Week 3: Virtual Machine Technology

NT524 — Cloud Architecture and Security

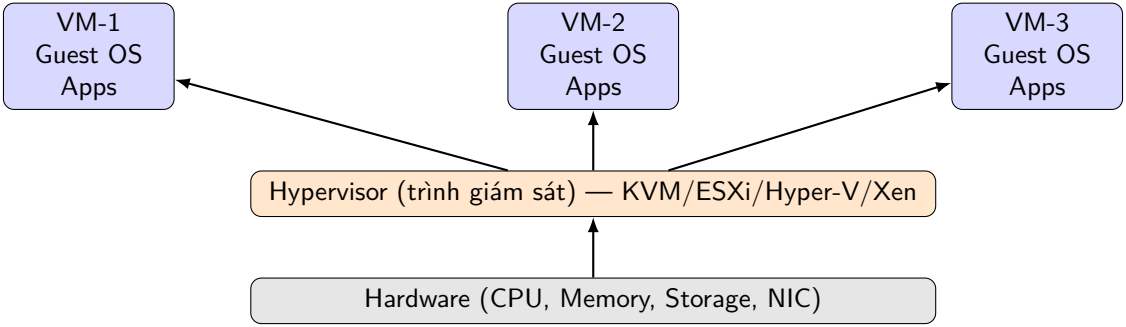
PhD. Nguyen Ngoc Tu

September 24, 2025

Learning Objectives (Mục tiêu)

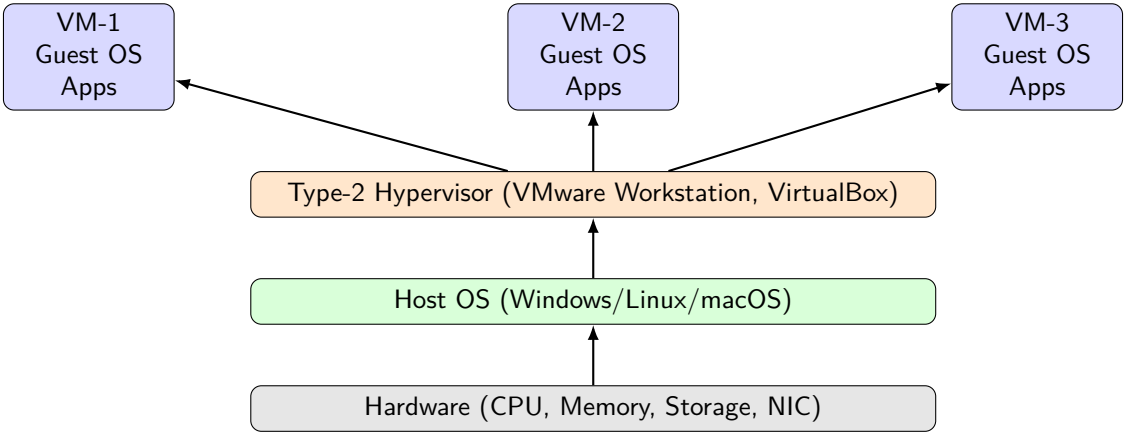
- Motivate virtualization for **resource utilization (tận dụng tài nguyên)** & **operational efficiency (vận hành hiệu quả)**.
- Understand hypervisor types (Type 1 vs. Type 2) and **where KVM fits** (kernel Type-1 via module).
- Master resource allocation: CPU scheduling & pinning, NUMA, hugepages, memory overcommit, I/O multiqueue, QoS.
- Manage VM lifecycle & image pipelines (Glance, cloud-init, hardening, **image signing/verification & governance**).
- Practice: Configure KVM in OpenStack with Ansible; automate VM deployment; **Placement traits/resources (PCPU vs VCPU)** for scheduling.

Type 1: Virtualization Stack (Ngăn xếp ảo hoá loại 1)



- **Device virtualization:** SR-IOV (Single Root I/O Virtualization/fast); virtio (paravirtualization/good); emulation (compatibility/slow).

Virtualization Stack (Type-2 Hypervisor)



- **Type-2 Hypervisor:** Runs *on top of a host OS*, easier to install but less efficient.
- Examples: `VirtualBox`, `VMware Workstation`, `Parallels`.

Type 1 Hypervisors (Hypervisor loại 1)

Definition (Định nghĩa): Run directly on hardware to manage CPU, memory, and I/O for Virtual Machines (VMs). (Chạy trực tiếp trên phần cứng để quản lý CPU, bộ nhớ và I/O cho máy ảo).

Key Insight (Điểm cốt lõi): Type 1 hypervisor can be:

- A **kernel-integrated module (mô-đun tích hợp trong nhân HĐH)** inside a general-purpose OS.
- A **special-purpose OS (hệ điều hành chuyên dụng)** designed only to manage hardware and VMs.

Examples (Ví dụ):

Hypervisor	Model	Note
KVM (Linux)	Kernel module (mô-đun nhân)	Linux kernel acts as hyperviso
Hyper-V (Windows)	Kernel-integrated (tích hợp nhân)	Windows root partition manag
VMware ESXi	Special-purpose OS (HĐH chuyên dụng)	Minimal OS, only for virtualiz
Xen	Small hypervisor + Dom0 OS	Dom0 helps manage devices

Takeaway (Kết luận): Type 1 = *"bare-metal control" (kiểm soát trực tiếp phần cứng)*, whether inside an OS kernel or as its own OS.

Type 2 Hypervisors (Hypervisor loại 2)

Definition (Định nghĩa): Run on top of a host operating system (HĐH chủ) and rely on it to access hardware. (Chạy trên HĐH chủ và phụ thuộc vào HĐH để truy cập phần cứng).

Key Insight (Điểm cốt lõi): Type 2 hypervisor is:

- An **application layer (tầng ứng dụng)** on top of the host OS.
- Uses host OS drivers for CPU, memory, and I/O access.

Examples (Ví dụ):

Hypervisor	Model	Note
VMware Workstation	Application (ứng dụng)	Dev/test on desktops
Oracle VirtualBox	Application (ứng dụng)	Cross-platform, free
Parallels Desktop	Application (ứng dụng)	Popular on macOS

Takeaway (Kết luận): Type 2 = *"hosted hypervisor" (hypervisor phụ thuộc HĐH)*, easier for dev/test but less performant than Type 1.

