BUC3129 · TOPIC 6

Server Virtualization

One machine, many servers. Hypervisors, virtual machines, virtual networks and containers.

THE OUTCOME FOR THIS TOPIC

Create, configure and manage a virtual machine

Including virtual networking and storage allocation. That full sentence is the outcome you are assessed against.

OBSERVATION

Correct VM configuration sequence — the order you do things in, watched during the lab.

CONVERSATION

When is a container preferable to a full virtual machine? You will be asked this out loud.

PRODUCT

A working virtual machine with verified connectivity. Working, and proven to be working.

Verified means you can show me the address, the ping, and where the traffic went. Not ‘it seemed to work’.

PART A

Why virtualise at all?

Before the tools, the problem they solve.

- One service per physical server, and what it costs
- What virtualisation actually does to the hardware
- The honest trade-offs — it is not free

THE PROBLEM VIRTUALISATION WAS INVENTED TO SOLVE

One service, one server, one bad idea

BEFORE — physical servers

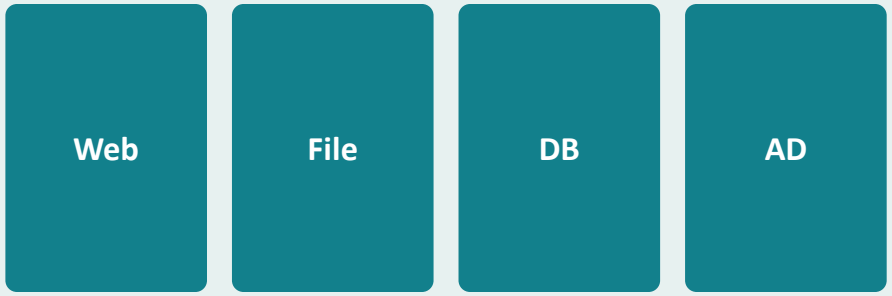
Web server
1 physical box
8% utilised

File server
1 physical box
8% utilised

Database
1 physical box
8% utilised

Domain controller
1 physical box
8% utilised

AFTER — one host, four guests



ONE physical server · 70% utilised

What it costs you

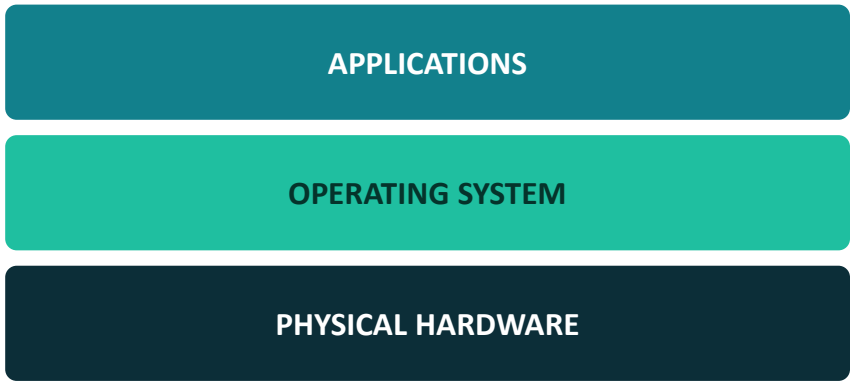
Four boxes to buy, power, cool, patch and replace — each idling at under ten per cent of capacity.

Consolidation is why virtualisation exists. Everything else — snapshots, portability, cloud — came afterwards.

