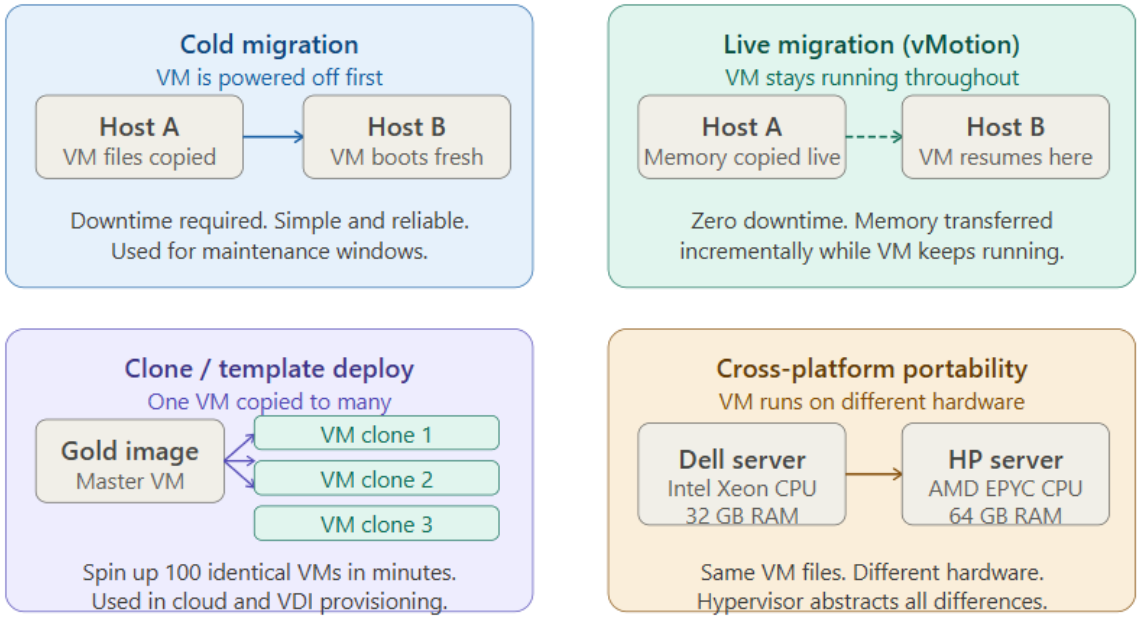
Empathize	Software fails across platforms. A program on Windows won't run on Linux without effort.
Define	Root causes: hardware diversity, OS API differences, resource conflicts, security isolation needs.
Ideate	Solutions: System VMs, Process VMs, Containers — each with different portability/performance balance.
Prototype	Hypervisors (Type-1/2), JVM bytecode pipeline, trap-and-emulate, hardware-assisted virtualization.
Test	VMs add overhead but give isolation, snapshots, live migration. Real world: AWS, Docker, WSL2.

Virtual machine — internal structure




How a VM Executes – Step by Step
1. Power On VM
Hypervisor loads VM config & allocates resources.
HYPERVISOR
CPU
RAM
DISK
2. Boot Guest OS
Guest OS starts up inside the VM.
Guest OS
3. Application Makes a System Call
"Write to Disk" Request
App Request
4. Hypervisor Intercepts
Translates Virtual Disk I/O to Real I/O.
HYPERVISOR
5. Return Result to VM
Send Data Back to Guest OS.
Data Sent to App
Fast & Transparent Process
Thousands of Times per Second

VM portability — four key scenarios



VM Portability — Advantages

Minutes, not days
Moving an entire server environment takes minutes instead of days of re-installation and reconfiguration.
Speed

No hardware vendor lock-in
Hardware vendor lock-in is eliminated — you are not tied to any specific manufacturer.
Flexibility

Risk-free testing via snapshots
Any damage can be reversed instantly by restoring a snapshot — test freely without fear.
Resilience

Fast scaling — clone 50 servers in minutes
Clone a template VM 50 times and you have 50 identical servers ready in minutes.
Scalability

Dramatically improved business continuity
Entire data centres can be replicated as VM files, enabling rapid disaster recovery anywhere.
Continuity