Hypervisor Types (Loại hypervisor)

- **Type 1 (bare metal – trực tiếp phần cứng):** ESXi, Hyper-V, **Linux+KVM (kernel module – mô-đun nhân)**, Xen. → High performance, production use (hiệu năng cao, dùng trong sản xuất).
- **Type 2 (hosted – phụ thuộc HĐH):** VMware Workstation, VirtualBox, Parallels. → Convenient for development/training (tiện cho phát triển/đào tạo).

Aspect (Khía cạnh)	**Type 1**	**Type 2**
Performance (Hiệu năng)	Near-native (gần như thật)	Depends on host OS (phụ thuộc HĐH)
Management (Quản trị)	API, Cluster, HA(High Availability)	Simple GUI (giao diện đồ hoạ)
Use-cases (Ứng dụng)	Data center, Cloud	Dev/test/lab (phát triển/thử nghiệm)
Hardware (Phần cứng)	Requires VT-x/AMD-V, IOMMU	Relies on host OS drivers (dùng driver HĐH)
Similarity (Điểm chung)	Both provide virtualization (cùng cung cấp ảo hoá)	

Warm-up Tasks (Nhiệm vụ khởi động)

In-class (Trong lớp):

1. **Windows Sandbox vs Linux VM**
   - **Windows Sandbox:** Bật qua `Control Panel` → `Programs` → `Turn Windows features on/off` → `Windows Sandbox`. Mỗi phiên là *ephemeral* (tạm thời), tự hủy khi đóng.
   - **Linux VM:** Tạo Ubuntu bằng VirtualBox/VMware hoặc WSL2. VM là *persistent* (lưu trạng thái): cần dùng **snapshots** (long-term storage) và kỷ luật quản lý để giữ môi trường sạch.
2. **Create Ubuntu 22 VM via Dashboard** (OpenStack / Azure / AWS) — hiểu rõ *ai* quản lý CPU, memory, I/O phía sau GUI.

WSL2 vs Windows Sandbox (Khác biệt lưu trữ)

WSL2 (Ubuntu 22.04 trên Windows)

- Filesystem nằm trong:
  `%USERPROFILE%\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu22.04LTS_`
- **Persistent (lưu giữ lâu dài)**: thay đổi được ghi vào `ext4.vhdx`, giữ nguyên sau reboot.
- Giống như VM disk thật sự (ổ đĩa ảo).

Windows Sandbox

- Storage layer nằm trong:
  `C:\ProgramData\Microsoft\Windows\Containers\ContainerStorages`
- **Ephemeral (tạm thời)**: mỗi lần mở Sandbox tạo instance mới từ base image, đóng lại thì layer bị xoá.
- Luôn khởi động ở trạng thái "sạch".

Takeaway (Kết luận)

- WSL2 $\Rightarrow$ lưu trữ bền (persistent) trong file `ext4.vhdx`.
- Sandbox $\Rightarrow$ cách ly tức thời, tự reset (ephemeral).

So sánh môi trường ảo hoá và Live OS (I)

Chức năng	**Live DVD/USB**	**Hyper-V (Wins)**	**OpenStack (KVM)**	**AWS / Azure**
Tên môi trường	Live DVD / Live USB	Windows VM / VHDX	OpenStack cloud VM	Public Cloud VM
Hypervisor Type (run on Host OS)	Không có	**Type 1** (Hyper-V)	**Type 1** (KVM qua kernel)	**Type 1** (AWS Nitro/KVM; Azure Hyper-V)
Boot OS	Có (boot trực tiếp trên phần cứng)	Có (guest OS)	Có (guest OS)	Có (guest OS)
Lưu dữ liệu run-time	RAM / tmpfs (*overlay* lưu bền tuỳ chọn)	RAM + VHDX (persistent)	RAM + đĩa ảo (ephemeral hoặc Cinder)	RAM + volume (EBS / Managed Disks)
Thay đổi dữ liệu OS	Có hạn chế (*đầy đủ* nếu bật overlay)	Đầy đủ	Đầy đủ	Đầy đủ
Snapshots / resize	Không	Có (checkpoint/resize)	Có (image/volume; *quiesce* khuyến nghị; resize + `growfs`)	Có (AMI/volume; resize + `growfs`)

So sánh môi trường ảo hoá và Live OS (II)

Chức năng	**Live DVD/USB**	**Hyper-V (Wins)**	**OpenStack (KVM)**	**AWS / Azure**
Mount trên host (chỉnh sửa)	Khó / ISO read-only	Dễ / read-write (mount VHDX trên Windows)	**Chỉ dễ khi tự quản hạ tầng** (nếu multi-tenant: snapshot → attach vào *rescue* VM)	**Gián tiếp:** snapshot → attach vào *helper* VM; hoặc console/remote
Cloud orchestration	Không	Có (Cluster/SCVMM)	Có (Nova, Heat, Placement, Cinder/Neutron)	Có (Auto Scaling/VMSS, managed services)

Ghi chú: *ephemeral* = xoá khi huỷ VM; *persistent* = đĩa/volume tồn tại sau khi huỷ VM. Snapshot ở VM đám mây là snapshot *volume/image*; ảnh hưởng nhất quán ứng dụng cần *quiesce*/agent.

Warm-up Tasks (Home Practice – Bài tập ở nhà)

1. **Use Cloud CLI (OpenStack/Azure/AWS):**

```
openstack server create --image ubuntu-22.04 --flavor m1.small myVM

az vm create --resource-group myRG --name myVM --image Ubuntu2204 \
  --admin-username azureuser --generate-ssh-keys

aws ec2 run-instances --image-id ami-xxxx --count 1 --instance-type t2.micro \
  --key-name myKey --security-groups mySG
```

2. **Automate with Ansible:** inventory + playbook cài Nginx, đảm bảo idempotent.

