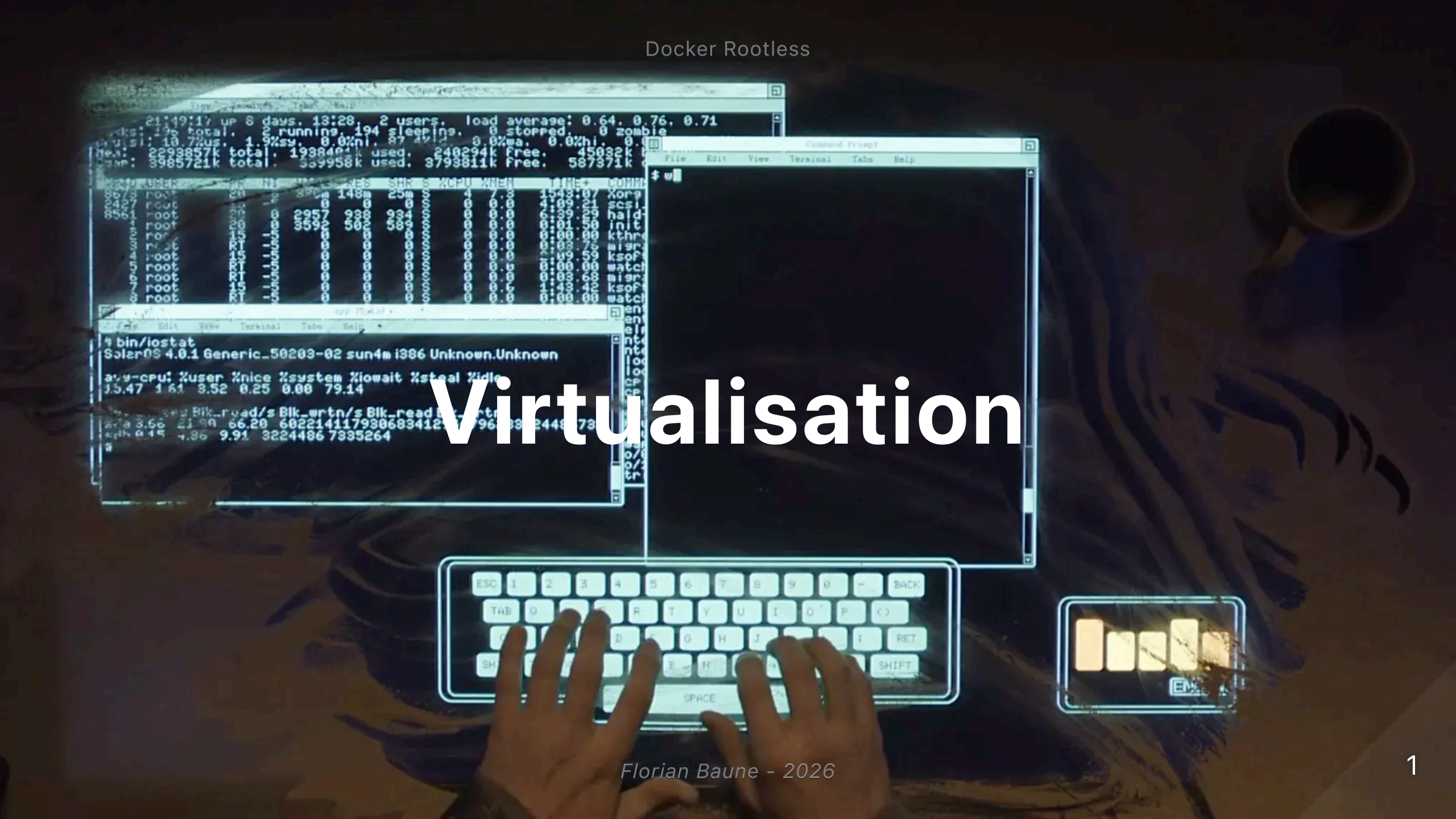

Virtualisation

Virtualiser : de quoi parle-t-on ?

Vous en faites déjà, sans la nommer

- La **mémoire virtuelle** de votre OS
- Les **VLAN** du réseau de votre entreprise
- Les volumes **LVM** ou **RAID** d'un serveur

Définition et vocabulaire

> Présenter une ressource **logique** à la place d'une ressource **physique**, via une **couche d'indirection** qui découple les deux.

