

Lesson 09 — Computing & Virtualization

From the closet under the stairs to AWS us-east-1 — what runs your software, and the trick that made it all possible.

IT Landscape

Module 3

Where does compute live?

A spectrum from the dusty server in the stationery cupboard all the way to a hyperscaler data center the size of several city blocks.

- **On-premises** — hardware you own, sit in your building, managed entirely by your team.
 - **Co-location (colo)** — your hardware, someone else's building, power, and cooling.
 - **Managed hosting** — rented hardware in a provider's data center; they handle physical maintenance.
 - **Public cloud** — AWS, Azure, GCP. You rent capacity by the second; you never touch a physical server.
-

CapEx vs OpEx — the real trade-off

	On-Premises	Public Cloud
How you pay	Buy hardware upfront (CapEx)	Pay-as-you-go monthly bill (OpEx)
Control	Total — every setting, every cable	Limited — you configure, they run the DC
Redundancy	You build it; expensive	Included; multi-AZ is a checkbox
3 AM call	Your team's pager goes off	Cloud provider's team gets the call

What virtualization actually is

One big house chopped into separately-locked apartments. Each tenant has their own space; none of them knows the others exist.

- Before virtualization: one server ran one operating system. If you needed 10 services, you bought 10 servers — typically at **5–15% utilization** each.
 - After virtualization: one beefy server runs many virtual machines. Average utilization jumps to **60–80%**.
 - The magic layer in between is called the **hypervisor**.
-

The hypervisor: landlord of compute

Type 1 — bare metal

Runs *directly on the hardware*. No host OS in between.

VMware ESXi · Microsoft Hyper-V · KVM · Xen

Used in data centers. Maximum performance.

Type 2 — hosted

Runs *inside a host OS* like an app.

VirtualBox · VMware Workstation · Parallels

Used on laptops for testing. Convenient, some overhead.

Rule of thumb: if it runs in a production data center, it's Type 1. If you ran it on your laptop to test something, it's Type 2.

Containers — not VMs, but neighbours

	Virtual Machine	Container (Docker)
What's isolated	Full guest OS + kernel	Process + filesystem only
Shares	Hardware via hypervisor	Host OS kernel
Boot time	Seconds → minutes	Milliseconds
Size on disk	Gigabytes	Megabytes
Typical use	Long-lived server	Microservice, CI job

Both solve “this works on my machine” by bundling the environment alongside the code.

Why virtualization changed everything

- **Utilization** — one physical host replaces a rack of idle servers. Same bill, far more work done.
 - **Isolation** — a runaway process in one VM can't eat the memory of another. Failure is contained.
 - **Mobility** — a VM is just a file. Pause it, copy it to another host, resume. Hardware maintenance stops being an emergency.
 - **Snapshots** — before a risky change, take a snapshot. If it breaks, revert in seconds. This made sysadmins less afraid to experiment.
-

Orchestration — managing thousands at once

*When you have one VM, you log into it. When you have a thousand, you need a conductor. That conductor is an **orchestrator**.*

- **VMware vCenter** — the orchestrator for ESXi hypervisors. Dashboard, migrations, alerts.
 - **OpenStack** — open-source cloud orchestration; what many telcos and universities run to build private clouds.
 - **Kubernetes** — the dominant container orchestrator. Schedules, restarts, scales thousands of containers across a cluster.
 - **Nomad** — lighter-weight alternative to Kubernetes, handles both containers and VMs.
-

Kubernetes in plain English

You describe the *desired state*: "I want three copies of this web server running at all times."

Kubernetes makes it happen and keeps it happening.

- **Pod** — the smallest unit; one or more containers running together.
- **Node** — a worker machine (VM or physical) in the cluster.
- **Control plane** — the brain that schedules pods, watches for failures, and reschedules them elsewhere.
- If a node dies, Kubernetes notices within seconds and re-creates the failed pods on healthy nodes.

You don't say "run this on server 7." You say "run three of these." The scheduler picks the servers.

Watching the herd: metrics, logs, traces

- **Metrics** — numbers sampled over time: CPU at 72%, memory at 4.2 GB, requests per second = 340. *Tool: Prometheus*
- **Logs** — lines of text emitted by running processes: timestamps, errors, warnings. *Tools: Loki, Elasticsearch*
- **Traces** — the path one request took through a system: "this HTTP call touched service A, then B, then the database; step B was slow." *Tool: Jaeger*

Together these three signals are called **observability** — the ability to understand what a system is doing from the outside.

The dashboard layer

Raw metrics mean nothing without a picture. The dashboard layer glues data sources together into charts that humans can act on.

- **Grafana** — open-source, connects to Prometheus and dozens of other sources. The standard self-hosted option.
- **Datadog** — commercial SaaS, one agent install, everything in one place. Common at startups and mid-size companies.
- **New Relic, Dynatrace** — similar commercial offerings, stronger on APM (application performance monitoring).

When an alert fires at 3 AM, the on-call engineer opens a Grafana or Datadog dashboard first — before touching anything.

Who does this work?

Title	Core responsibility	Typical focus
Sysadmin	Keep servers alive	On-prem, legacy systems
SRE	Reliability as an engineering discipline	Large-scale cloud services
Platform Engineer	Build the internal tools devs use to deploy	Kubernetes, CI/CD pipelines
Cloud Architect	Design the infrastructure blueprint	Multi-cloud strategy, cost

Things people get wrong

- **“Cloud is always cheaper.”** Not true at scale. Large companies often save millions by repatriating workloads to owned hardware. The break-even is typically 3–5 years of consistent load.
 - **“Containers replace VMs.”** They coexist. Kubernetes itself usually runs *on top of* VMs. Containers package apps; VMs provide hardware isolation.
 - **“Kubernetes is for everyone.”** Kubernetes is powerful and genuinely complex. A five-person startup often does better with a simpler platform like Railway, Fly.io, or a managed container service.
-

Recap

- Compute lives on a spectrum: closet server → colo → cloud. CapEx vs OpEx is the key trade-off.
- Virtualization is one big house chopped into apartments. The hypervisor is the landlord.
- Type 1 hypervisors run on bare metal (data centers); Type 2 run inside an OS (laptops).
- Containers share the host OS kernel; VMs bring their own. Both isolate "this works on my machine."
- Orchestrators (vCenter, Kubernetes) manage thousands of VMs/containers like a conductor leads an orchestra.
- Observability = metrics + logs + traces. Grafana and Datadog make them visible.

Coming up: Lesson 10 — DevOps & Deployment (CI/CD, pipelines, and how code gets from a laptop to a live server).