Glossary (Thuật ngữ)

- **Overcommit:** Cấp phát tổng tài nguyên ảo (vCPU/RAM/IOPS) lớn hơn tài nguyên vật lý, giả định VM không đồng thời đạt đỉnh.
- **Resource contention:** Nhiều VM cạnh tranh cùng CPU/RAM/I/O hữu hạn → giảm thông lượng hoặc tăng độ trễ.
- **Blast radius:** Phạm vi ảnh hưởng tối đa của một sự cố/tấn công; cách ly tốt giúp thu hẹp phạm vi này.
- **Provisioning:** Quá trình tạo và cấu hình tài nguyên (VM, storage, network) sẵn sàng cho workload.
- **Snapshot/Clone:** Snapshot = ảnh chụp trạng thái hệ thống tại một thời điểm; Clone = nhân bản một bản sao từ snapshot/image.
- **Live migration:** Di chuyển VM giữa các host mà không tắt máy, giúp bảo trì hạ tầng.
- **IaC (Infrastructure as Code):** Mô tả hạ tầng bằng mã/script, có thể quản lý như phần mềm.
- **Isolation (Cách ly):** Thuộc tính **miền bảo vệ** trong ảo hoá: đảm bảo mã và dữ liệu của một VM/tenant không thể truy cập hoặc gây gián đoạn tới VM/tenant khác, nhờ hypervisor và cơ chế phần cứng (EPT/NPT, IOMMU/VT-d, phân quyền CPU).

Motivation (Động lực) I

- **Consolidation (Hợp nhất):** Chạy nhiều workload/VM trên **máy chủ vật lý** để tăng mức sử dụng tài nguyên.
  - *Ưu điểm:* Tăng mật độ VM; tiết kiệm phần cứng, điện, làm mát; giảm footprint trung tâm dữ liệu; tận dụng tài nguyên nhàn rỗi.
  - *Thách thức:* **Resource contention** khi **overcommit**; hiệu năng/độ trễ khó dự đoán cho workload nhạy cảm; nguy cơ "noisy neighbor"; cần giám sát & capacity planning.

- **Isolation (Cách ly):** Cơ chế cách ly theo miền bảo vệ (protection domain) giúp một VM/tenant không thể đọc, ghi hoặc làm suy giảm tính sẵn sàng của VM/tenant khác.
  - *Ưu điểm:* Thu hẹp **blast radius**; bảo vệ tính bí mật, toàn vẹn và sẵn sàng (CIA); hỗ trợ đa tenant an toàn; đáp ứng tuân thủ.
  - *Thách thức:* Nguy cơ **side-channel** (cache, SMT); **co-residency** trên cùng host; cấu hình sai với thiết bị SR-IOV làm yếu cách ly; lỗ hổng hypervisor dẫn tới **VM escape**; live migration cần kênh truyền an toàn.

Motivation (Động lực) II

- **Agility (Linh hoạt):** Khả năng triển khai, điều chỉnh hoặc di chuyển workload nhanh chóng trong hạ tầng ảo hoá/đám mây.
  - *Ưu điểm:* Tạo VM chỉ trong vài phút; snapshot/clone nhanh; *live migration* hỗ trợ bảo trì không gián đoạn; thúc đẩy đổi mới.
  - *Thách thức:* Live migration tiêu tốn băng thông và I/O; snapshot/clone phình dữ liệu; rủi ro khi không kiểm soát phiên bản; phụ thuộc controller/API $\rightarrow$ có thể trở thành single point of failure.

- **Operational Efficiency (Hiệu quả vận hành):** Tự động hoá, API và giám sát giúp giảm công việc thủ công, giảm lỗi và tối ưu tài nguyên.
  - *Ưu điểm:* IaC giúp tái lập cấu hình nhanh chóng, nhất quán; automation giảm rủi ro lỗi con người; dễ mở rộng/thu hẹp (scaling); observability nâng cao hiệu năng.
  - *Thách thức:* Cần đầu tư công cụ (Ansible, Terraform, Prometheus...); chuẩn hoá CI/CD; khó tích hợp đa nền tảng; nguy cơ lỗi lan rộng khi automation sai.

Resource Allocation Fundamentals I: CPU (Bộ xử lý)

How to Manage (Cách quản lý)

- **vCPU Scheduling:** Hypervisor ánh xạ vCPU → pCPU theo timeslice.
- **Overcommit:** Cấp phát vCPU nhiều hơn pCPU để tăng mức sử dụng.
- **SMT / Hyper-Threading:** Một core vật lý hiển thị thành 2 CPU logic.
- **vCPU Pinning:** Gắn vCPU cố định với pCPU để hiệu năng ổn định.
- **Emulatorpin:** Ghim luồng QEMU/KVM I/O riêng để giảm jitter.

Issues (Các vấn đề)

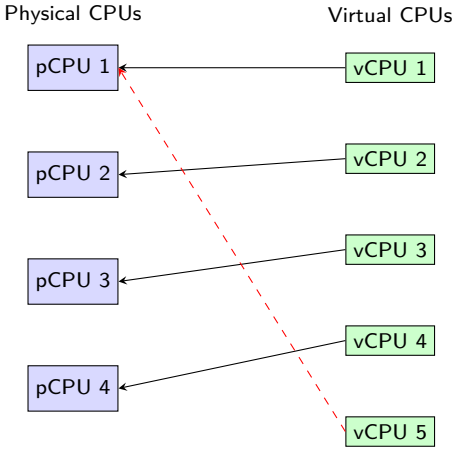
- **vCPU Scheduling:** Có thể gây *latency* khi nhiều VM tranh CPU.
- **Overcommit:** Dễ gây *resource contention* (tranh chấp tài nguyên).
- **SMT / Hyper-Threading:** Nguy cơ lỗ hổng side-channel (Spectre, Meltdown).
- **vCPU Pinning:** Giảm tính linh hoạt trong cân bằng tải.
- **Emulatorpin:** Cần tinh chỉnh thủ công, khó quản lý ở quy mô lớn.

Terminology

Resource Allocation Fundamentals I: CPU (Bộ xử lý)

- **vCPU Scheduling (Lập lịch vCPU):** Mapping vCPUs → pCPUs.
- **Overcommit (Phân bổ quá mức):** vCPU > pCPU, nguy cơ *resource contention*.
- **SMT / Hyper-Threading (Siêu phân luồng):** 1 pCPU = 2 vCPU logic.
- **vCPU Pinning (Ghim vCPU):** Cố định vCPU → pCPU.
- **Emulatorpin (Ghim luồng mô phỏng):** Tách I/O thread QEMU/KVM khỏi workload vCPU.