- **Virtualisation** : englobant toutes formes confondues
- **Machine virtuelle (VM)** : *uniquement* une machine virtuelle système — OS invité complet

En entreprise...

- ⚠️ **Terme souvent galvaudé** : « virtualisation » désigne souvent **les seules VM**, les conteneurs étant présentés par contraste

Axe 1 : Virtualiser quoi ? quel niveau ?

Niveau	Exemples
Poste de travail	VDI
Réseau	VLAN, VXLAN, cartes réseau virtuelles, SDN
Stockage	LVM, RAID, SAN
Mémoire	mémoire virtuelle

Virtualisation vs. abstraction ?

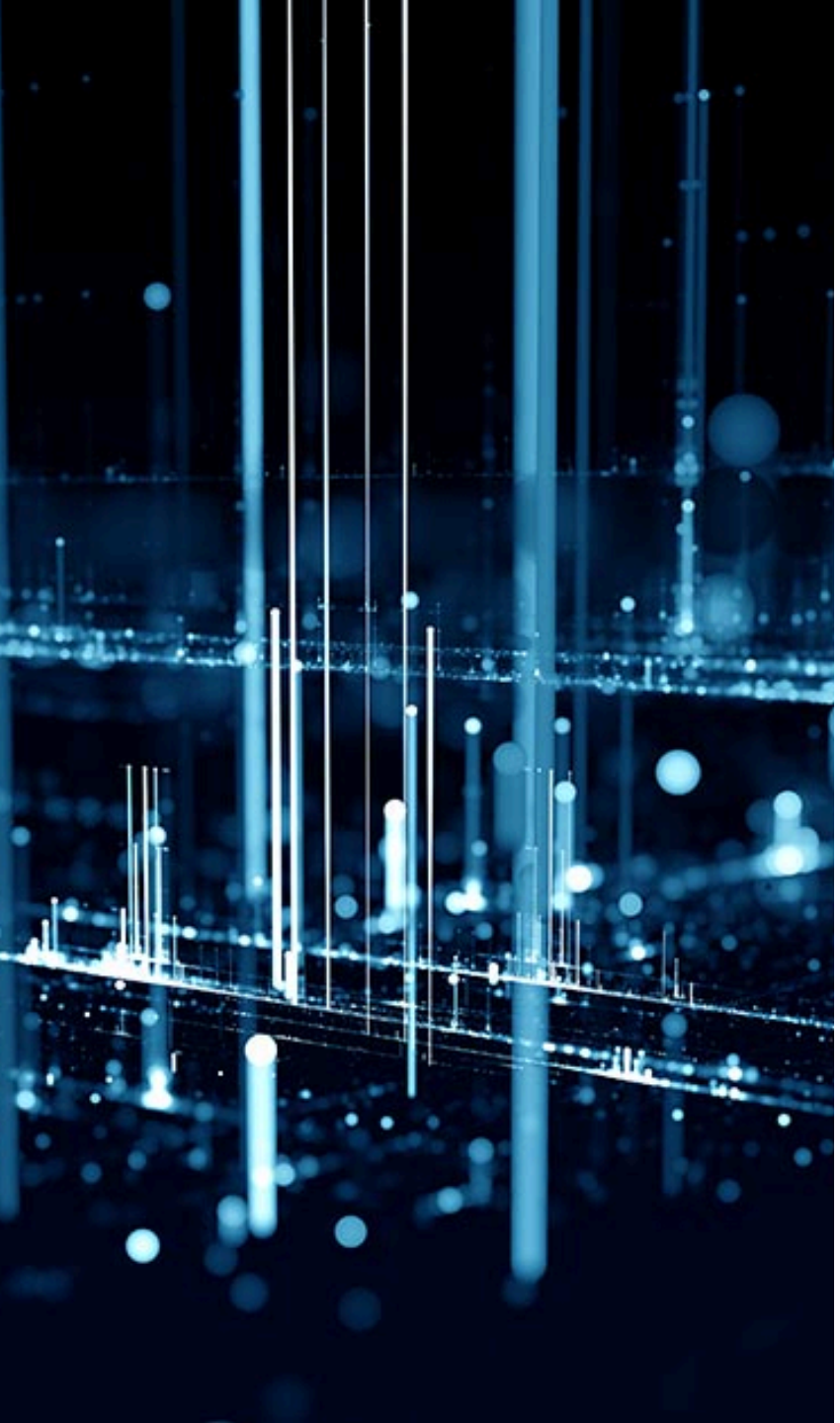
- **Critère** : la ressource logique offre **la même interface** que la ressource physique
- Celui qui l'utilise **croit avoir affaire à la vraie**
  - un VLAN se comporte comme un LAN
  - un volume LVM comme un disque
  - la mémoire virtuelle comme de la mémoire physique
- **Contre-exemple** : un fichier n'est **pas** un disque virtuel — interface différente, abstraction de plus haut niveau