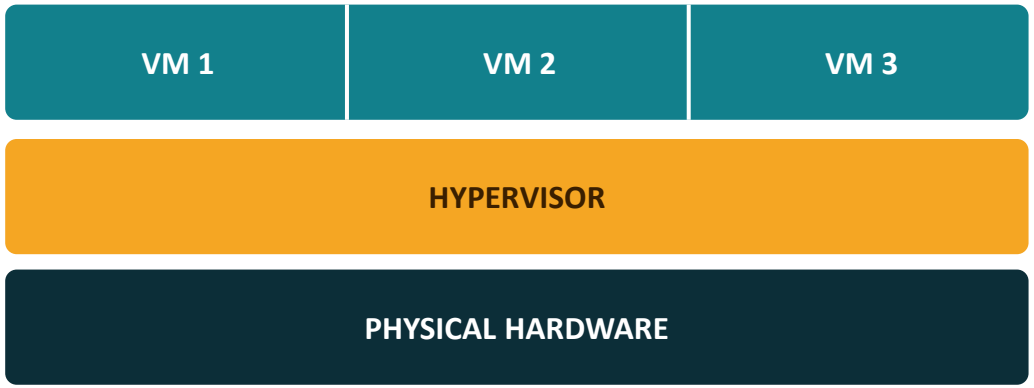
THE CORE IDEA

What virtualisation actually does

A layer of software divides one physical machine into several complete, isolated computers.



Without virtualisation
one OS owns the machine



With virtualisation
three isolated machines on one box

Each guest believes it owns real hardware. The hypervisor lies to all of them, convincingly and simultaneously.

BE HONEST ABOUT BOTH SIDES

What you gain, and what it costs

YOU GAIN

Consolidation — fewer physical boxes

Isolation — one guest cannot break another

Snapshots — undo a change in seconds

Portability — move a server to new hardware

Speed — a new server in minutes, not weeks

Test safely — break things on purpose

IT COSTS YOU

Overhead — the hypervisor takes resources

One host down = every guest down

Licensing — often per socket or per core

VM sprawl — forgotten, unpatched machines

Skills — a new layer to learn and diagnose

Noisy neighbours — one greedy VM starves others

‘One host down = every guest down’ is the sentence that pays for clustering, shared storage and backups.

PART B

Hypervisors

The software layer that makes it all possible — and the two families it comes in.

- What a hypervisor is and where it sits
- Type 1 versus Type 2, drawn side by side
- ESXi, Hyper-V, KVM, Proxmox, VirtualBox — which is which
- Choosing one for a given scenario

DEFINITION YOU WILL BE ASSESSED ON

What is a hypervisor?

Software that creates and runs virtual machines, and shares the physical hardware between them.

Also called a virtual machine monitor, or VMM. Both names appear in exam questions.

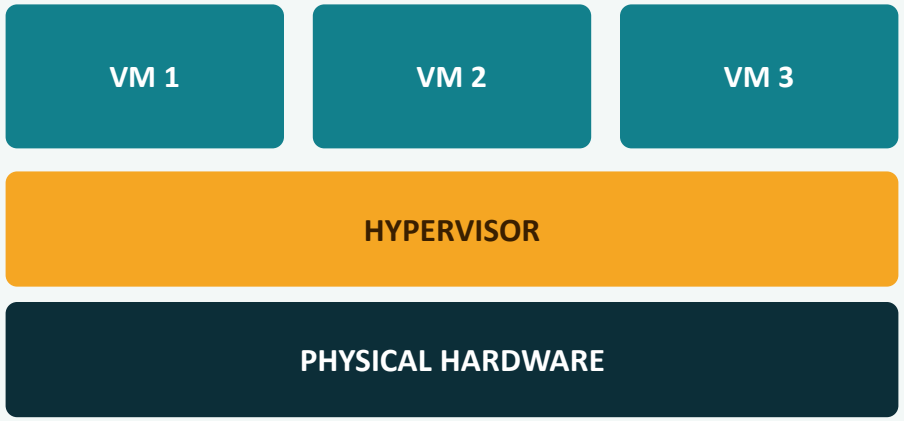
It allocates	vCPU time, memory, disk and network to each guest, and stops any one guest taking everything.
It isolates	Each guest is walled off. A kernel panic in one does not reach the others.
It emulates	The guest sees a standard virtual disk, virtual NIC and virtual chipset — whatever the real hardware is.

Because guests see standard virtual hardware, a VM will boot on any host — that is why VMs are portable.