VM Portability — Disadvantages

Limitations — 3 key constraints

01 — Large VMs migrate slowly

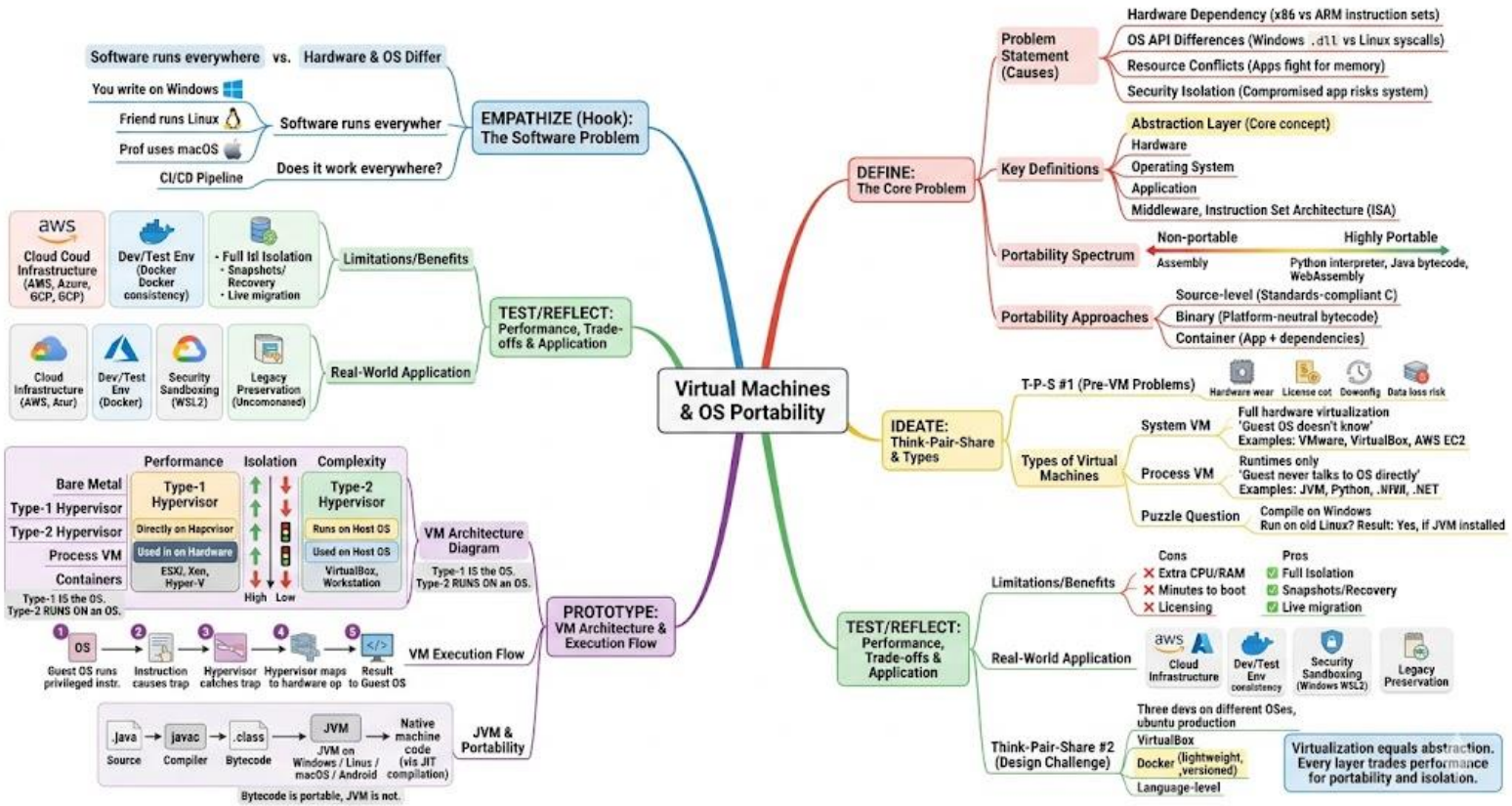
Very large VMs (with terabytes of disk data) take a long time to migrate even over fast networks.

02 — Cross-hypervisor conversion required

Moving between different hypervisor vendors (e.g. VMware → KVM) may require conversion tools like qemu-img or VMware Converter.