Terminology: *Resource contention* = tranh chấp tài nguyên, *Overcommit* = phân bổ quá mức, *Pinning* = ghim CPU.



Resource Allocation Fundamentals II: Memory (Bộ nhớ)

How to Manage (Cách quản lý)

- **Ballooning:** Guest driver trả lại RAM cho hypervisor khi host thiếu bộ nhớ.
- **KSM (Kernel Same-page Merging):** Hợp nhất các trang bộ nhớ giống nhau giữa nhiều VM để tiết kiệm RAM.
- **Hugepages:** Dùng trang lớn (2MB/1GB) để giảm TLB miss, tối ưu cho DB/ứng dụng nhạy cảm độ trễ.
- **NUMA Locality:** Gán bộ nhớ VM vào cùng NUMA node với vCPU để giảm độ trễ.

Issues (Các vấn đề)

- **Ballooning:** Có thể gây *swap* trong guest $\rightarrow$ hiệu năng kém.
- **KSM:** Overhead CPU cho việc so sánh trang; nguy cơ rò rỉ side-channel.
- **Hugepages:** Thiếu linh hoạt, cần pre-allocation; khó khi VM động (dynamic resize).
- **NUMA Locality:** Sai cấu hình $\rightarrow$ cross-node access gây *latency*.

Resource Allocation Fundamentals II: Memory (Bộ nhớ)

Terminology:
Ballooning = bơm bộ nhớ, KSM = hợp nhất trang kernel, Hugepages = trang nhớ lớn, NUMA locality = tính cục bộ của kiến trúc NUMA.

Resource Allocation Fundamentals IIIa: I/O Virtualization (Ảo hoá I/O)

- **virtio-blk/scsi:** Thiết bị block paravirtualized để tăng hiệu năng.
- **Multiqueue:** Nhiều hàng đợi cho thiết bị mạng/lưu trữ; mở rộng theo số lượng vCPU.
- **vhost-net / vhost-user:** Đường tăng tốc trong kernel hoặc user-space cho virtio-net.
- **SR-IOV (Single Root I/O Virtualization):** Gán trực tiếp thiết bị cho VM để đạt hiệu năng cao.
  - **Live migration:** Khả thi với *representor/vDPA/indirect* (phụ thuộc NIC/driver).
  - **Direct mode** thường không tương thích với LM.

Terminology:
Virtio = ảo hóa bán phần (paravirtual), SR-IOV = ảo hoá I/O gốc đơn.

Resource Allocation Fundamentals IIIb: Quality of Service (QoS – Chất lượng dịch vụ)

- **IOPS / MBps limits:** Giới hạn băng thông và số thao tác lưu trữ mỗi giây.
- **CPU shares / quotas:** Điều khiển mức độ công bằng CPU giữa các tenant.
- **Network rate limiting:** Giới hạn băng thông mỗi vNIC.

Terminology:
QoS = chất lượng dịch vụ.

NUMA Basics (Cơ bản về NUMA)

- **NUMA (Non-Uniform Memory Access) – Kiến trúc bộ nhớ không đồng nhất:** Each CPU socket has its own local memory region (node). Mỗi socket CPU có vùng bộ nhớ cục bộ riêng (node).
- **Local vs Remote Access (Truy cập cục bộ vs từ xa):** Accessing local memory has lower latency and higher bandwidth. Truy cập bộ nhớ cục bộ có độ trễ thấp và băng thông cao. Accessing remote memory across sockets increases latency. Truy cập bộ nhớ từ xa giữa các socket làm tăng độ trễ.
- **NUMA Awareness (Hiểu biết NUMA):** Workloads sensitive to latency (databases, NFV) should be placed entirely in one NUMA node. Các workload nhạy cảm với độ trễ (CSDL, NFV) nên đặt gọn trong một NUMA node.

Terminology (Thuật ngữ):
NUMA = kiến trúc bộ nhớ không đồng nhất, Node NUMA = cụm CPU + bộ nhớ cục bộ, Remote memory access = truy cập bộ nhớ từ xa.

Pinning & NUMA Policies (Ghim CPU/Bộ nhớ & Chính sách)

- **vCPU Pinning (Ghim vCPU):** Assign virtual CPUs to specific physical CPUs for predictable performance. Gán vCPU vào pCPU cụ thể để có hiệu năng ổn định.
- **Memory Pinning (Ghim bộ nhớ):** Allocate VM memory from the same NUMA node as its pinned vCPUs. Gán bộ nhớ VM cùng NUMA node với các vCPU được ghim.
- **Emulator Thread Isolation (Cô lập luồng mô phỏng):** Place QEMU I/O threads (*emulator threads*) on different cores from application vCPUs. Đặt luồng QEMU/I/O trên core khác với core ứng dụng để tránh jitter.
- **Policies (extra_specs in OpenStack flavors):**
  - `hw:cpu_policy=dedicated` – cấp phát CPU vật lý riêng (không chia sẻ).
  - `hw:numa_nodes=1` – ép VM vào một NUMA node duy nhất.
  - `hw:mem_page_size=large/1GB` – yêu cầu hugepages (2MB/1GB).

Terminology (Thuật ngữ):
vCPU pinning = ghim CPU ảo vào CPU vật lý, Emulator threads = luồng mô phỏng (QEMU/I/O), Flavor extra_specs = chính sách cấu hình flavor trong OpenStack.