THE DIAGRAM TO MEMORISE

Type 1 and Type 2, side by side

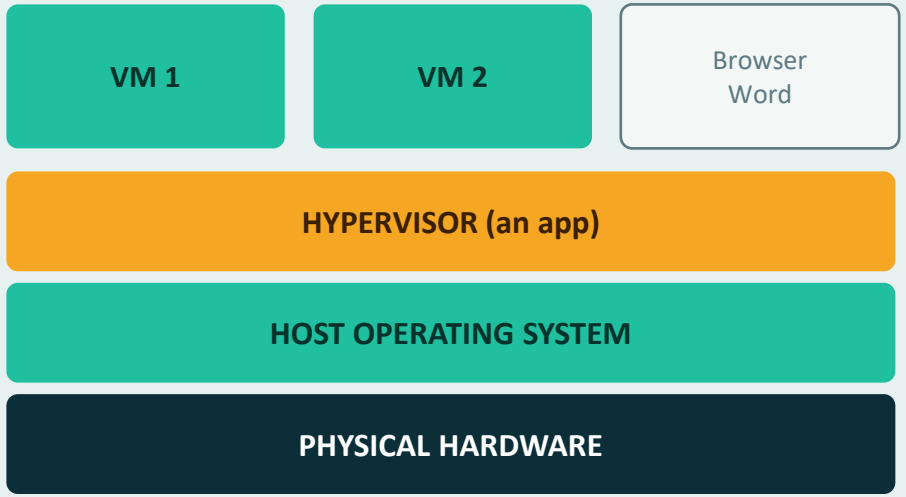
TYPE 1 — BARE METAL



No host operating system.
The hypervisor IS the operating system.

ESXi · Hyper-V Server · KVM · Proxmox · XCP-ng

TYPE 2 — HOSTED



Runs as a program on Windows or Linux,
alongside your other applications.

VirtualBox · VMware Workstation · Parallels

Count the layers between the VM and the hardware. That is the whole difference — and the whole performance story.

EXAMINABLE COMPARISON

Type 1 versus Type 2 in practice

	TYPE 1 (bare metal)	TYPE 2 (hosted)
Runs on	The bare hardware, directly	On top of a host OS, as an application
Performance	Near native — fewer layers	Lower — the host OS takes its cut
Boots into	The hypervisor itself	Windows or Linux, then you open it
Managed by	A web console or vCenter, over the network	A window on your desktop
Typical use	Production servers, data centres, hosting	Learning, testing, development, demos
Cost	Licensed (ESXi, Hyper-V) or free (KVM, Proxmox)	Free or low cost (VirtualBox)
Hardware access	Direct, with CPU virtualisation extensions	Mediated by the host OS

You will use Type 2 all semester. Your employer will run Type 1. Know both, and know why they differ.

NAMES YOU MUST RECOGNISE

The platforms, and where you meet them

VMware ESXi	Type 1. The enterprise standard for many years.	*Banks, telecoms, large data centres*
Microsoft Hyper-V	Type 1 role on Windows Server; also on Windows Pro.	*Windows-heavy organisations*
KVM	Type 1, built into the Linux kernel itself.	*Linux shops, and under most public clouds*
Proxmox VE	Free Type 1 platform built on KVM, web managed.	*Small organisations and labs*
VirtualBox	Type 2, free, cross-platform.	*This course, and your own laptop*

AWS, Azure and Google all run virtualisation underneath. When you rent a cloud server in Topic 8, you are renting a VM.

GROUP DISCUSSION

10 MIN

Which hypervisor, and why?

Four scenarios. In groups, choose Type 1 or Type 2, name a product, and be ready to defend it.

A A student who must run Ubuntu Server and Windows Server on a personal laptop for coursework.

B A SACCO consolidating four ageing physical servers onto one new machine in a locked room.

C A software team that needs twenty identical test machines created and destroyed every week.

D A hospital running a records system that must not stop, in a building with unstable power.

Scenario D has a second answer hiding in it. Find it.