03 — Live migration needs shared high-speed infrastructure

Requires both hosts on the same fast network and ideally shared storage; otherwise memory copy takes too long and causes brief pauses.


Virtual Machines & OS Portability
EMPATHIZE (Hook): The Software Problem
Software runs everywhere
vs.
Hardware & OS Differ
You write on Windows
Friend runs Linux
Prof uses macOS
Software runs everywher
CI/CD Pipeline
Does it work everywhere?
TEST/REFLECT: Performance, Trade-offs & Application
Limitations/Benefits
Cloud Coud Infrastructure (AMS, Azure, 6CP, 6CP)
Dev/Test Env (Docker Docker consistency)
- Full Isl Isolation
- Snapshots/ Recovery
- Live migration
Real-World Application
Cloud Infrastructure (AWS, Azur)
Dev/Test Env (Docker)
Security Sandboxing (WSL2)
Legacy Preservation (Uncomonaned)
PROTOTYPE: VM Architecture & Execution Flow
VM Architecture Diagram
Performance
Isolation
Complexity
Bare Metal
Type-1 Hypervisor
Type-2 Hypervisor
Process VM
Containers
Directly on Hapcvisor
Used in on Hardware
ESXI, Xen, Hyper-V
Runs on Host OS
Used on Host OS
VirtualBox, Workstation
High
Low
Type-1 IS the OS. Type-2 RUNS ON an OS.
Type-1 IS the OS. Type-2 RUNS ON an OS.
VM Execution Flow
OS
Guest OS runs privileged instr.
Instruction causes trap
Hypervisor catches trap
Hypervisor maps to hardware op
Result to Guest OS
JVM & Portability
.Java
Source
javac
Compiler
.class
Bytecode
JVM
JVM on Windows / Linus / macOS / Android
Native machine code (vis JIT compilation)
Bytecode is portable, JVM is not.
DEFINE: The Core Problem
Problem Statement (Causes)
Hardware Dependency (x86 vs ARM instruction sets)
OS API Differences (Windows .dll vs Linux syscalls)
Resource Conflicts (Apps fight for memory)
Security Isolation (Compromised app risks system)
Key Definitions
Abstraction Layer (Core concept)
Hardware
Operating System
Application
Middleware, Instruction Set Architecture (ISA)
Portability Spectrum
Non-portable
Highly Portable
Assembly
Python interpreter, Java bytecode, WebAssembly
Portability Approaches
Source-level (Standards-compliant C)
Binary (Platform-neutral bytecode)
Container (App + dependencies)
IDEATE: Think-Pair-Share & Types
T-P-S #1 (Pre-VM Problems)
Hardware wear
License cot
Dowonfig
Data loss risk
Types of Virtual Machines
System VM
Full hardware virtualization
'Guest OS doesn't know'
Examples: VMware, VirtualBox, AWS EC2
Process VM
Runtimes only
'Guest never talks to OS directly'
Examples: JVM, Python, .NWII, .NET
Puzzle Question
Compile on Windows
Run on old Linux? Result: Yes, if JVM installed
TEST/REFLECT: Performance, Trade-offs & Application
Limitations/Benefits
Cons
Extra CPU/RAM
Minutes to boot
Licensing
Pros
Full Isolation
Snapshots/Recovery
Live migration
Real-World Application
Cloud Infrastructure
Dev/Test Env consistency
Security Sandboxing (Windows WSL2)
Legacy Preservation
Think-Pair-Share #2 (Design Challenge)
Three devs on different OSes, ubuntu production
VirtualBox
Docker (lightweight, versioned)
Language-level
Virtualization equals abstraction. Every layer trades performance for portability and isolation.

CONCEPT CHECK — Quick Puzzle

You compile a Java program on a Windows laptop. Your friend has an old Linux server from 2005.
Will your program run? WHY or WHY NOT? What assumption are you making?

A Yes — Java bytecode is platform-independent, so as long as the Linux server has a compatible JVM installed, it works regardless of OS version or hardware.

B No — Java programs only work on Windows because they were compiled there.

C Yes — all programs run on all computers by default.

D Only if you recompile on Linux — bytecode does not work cross-platform.