Axe 2 : Cardinalité & Nature

- **Combien ?** — un vers plusieurs (*partitionnement*) ou plusieurs vers un (*agrégation*)
- **De quelle nature ?** — même type que la ressource physique, ou type différent (*traduction*)

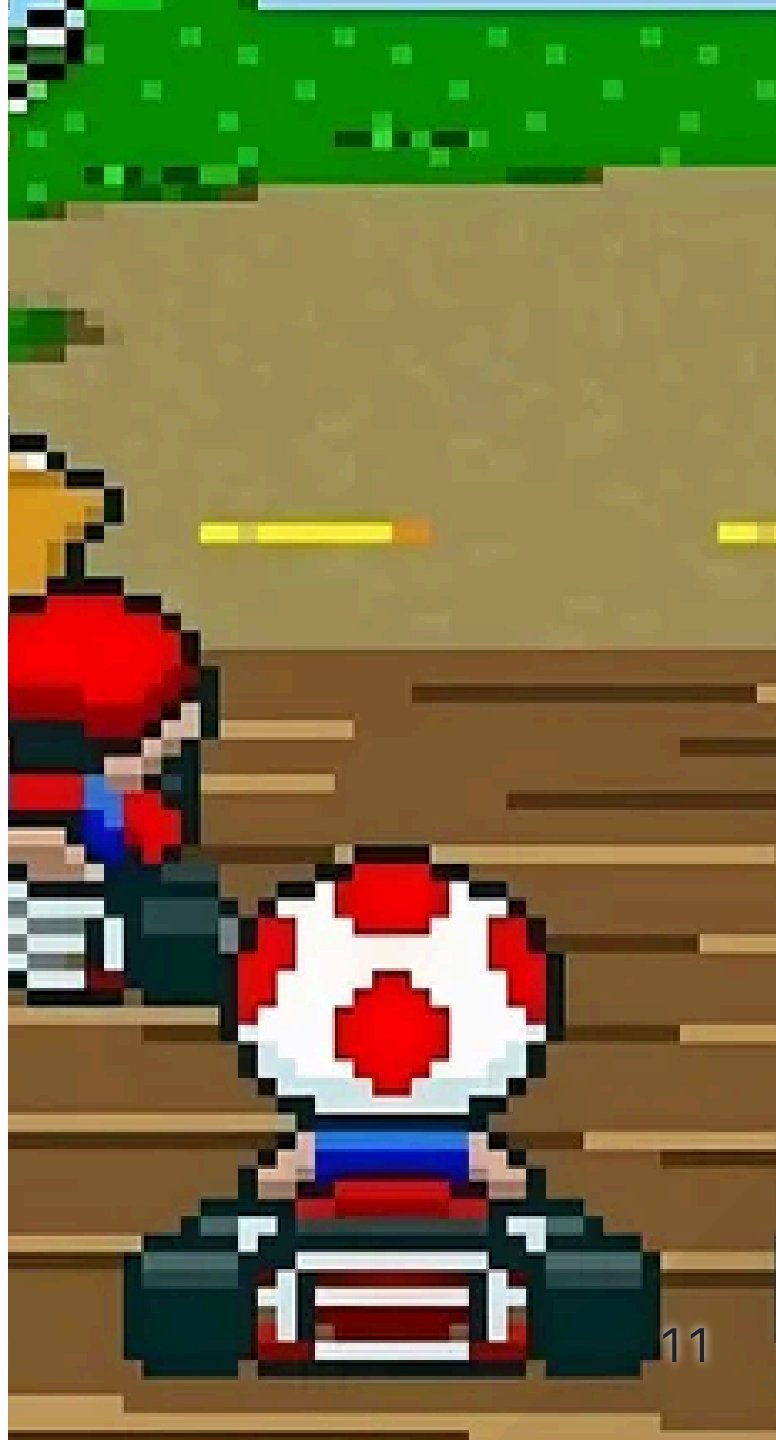
Axe 2 : Cardinalité & Nature

	Un vers plusieurs	**Plusieurs vers un**
Même type	VLAN, vGPU	RAID
Type différent	GameBoy émulées	—



Formes de virtualisation

L'Émulation



Ce qu'on virtualise ici

- **Niveau** : le **jeu d'instructions** du processeur, et les périphériques autour
- **Relation** : *un type vers un autre* — on ne découpe rien, on **traduit**