VM Lifecycle (Vòng đời VM) I

- **Lifecycle Phases (Các giai đoạn vòng đời): Create (Tạo) → Configure (Cấu hình) → Start/Stop/Suspend (Khởi động/Dừng/Tạm dừng) → Snapshot/Clone (Ảnh chụp/Nhân bản) → Resize (Thay đổi cấu hình) → Live Migrate (Di chuyển trực tiếp) → Retire (Ngừng sử dụng)**.
- **Snapshot & Backup (Ảnh chụp và Sao lưu):**
  - *Crash-consistent (nhất quán khi sự cố):* dữ liệu giống như khi máy bị tắt đột ngột.
  - *App-consistent (nhất quán ứng dụng, quiesce):* dừng tiến trình I/O tạm thời → an toàn hơn.
  - *Incremental (gia tăng):* chỉ lưu thay đổi kể từ snapshot trước, tiết kiệm dung lượng.
- **Live Migration (Di chuyển trực tiếp):**
  - *Pre-copy:* copy bộ nhớ dần dần rồi cắt sang host mới.
  - *Post-copy:* chuyển VM nhanh, load page từ xa khi cần.
  - *Auto-converge:* giảm tốc độ VM tạm thời để kịp đồng bộ.
  - Yêu cầu *shared storage* hoặc block-migration; CPU mode (host-model vs host-passthrough) ảnh hưởng tương thích.

VM Lifecycle (Vòng đời VM) II – Cloud Examples

- **OpenStack:**
  - *Create:* `openstack server create -image ubuntu -flavor m1.small myVM`.
  - *Snapshot:* `openstack server image create myVM -name snap1`.
  - *Resize:* `openstack server resize myVM -flavor m1.medium`; confirm với `openstack server resize -confirm`.
  - *Live migrate:* `openstack server migrate -live <dest-host> myVM`.
- **AWS EC2:**
  - *Snapshot:* tạo EBS snapshot → `aws ec2 create-snapshot -volume-id vol-123`.
  - *Clone/AMI:* `aws ec2 create-image -instance-id i-123 -name myAMI`.
  - *Resize:* thay đổi *instance type* → `aws ec2 modify-instance-attribute -instance-type m5.large`.
  - *Live migrate:* **provider-managed** during maintenance; *Dedicated Hosts* support platform live migration; no user-triggered LM.
- **Azure VM:**
  - *Snapshot:* `az snapshot create -source myDisk -name snap1`.
  - *Clone:* tạo managed image từ VM → deploy VM mới.
  - *Resize:* `az vm resize -resource-group RG1 -name MyVM -size Standard_DS3_v2`.
  - *Live migrate:* Azure thực hiện *maintenance live migration* (memory-preserving) để giảm downtime, không có lệnh trực tiếp từ người dùng.

Image Management I.a: Formats (Định dạng ảnh)

- `qcow2` – thin provisioning, snapshots (OpenStack default).
- `raw` – fastest I/O, large size (no compression).
- `vmdk` – VMware format.
- `vhd/vhdx` – Microsoft Hyper-V, Azure.
- `AMI` (Amazon Machine Image) – AWS-specific wrapper around snapshots/volumes.

Terminology (Thuật ngữ):
qcow2 = định dạng ảnh nén + snapshot, raw = định dạng thô.

Image Management I.b: Metadata & Cloud-init (Siêu dữ liệu & Khởi tạo)

- **Metadata (Siêu dữ liệu):** `hw_disk_bus`, `hw_vif_model`, `os_distro`, checksum, version, license info.
- **Cloud-init (Khởi tạo đám mây):**
  - *user_data* (YAML/bash scripts), SSH keys, hostname, network config.
  - Option: **Config-Drive** for stronger isolation than metadata service.

Terminology (Thuật ngữ):
cloud-init = công cụ khởi tạo VM bằng metadata/userdata, Config-Drive = ổ ảo chứa metadata an toàn.

Image Management II.a: Multi-Cloud Locations & Management

- **Locations (Vị trí lưu trữ):**
  - **OpenStack:** Glance image store (file, Swift, Ceph RBD, NFS).
  - **AWS:** S3-backed AMIs, linked to EBS snapshots; stored regionally.
  - **Azure:** Managed Images or Shared Image Gallery; stored in Blob Storage; supports replication across regions.
  - **Multi-Cloud:** Convert images with `qemu-img` between `qcow2/raw/vhd/vmdk`.
- **Management (Quản lý):**
  - **Versioning:** Tag images (v1, v2) for reproducibility.
  - **Hardening:** Apply CIS benchmarks, security updates.
  - **Signing/Verification:**
    - **OpenStack:** Glance + Barbican can cryptographically sign/verify images.
    - **AWS:** Governance controls — AMI sharing restrictions (Block Public AMIs), *Allowed AMIs* via Organizations/SCP, EC2 Image Builder pipelines; *no native boot-time signature check*.
    - **Azure:** Trusted Launch (vTPM/Secure Boot), validation features in SIG.
  - **Sharing:** Publish to tenants (OpenStack), share AMIs (AWS), share to subscriptions/SIG (Azure).

Terminology (Thuật ngữ):
Glance = dịch vụ lưu trữ ảnh OpenStack, AMI = ảnh máy ảo AWS, Azure SIG = thư viện ảnh chia sẻ Azure.

Image Management II.b: Pipeline (Chuỗi công việc)

- `Packer build`
- → *OS/agent installation*
- → *CIS hardening (bảo mật)*
- → **Sign/Encrypt or Govern**
- → Publish to registry:
  - **OpenStack:** Glance
  - **AWS:** AMI (S3/EBS backed)
  - **Azure:** Shared Image Gallery

Terminology (Thuật ngữ):
Packer = công cụ tự động build ảnh.

Industry Perspective — Survey Highlights

- **Enterprise strategy (Chiến lược DN):**
  - Standardize on a Type-1 stack (KVM/ESXi/Hyper-V); golden images; policy-driven flavors/traits (OpenStack Placement) and image catalogs.
  - Security baselines: CIS hardening, **signed/verified images in OpenStack (Glance+Barbican)**, Trusted/Measured boot (Azure Trusted Launch), and Nitro protections on AWS.
- **Consolidation vs Performance (Hợp nhất vs Hiệu năng):**
  - Overcommit for general workloads; **dedicated pCPUs + NUMA pinning + hugepages** for DB/NFV/latency-sensitive. Vendor guidance favors "fit within one NUMA node" when possible.
  - **SR-IOV:** low-latency/throughput; **LM feasible with representor/vDPA/indirect** in some stacks; **direct** mode typically reduces or prevents mobility.
- **Compliance & Risk (Tuân thủ & rủi ro):**
  - Audit trails (build/migrate/snapshot), image provenance (sign/verify or governance), patch cadence, backup/DR drills.
  - Isolation hardening: EPT/NPT, IOMMU/VT-d, vTPM/Secure Boot; monitor side-channels on SMT; encrypted or TLS-protected live migration channels.