PART C

Creating and managing a VM

The sequence you will be observed performing, and the commands behind the buttons.

- What a virtual machine is actually made of
- The eight-step creation sequence
- Sizing vCPU, memory and disk without guessing
- The same job in VirtualBox, KVM and Hyper-V — in commands

A VM IS JUST FILES

What a virtual machine is made of

This is the idea that makes backup, cloning and migration possible: a whole server is a folder on a disk.

Configuration file	vCPU, RAM, NIC, boot order, attached devices	*.vbox .vmx .xml*
Virtual disk	The guest's entire hard drive, in one file	*.vdi .vmdk .vhdx .qcow2*
Snapshot files	The differences since a point in time	*delta / diff files*
NVRAM / state	Firmware settings and saved memory state	*.nvram .sav*

Copy that folder to another machine, import it, and the server runs again — same hostname, same data, same everything. That is what people mean by portability, and it is why a VM is easier to protect than a physical server.

OBSERVATION CRITERION · CORRECT CONFIGURATION SEQUENCE

The eight steps, in this order

1	**Name and OS type**	Tells the hypervisor which defaults and drivers to assume
2	**Allocate memory**	Leave the host at least 4 GB. Never allocate into the red zone
3	**Create the virtual disk**	Choose format and thin or thick provisioning now
4	**Set vCPU count**	Start with 1 or 2. More is not automatically faster
5	**Attach the installation media**	Mount the ISO to the virtual optical drive
6	**Choose the network mode**	NAT, bridged, host-only or internal — decide before you boot
7	**Boot and install the guest OS**	Then install guest additions or integration tools
8	**Verify and snapshot**	Confirm the address, ping out, then snapshot the clean build

Step 8 is the one learners skip — and it is the one that carries the marks. No verification, no product evidence.

STOP GUESSING AT THE NUMBERS

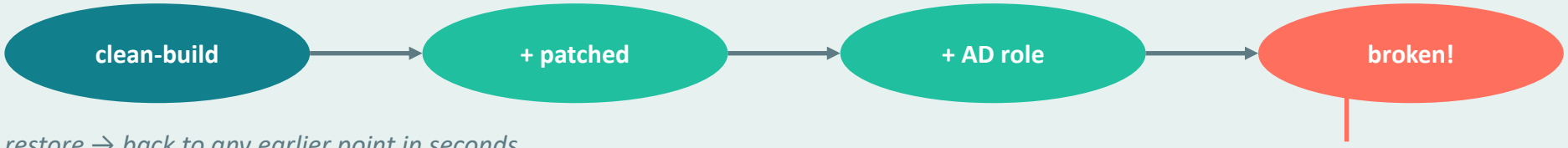
How much to allocate

Memory	Ubuntu Server needs 1–2 GB. Windows Server needs 4 GB minimum, 8 to be usable. Leave the host at least 4 GB.
vCPU	Start with 1 or 2. Never allocate more vCPU to one guest than the host has physical cores.
Disk	Ubuntu Server 25 GB, Windows Server 60 GB. Thin provisioning lets you be generous with the maximum.
Overcommit	You may allocate more total memory than the host has — until every guest wants it at once, and then everything swaps.

Worked example: a host with 16 GB. Windows Server guest 6 GB, Ubuntu guest 2 GB, host keeps 8 GB.
That runs comfortably. Give the Windows guest 12 GB instead and both the host and the guest crawl.

THE FEATURE THAT CHANGES HOW YOU LEARN

Snapshots and clones



restore → back to any earlier point in seconds

Snapshot	A saved point in time for one VM. Restore returns the machine exactly there. Take one before every risky change.
Full clone	An independent copy of the whole VM. Uses full disk space. Survives deletion of the original.
Linked clone	A copy that shares the parent's disk and stores only differences. Fast and small — but dies if the parent is deleted.

Never leave snapshots on a production server for weeks. They grow, they slow the disk, and they are not a backup.