- **Physique** : un CPU x86 — **logique** : un CPU ARM, ou le 6502 d'une NES

Principe de l'émulation

- Reproduire par logiciel le **comportement complet d'une autre machine**
- Simule le processeur, la mémoire, les périphériques…
- Chaque instruction est **interprétée ou traduite** pour l'architecture hôte

Illustration : rétro-gaming

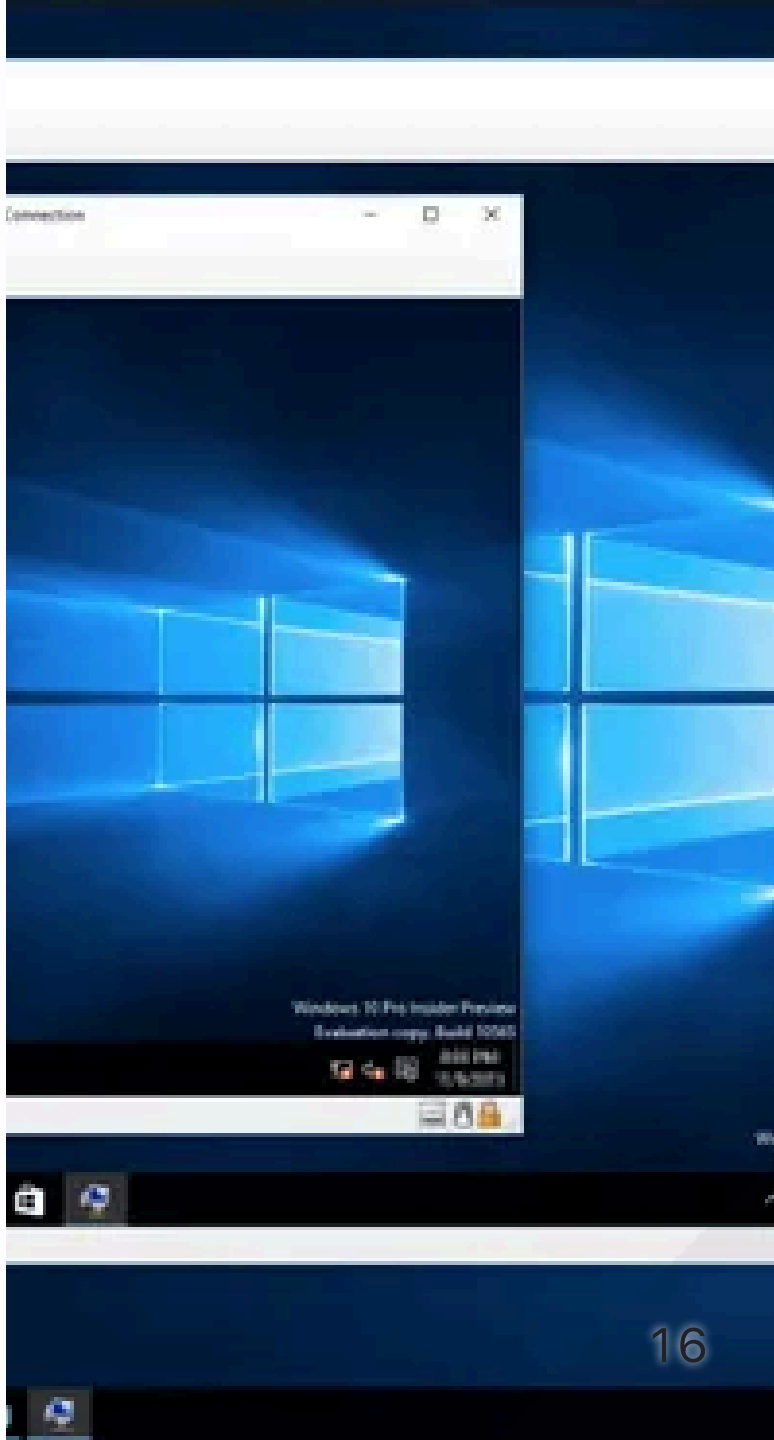
- Le jeu croit fonctionner sur une vraie console
- L'émulateur imite le **CPU, GPU, son, contrôleur**, etc.
- **Le critère de l'interface est rempli** : la ROM n'a aucun moyen de savoir qu'elle n'est pas sur la vraie machine