Industry Perspective — Survey Highlights

- **Platform specifics (Theo nền tảng):**
  - **OpenStack:** Glance image signing/verification with Barbican; Placement traits/resources; **SR-IOV LM possible with representor/vDPA in some setups**.
  - **AWS:** Nitro System (KVM-based lightweight hypervisor) for isolation/perf; AMIs as image unit; provider-managed maintenance migrations.
  - **Azure:** Trusted Launch (vTPM, Secure Boot, attestation) for VM integrity; Shared Image Gallery for controlled distribution.

Industry Perspective — Selected References (Tài liệu tham khảo)

- AWS — *Security Design of the AWS Nitro System* (whitepaper/overview).
- Azure — *Trusted Launch for Azure VMs* (vTPM, Secure Boot, attestation).
- OpenStack — *Glance Image Signature Verification* (with Barbican); *Nova certificate validation example*.
- OpenStack/Red Hat — NUMA tuning guidance (keep a VM within one NUMA node when possible).
- OpenStack/Red Hat/Neutron — SR-IOV live migration design/limitations.
- VMware — *vSphere 8 Performance Best Practices / Virtual Topology* (NUMA-aware placement).

(See course site for links or scan QR on this slide.)

Hypervisor Market (So sánh nền tảng) I

Criteria (Tiêu chí)	VMware vSphere	Microsoft Hyper-V	KVM / Xen
Features (Tính năng)	HA/DRS, vMotion, vSAN, NSX	Failover Clustering, SCVMM	Open-source stack (libvirt, oVirt, OpenStack)
Cost (Chi phí)	License/subscription; premium support	Included with Windows Server	Open-source license, higher staffing/skills cost
Performance (Hiệu năng)	Strong, highly optimized stack	Good, Windows integration	Excellent with tuning (hugepages, NUMA pinning, SR-IOV)
Use-cases (Ứng dụng)	Commercial enterprises	Windows-centric shops	Telco, cloud, NFV, private cloud
Ecosystem (Hệ sinh thái)	Broad enterprise ecosystem	Microsoft ecosystem	Ceph, OVS/OVN, oVirt/OKD/virt-operator

Hypervisor Market (So sánh nền tảng) II — Industry Trends

- **VMware vSphere:** Still dominant in large enterprises (finance, government) due to mature features, but adoption slowing in cloud-native environments; cost is a major factor.
- **Hyper-V:** Strong in organizations already standardized on Windows Server/Active Directory. Azure Stack HCI extends Hyper-V into hybrid scenarios.
- **KVM:** Default hypervisor for OpenStack, Red Hat OpenShift Virtualization, and public clouds (AWS Nitro, Google Cloud, Oracle Cloud).
- **Xen:** Historically used in AWS and Citrix; now mostly legacy, with AWS moving fully to Nitro/KVM.
- **Survey Data (2023–2024):**
  - VMware share dropping in telco/cloud, replaced by KVM-based platforms.
  - AWS, Azure, GCP all run on KVM or Hyper-V variants under the hood.
  - Enterprises often adopt hybrid: VMware on-prem + KVM/OpenStack in cloud labs.

References: VMware vSphere 8 Performance Best Practices; AWS Nitro System Security Whitepaper; Azure Stack HCI overview; OpenStack Foundation reports (2023–2024); Red Hat virtualization/NUMA tuning guides.

Operational Challenges I (Thách thức kỹ thuật)

- **VM sprawl / zombie (VM dư thừa/không quản lý):** Too many VMs without clear owner or purpose. ⇒ Apply lifecycle policy, tagging, chargeback/showback. *Example:* In OpenStack, use `project quotas`; in AWS, enforce tags with Config rules; in Azure, use Resource Locks & Cost Management.
- **Noisy neighbor (Láng giềng ồn ào):** Shared CPU/I/O resources cause unpredictable latency. ⇒ QoS policies, resource isolation, dedicated flavors. *Example:* AWS *Dedicated Host*; OpenStack flavor with `hw:cpu_policy=dedicated`; Azure *Isolated VM sizes*.
- **NUMA mismatch (Sai lệch NUMA):** vCPU topology exceeds NUMA boundaries ⇒ performance loss. ⇒ Pinning, NUMA hints, flavor extra_specs. *Example:* OpenStack flavor `hw:numa_nodes=1`; VMware NUMA scheduler; Azure HPC VM sizes pre-aligned to NUMA.

Operational Challenges II (Di trú, trôi dạt, đa đám mây)

- **Migration risk (Rủi ro di trú):** CPU feature mismatch breaks live migration. ⇒ Use compatible CPU mode (`host-model` vs `host-passthrough`). *Example:* OpenStack live-migrate checks CPU features; AWS/Azure hide migration (done transparently by provider).
- **Drift (Trôi dạt cấu hình):** Image drift (patch levels differ), kernel ABI drift (guest tools mismatch). ⇒ Infrastructure as Code (IaC), golden images, policy-as-code. *Example:* AWS AMI lifecycle policy, Azure Shared Image Gallery versioning, OpenStack Glance image catalog with signed images.
- **Multi-cloud consistency (Nhất quán đa đám mây):** Different hypervisors, image formats, and metadata services complicate ops. ⇒ Use common build pipelines (Packer), image conversion (`qemu-img`), and centralized governance. *Example:* Publish one golden image to Glance (OpenStack), AMI (AWS), and SIG (Azure).