PART D

Virtual networking and storage

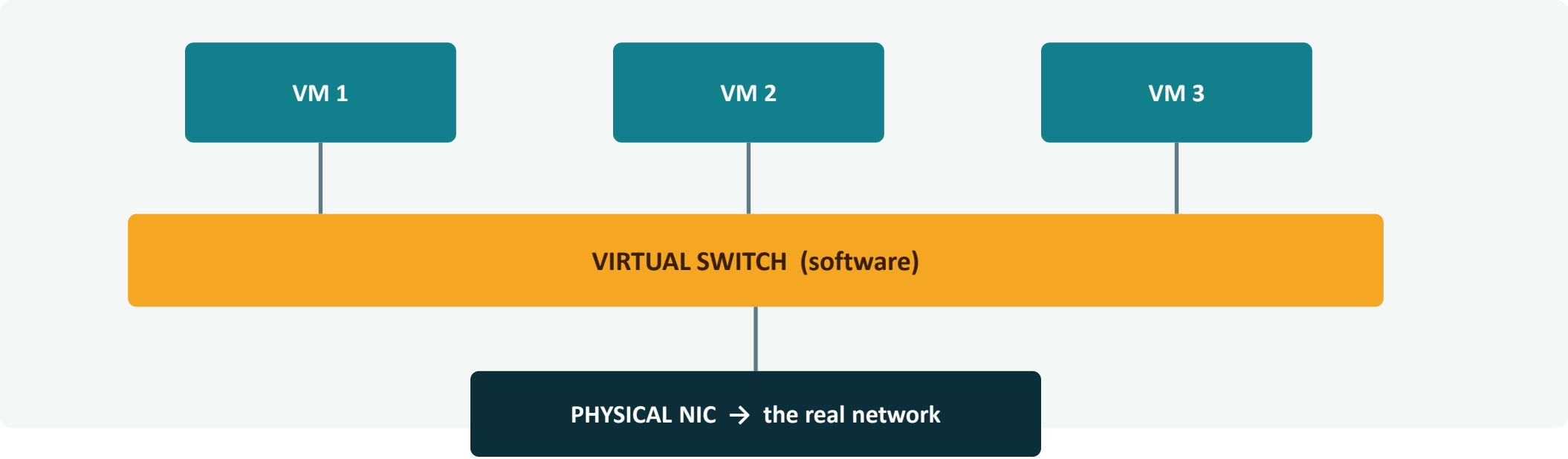
The half of the outcome that learners forget — and the half that carries the connectivity evidence.

- The virtual switch, and what it really is
- NAT, bridged, host-only and internal — drawn and compared
- Virtual disks: format, and thin versus thick provisioning
- Proving connectivity, with commands

THE CONCEPT UNDERNEATH EVERY NETWORK MODE

The virtual switch

The hypervisor implements a switch in software. Your guests plug into it exactly as machines plug into a real one.



Whether that switch reaches the real network, and how, is exactly what the four network modes decide.

DECIDE THIS BEFORE YOU BOOT THE GUEST

The four network modes

NAT

Guest gets a private address from the hypervisor. It can reach the internet. Nothing outside can reach it.

Default. Fine for updates and browsing.

BRIDGED

Guest appears on the real LAN with its own address from your router. Others can reach it.

Use for a server that must be reachable.

HOST-ONLY

A private network between the host and its guests. No internet at all.

Use for a safe, isolated lab.

INTERNAL

Guests talk only to each other. Not even the host can reach them.

Use to simulate an isolated network.

For your assessed VM, use BRIDGED. Verified connectivity is far easier to demonstrate when the machine is really on the network.