Performances et flexibilité

- **Très grande compatibilité** entre architectures
- Peut émuler des systèmes totalement différents
- **Coût élevé en performance** : traduire se paie à chaque instruction

> C'est le prix de l'axe 2 : la traduction est la seule des trois relations qui travaille en permanence.

Machines Virtuelles



Florian Baune - 2026

Ce qu'on virtualise ici

- **Niveau** : la **machine matérielle** complète — CPU, RAM, BIOS, disques, cartes réseau
- **Relation** : *un vers plusieurs* — une machine physique, N machines logiques
- **Rappel de vocabulaire** : dans ce cours, « VM » désigne **ça et rien d'autre**

Principe général

- Plusieurs **systèmes d'exploitation** sur une même machine physique
- L'hyperviseur **est** la couche d'indirection : il découple les machines logiques du matériel réel
- Chaque VM croit disposer de son propre matériel : l'OS invité boote **sans rien savoir**

Techniques de virtualisation

- **Trap-and-emulate** : interception des instructions sensibles
- **Binary translation** : remplacement d'instructions interdites — de la *traduction* au service d'un partitionnement
- **Paravirtualisation** : l'OS invité coopère avec l'hyperviseur via des *hypercalls*

> Cas limite du critère de l'interface : en paravirtualisation, l'invité **sait** qu'il est virtualisé.

L'accélération matérielle

- **VT-x / AMD-V** : un mode du processeur dédié à l'hyperviseur
- **EPT / NPT** : la double traduction mémoire passe dans le matériel
- L'invité s'exécute **directement sur le CPU** : overhead quasi nul

> Avant, l'hyperviseur rusait en logiciel ; depuis, le matériel sait virtualiser. C'est là que la VM décroche de l'émulation, qui traduit toujours.

Avantages des VM

- **Isolation forte** : chaque invité a son propre noyau
- **N'importe quel OS invité** : Windows sur Linux, Linux sur Windows
- **Rien à adapter** : l'OS invité ne change pas d'une ligne
- **Une VM est un fichier** : snapshot, sauvegarde, migration

Limites des VM

- **Lourd** : un OS complet par instance, en RAM et sur disque
- **Démarrage lent** : il faut booter
- **Ne franchit pas la barrière des architectures** : même jeu d'instructions que l'hôte

> Pour en changer, il faudrait traduire — et on retombe sur l'émulation.

Hyperviseurs

Où se pose la couche d'indirection ?