Performance Considerations I (Cross-Hypervisor CPU/Memory)

- **CPU (Bộ xử lý):**
  - *VMware ESXi:* monitor **ready time**; use DRS/NUMA scheduler for balance; pin latency-sensitive VMs if needed.
  - *KVM (Linux):* monitor **steal time** and run queues; use `vCPU pinning` + isolate emulator threads (`emulatorpin`); prefer `host-model` CPU mode for compatibility, `host-passthrough` for perf.
  - *Hyper-V:* leverage CPU reservations, caps, and relative weights; NUMA spanning off for DB workloads.
  - *Hardware:* Intel Hyper-Threading (SMT) vs AMD SMT: throughput gains but potential side-channels; ARM often no SMT, lower leakage.
- **Memory (Bộ nhớ):**
  - *Hugepages (Trang lớn):* 2MB/1GB pages reduce TLB misses (good for DB, NFV).
  - *Ballooning/KSM:* save memory but can add jitter; disable for RT/DB workloads.
  - *Swappiness (Linux):* set low for DB/RT workloads; VMware uses transparent page sharing (TPS) but off by default for security.
  - *Hardware NUMA:* pin memory to local NUMA nodes; Intel vs AMD EPYC differences (chiplet design $\rightarrow$ more NUMA nodes).

Performance Considerations II (Cross-Hypervisor Network/Storage)

- **Network (Mạng):**
  - *Virtio-net (KVM):* use multiqueue and vhost-net/vhost-user for scale.
  - *VMware:* VMXNET3 paravirtual NIC for low overhead; supports RSS for multiqueue.
  - *Hyper-V:* use Synthetic NICs (vs Legacy); SR-IOV for HPC/latency-sensitive.
  - *Hardware:* Intel SR-IOV NICs mature; AMD Pensando/DPUs emerging; offload TLS/IPsec to SmartNICs.
- **Storage (Lưu trữ):**
  - *Virtio-scsi (KVM):* multiqueue; `cache=none` for DB; dedicate iothread per-disk.
  - *VMware:* use Paravirtual SCSI (PVSCSI) adapter for high IOPS; align VMFS blocks.
  - *Hyper-V:* use VHDX format with fixed size for perf; Offloaded Data Transfer (ODX) for SAN.
  - *Hardware:* NVMe SSDs (low latency) vs SATA/SAS; Intel Optane persistent memory for DB; ARM servers often optimized for NVMe storage.

Security Considerations I (Native Cloud VM Challenges)

- **Side-channels (Kênh bên):** Spectre/Meltdown/MDS/L1TF still relevant in multi-tenant clouds. Trade-off: enabling mitigations $\Rightarrow$ performance overhead; policies on *overcommit* and *co-residency* matter more in cloud scale.
- **Encrypted Live Migration (Mã hoá di trú trực tiếp):** Use TLS between libvirt daemons (OpenStack) or provider-managed encrypted migration (AWS Nitro, Azure Live Migration). Requires cert validation to prevent MitM during migration.
- **Confidential Computing (Điện toán bảo mật):** AMD SEV / SEV-ES / SEV-SNP, Intel TDX, Azure Confidential VMs, Google Confidential VMs. Encrypts guest memory against hypervisor or cloud operator access.

Security Considerations II (Cloud-Native Specific Risks)

- **Metadata/IMDS Abuse (Lạm dụng dịch vụ metadata):** Instance Metadata Service (IMDS) attacks → credential theft via SSRF. Mitigations: IMDSv2 (AWS), Azure Managed Identity, OpenStack Config-Drive, rate limiting, hardened proxy.
- **Control Plane Exposure (Bề mặt điều khiển):** Cloud APIs are the real attack surface (not just hypervisor). Risks: weak IAM policies, token leakage, excessive privileges. Mitigations: least privilege, API rate limiting, logging/auditing.
- **File Sharing & Guest/Host Boundary (Chia sẻ file & ranh giới):** virtio-fs/9p for file sharing can expand trust boundary. Mitigations: SELinux/AppArmor confinement, strict mount namespaces, use only within same tenant.
- **Supply Chain of Images (Chuỗi cung ứng ảnh):** Risks: poisoned base images, drift across tenants. Mitigations: **signed images (Glance + Barbican)**, **AWS governance** (Block Public AMIs/Allowed AMIs/Image Builder), **Azure** Trusted Launch & Compute Gallery controls.

Threat Model Lite (STRIDE) cho VM

- **Spoofing (Giả mạo):** Metadata service spoofing, token/credential theft. ⇒ IMDSv2, Config-Drive, mTLS, IAM least privilege.
- **Tampering (Sửa đổi trái phép):** Altering images, flavors, or control-plane configs. ⇒ Signed images, checksum, RBAC on image publish, IaC with policy enforcement.
- **Repudiation (Phủ nhận hành vi):** Lack of audit trails for VM actions (boot, migrate, snapshot). ⇒ Enable hypervisor/compute/nova logs, integrate with CloudTrail / Azure Activity Logs / Keystone audit.
- **Information Disclosure (Rò rỉ thông tin):** Snapshot or volume leaks, multi-tenant isolation failure. ⇒ Encrypt volume/snapshot, enforce tenant isolation (SR-IOV caution, vTPM).
- **DoS (Từ chối dịch vụ):** Overcommit + noisy neighbor; API flooding. ⇒ Quotas, QoS, admission control, API rate limiting, reserve headroom.
- **Elevation of Privilege (Leo thang đặc quyền):** SR-IOV/nested VM abuse; hypervisor escape; misconfigured IAM roles. ⇒ Strict policy, privilege review, timely patching, disable unneeded features.

Practice Overview: OpenStack (Thực hành với OpenStack)

- **Task 1:** Configure KVM/Libvirt on compute nodes; edit `nova.conf [libvirt]` to set `virt_type=kvm`.
- **Task 2:** Create flavors with **CPU pinning/hugepages/NUMA**; assign Placement traits/resources.
- **Task 3:** Automate VM provisioning: upload signed image to Glance, define network, keypair, and cloud-init user-data (optionally via **Config-Drive**).
- **Task 4:** Image lifecycle: versioning, set properties, test snapshot and restore.

Practice Overview: AWS EC2 (Thực hành với AWS)

- **Task 1:** Select AMI (Ubuntu 22.04 LTS or hardened AMI) and choose instance type (e.g., `t2.micro`, `m5.large`).
- **Task 2:** Apply instance attributes: vCPUs, RAM, storage (EBS), placement groups (for NUMA-aware HPC).
- **Task 3:** Automate deployment using `aws ec2 run-instances` with key pair, security group, and user-data script.
- **Task 4:** Image management: create AMI from running instance (`create-image`), snapshot EBS volumes, version images, share AMIs across accounts.