PRODUCT CRITERION · VERIFIED CONNECTIVITY

Setting the mode, then proving it works

```
# VirtualBox: change mode on a stopped VM
VBoxManage modifyvm "srv-ubuntu" \
  --nic1 bridged --bridgeadapter1 wlan0
# NAT instead
VBoxManage modifyvm "srv-ubuntu" --nic1 nat
# host-only network
VBoxManage modifyvm "srv-ubuntu" \
  --nic1 hostonly --hostonlyadapter1 vboxnet0
# list host-only networks
VBoxManage list hostonlyifs
```

```
# inside the guest - the four proofs
ip a                    # do I have an address?
ip route                # do I have a gateway?
ping -c 3 192.168.1.1   # can I reach the gateway?
ping -c 3 8.8.8.8       # can I reach the internet?
ping -c 3 google.com    # does DNS resolve?
# from the host, prove the guest is reachable
ping -c 3 192.168.1.57
ssh admin@192.168.1.57
```

The diagnostic order matters, and it is the same order as Topic 5's method: address → gateway → internet → name. Address but no gateway is a configuration fault. Gateway but no internet is a NAT or bridge fault. Internet but no names is DNS.

STORAGE ALLOCATION · PART OF THE OUTCOME

Thin or thick provisioning?

THIN (dynamically allocated)

Declared 60 GB, uses 12 GB on disk today and grows as the guest writes.

Saves space. Slightly slower. Risk: several thin disks can promise more space than the host actually has.

THICK (fixed size)

Declared 60 GB, takes 60 GB immediately.

Slightly faster and predictable. No nasty surprises — the space is already yours.

```
# VirtualBox: Standard = thin, Fixed = thick
VBoxManage createmedium disk --filename srv.vdi --size 60000 --variant Fixed
# resize a disk later (guest must then extend the partition)
VBoxManage modifymedium disk srv.vdi --resize 80000
# KVM / qemu: inspect and convert
qemu-img info srv-web.qcow2
```

Growing the virtual disk does not grow the guest's partition. That is a second job, inside the guest.

HANDS ON · ASSESSED

40 MIN

Build a VM with verified connectivity

This is the Product evidence for Topic 6. Work alone. I will be watching the sequence, not the speed.

1. Create a VM named srv-<yourname>, Ubuntu Server 64-bit, 2 GB RAM, 2 vCPU
2. Create a 25 GB thin-provisioned virtual disk and attach the installer ISO
3. Set the network adapter to BRIDGED before first boot
4. Install the guest OS, then install guest additions
5. Capture: ip a, ip route, ping to the gateway, ping to 8.8.8.8, ping to a name
6. From the HOST, ping the guest's address and paste that output too
7. Take a snapshot named clean-build, then show me the snapshot list

Submit: the seven screenshots, in order, in one document, with your VM name visible in each.

PART E

Containers — orientation

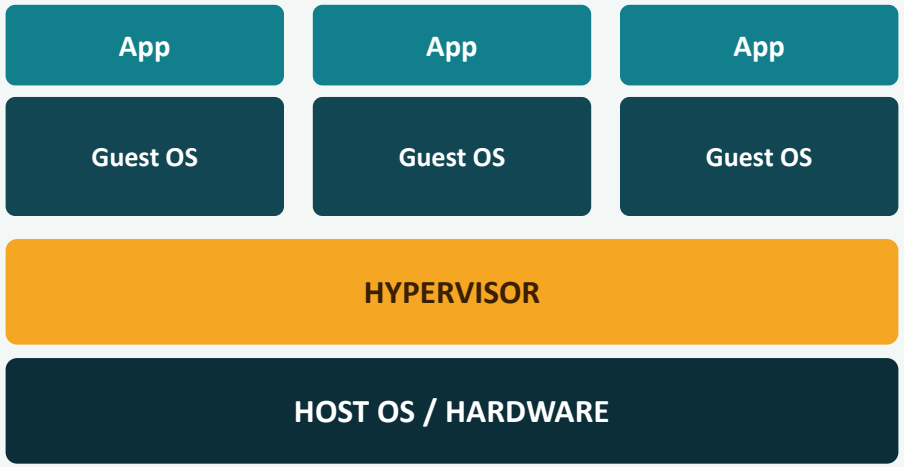
A lighter alternative. You are not expected to deploy one; you are expected to know when you would.

- How a container differs from a virtual machine
- What you gain and what you give up
- The conversation question you will be asked

THE SECOND DIAGRAM TO MEMORISE

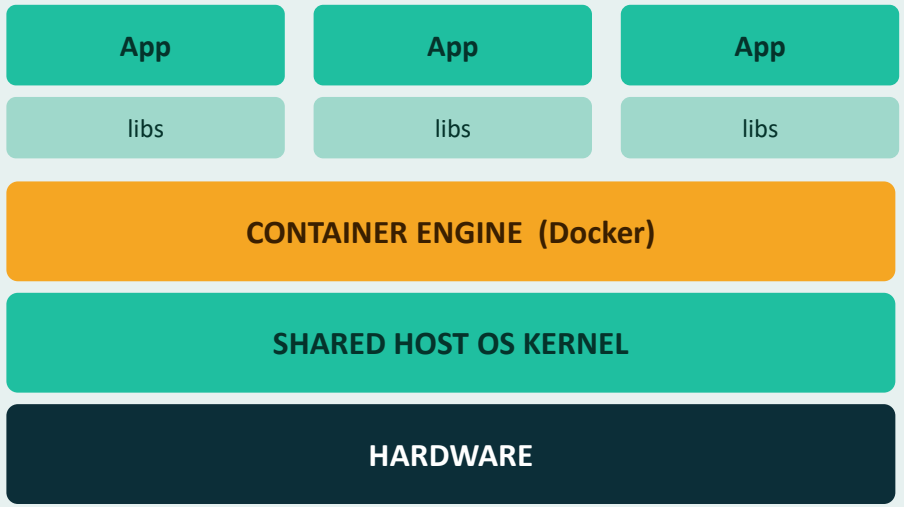
Virtual machines and containers

VIRTUAL MACHINES



Three full operating systems.
Gigabytes each. Boots in minutes.

CONTAINERS



One operating system, shared.
Megabytes each. Starts in seconds.

Look at the middle row. That is the entire difference: a VM carries its own operating system, a container borrows the host's.

CONVERSATION CRITERION · YOU WILL BE ASKED THIS

When is a container preferable?

Choose a container	One application, packaged with its libraries	*Web app, API, microservice*
Choose a container	You need it to start in seconds and scale out	*Traffic spikes, CI pipelines*
Choose a VM	You need a different operating system from the host	*Windows guest on a Linux host*
Choose a VM	You need full isolation, or a whole machine to administer	*Domain controller, database server*
	The workload needs kernel-level control or special	

```
# the whole idea, in three commands
docker run -d --name web -p 8080:80 nginx     # a web server, running, in about two seconds
docker ps                                     # what is running
```

STRUCTURED DISCUSSION

12 MIN

VM, container, or neither?

In groups of four. For each workload: VM or container, and one sentence saying why.

1. A company website that gets busy at month end and quiet the rest of the time
2. The organisation's Active Directory domain controller
3. A student's practice environment for learning Windows Server
4. Twenty identical copies of one API, behind a load balancer
5. A legacy accounting application that only runs on Windows Server 2012

Then the harder question: which of the five would you refuse to virtualise at all, and why?

YOUR EVIDENCE FOR TOPIC 6

What you hand in

#	What you do	What you hand in	Column
1	Build the VM. Ubuntu Server, bridged, snapshotted.	Seven screenshots in sequence, VM name visible in each.	Product
2	Prove connectivity. From inside the guest and from the host.	Output of ip a, ip route, three pings, and a ping from the host.	Product
3	Compare the hypervisors. Type 1 against Type 2.	A six-row table in your own words plus a paragraph choosing one for a stated scenario.	Product
4	Answer the container question. Prepare it, do not memorise it.	Be ready to explain out loud when a container beats a VM, and why.	Conversation

This VM is not disposable. Keep it — Topics 7, 8 and the capstone all build on the machine you make today.

BEFORE YOU GO

Name one thing a container cannot do that a VM can.

One line each. No repeats.