Practice Overview: Azure VM (Thực hành với Azure)

- **Task 1:** Provision VM from Marketplace image (Ubuntu 22.04 LTS) or custom image in Shared Image Gallery (SIG).
- **Task 2:** Choose VM size (Standard DSv3, HBv3 for HPC) $\rightarrow$ NUMA-aligned for performance-sensitive workloads.
- **Task 3:** Automate deployment with `az vm create`, include SSH keys, NSG (network security group) rules, and cloud-init in `-custom-data`.
- **Task 4:** Manage images: capture VM into Managed Image, version in Shared Image Gallery, replicate across regions, test snapshot/restore.

Ansible: KVM Enablement & Nesting

```
- hosts: computes
  become: true
  tasks:
    - name: Ensure KVM modules are present (Intel)
      modprobe:
        name: "{{ item }}"
        state: present
      loop: ["kvm", "kvm-intel"]

    - name: Enable nested virtualization (Intel)
      copy:
        dest: /etc/modprobe.d/kvm-intel.conf
        content: 'options kvm-intel nested=1'

    - name: Enable nested virtualization (AMD) # if AMD hosts
      copy:
        dest: /etc/modprobe.d/kvm-amd.conf
        content: 'options kvm-amd nested=1'

    - name: Set libvirt qemu user/group
      lineinfile:
```

Ansible: Scheduler via Placement (Modern)

```
- hosts: controller
  tasks:
    - name: Create a latency-sensitive flavor (dedicated pCPUs + hugepages)
      openstack.cloud.flavor:
        state: present
        name: m1.latency
        ram: 8192
        vcpus: 4
        disk: 40
        extra_specs:
          "resources:PCPU": "4" # placement pCPUs
          "trait:CUSTOM_NUMA_AWARE": "required"
          "hw:cpu_policy": "dedicated"
          "hw:numa_nodes": "1"
          "hw:mem_page_size": "large" # or "1GB" if hosts provisioned
          "hw:scsi_model": "virtio-scsi"
          "hw:disk_bus": "scsi"
```

Ansible: Image Upload (Signed) & Properties

```
- hosts: controller
  tasks:
    - name: Upload hardened image (signed)
      openstack.cloud.image:
        state: present
        name: ubuntu-golden-v1
        container_format: bare
        disk_format: qcow2
        filename: /tmp/ubuntu-golden-v1.qcow2
        properties:
          os_distro: ubuntu
          hw_disk_bus: scsi
          hw_scsi_model: virtio-scsi
          hw_vif_model: virtio
          os_require_quiesce: "yes"
          img_signature: "<base64-signature>"
          img_signature_certificate_uuid: "<barbican-cert-uuid>"
          img_signature_hash_method: "SHA-256"
          img_signature_key_type: "RSA-PSS"
```

Ansible: Deploy VM with Cloud-Init (+Config-Drive)

```
- hosts: controller
  tasks:
    - name: Create keypair
      openstack.cloud.keypair:
        state: present
        name: wk3-key
        public_key_file: ~/.ssh/id_rsa.pub

    - name: Create server
      openstack.cloud.server:
        state: present
        name: wk3-app-01
        image: ubuntu-golden-v1
        flavor: m1.latency
        key_name: wk3-key
        network: wk3-net
        security_groups: ["default"]
        config_drive: true # increase isolation from metadata spoofing
        userdata: |
          #cloud-config
          packages: [nginx]
```

Nova.conf: Emulator Threads & CPU Mode (Concept)

```
# /etc/nova/nova.conf
[libvirt]
virt_type = kvm
cpu_mode = host-model # host-passthrough: max perf, less migration compatibility

[compute]
# Reserve host CPUs for emulator/I/O threads to reduce jitter
reserved_host_cpus = 0,1
```

Mini Labs (Hands-on)

Lab A — Deterministic VM:

1. Flavor: dedicated pCPUs, `hw:numa_nodes=1`, hugepages (2MB trước, 1GB nếu sẵn có).
2. Tắt ballooning/KSM cho flavor này; pin *emulator threads*.
3. Benchmark p99 latency với `wrk` (Nginx) hoặc `stress-ng`.
4. Báo cáo bảng *before/after* (CPU steal/run queue, p95/p99).

Lab B — Migration Trade-off:

1. VM-A: virtio-net multiqueue; VM-B: SR-IOV.
2. Thử live migration; ghi nhận (A di chuyển được/B thường không).
3. So sánh throughput/latency bằng `iperf3`; phân tích đổi chác.

Performance Telemetry Map

Khái niệm	VMware/K8s phổ biến	KVM/Linux
CPU contention	Ready time	`steal`, run queue, `/proc/schedstat`
Pinning	CPU affinity (ESXi)	`virsh vcpupin`, `virsh emulatorpin`
NUMA	NUMA scheduler	`numactl -H`, `virsh dominfo`
Disk	vscsi queues	virtio-scsi multiqueue, iothreads
Net	vmxnet3 multiqueue	virtio-net multiqueue, vhost-net/user

Assessment (Đánh giá) & Deliverables (Bài nộp)

- **Báo cáo (2–3 trang):** thiết kế flavor/NUMA, pipeline image (ký số), benchmark ngắn (p95/p99).
- **Minh chứng:** ảnh chụp: cấu hình KVM, flavor extra_specs, VM cloud-init, snapshot/restore, `openstack resource provider show`.
- **Files:** playbooks, `nova.conf` trích đoạn, image properties, runbook migration.
- **Rubric (10 điểm):** cấu hình đúng (4) — bao gồm chứng cứ Placement; hiệu năng/tuning (3) — bảng p95/p99; tự động hoá (2) — playbook idempotent; tài liệu (1).

Wrap-up (Tổng kết)

- Ảo hoá VM cung cấp **cách ly**, **linh hoạt**, **hiệu quả** khi được cấu hình & giám sát đúng.
- Tuning quan trọng: NUMA, pinning, hugepages, virtio/SR-IOV, cache modes, **emulator threads**.
- Labs: deterministic VM & migration trade-off; bảo mật: image signing, metadata hardening, libvirt TLS.
- Tuần 4: **Fundamentals of Virtual Networks**.