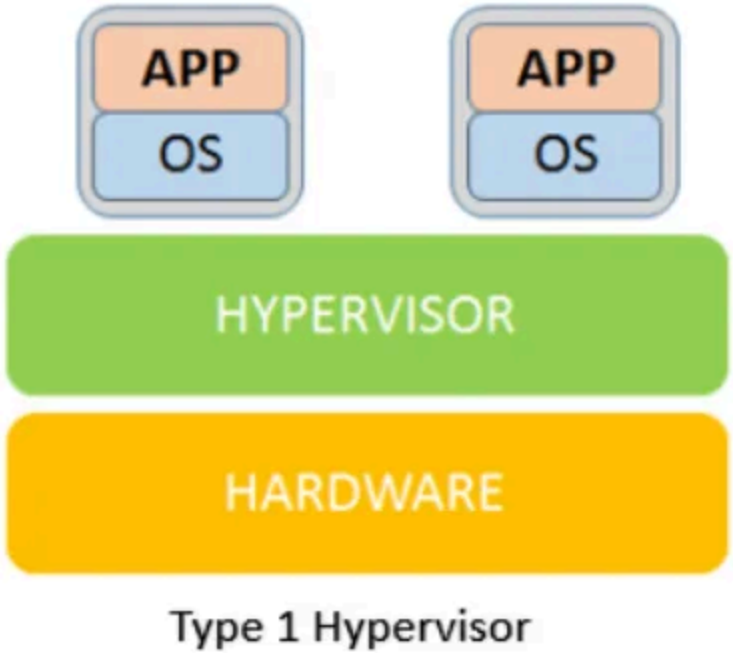
- **Type 2** : au-dessus d'un OS hôte — l'indirection s'empile sur une autre
- **Type 1** : directement sur le matériel — l'indirection est la première chose qui démarre

Hyperviseurs de type 2 (hébergés)

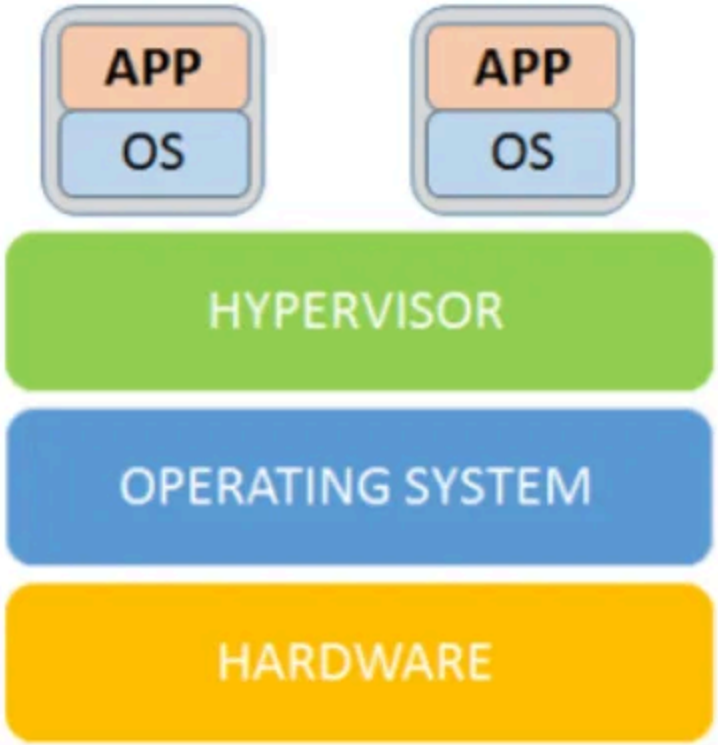
- Installés **comme une application classique**
- S’appuient sur l’OS hôte
- Idéal pour tests, TP, environnements personnels

Hyperviseurs de type 1 (bare metal)

- Installés **directement sur le matériel**
- Aucun OS hôte : l'hyperviseur est le premier lancé
- Performances supérieures, usage professionnel



Type 1 Hypervisor



Type 2 Hypervisor

Conteneurisation

Ce qu'on virtualise ici

- **Niveau** : non plus le matériel, mais la **vue qu'un processus a de son OS**
- **Relation** : *un vers plusieurs* — un noyau unique, N vues logiques
- **La couche d'indirection est le noyau lui-même**, pas un hyperviseur posé à côté

Principe général

- Forme de virtualisation **au niveau du système d'exploitation**
- Plusieurs **conteneurs** partagent le **même noyau**
- Chaque conteneur a son propre **espace isolé** : processus, fichiers, réseau...
- Le processus fait des appels système ordinaires : il voit un `/` et un PID 1 **à lui**

Mécanismes techniques

- **Namespaces** : isolation des processus, fichiers, réseau, etc.
- **Cgroups** : contrôle des ressources (CPU, RAM, I/O)
- **Images minimales** : système de fichiers embarqué avec l'application
- **Pas de noyau dans le conteneur** : il utilise celui de l'hôte

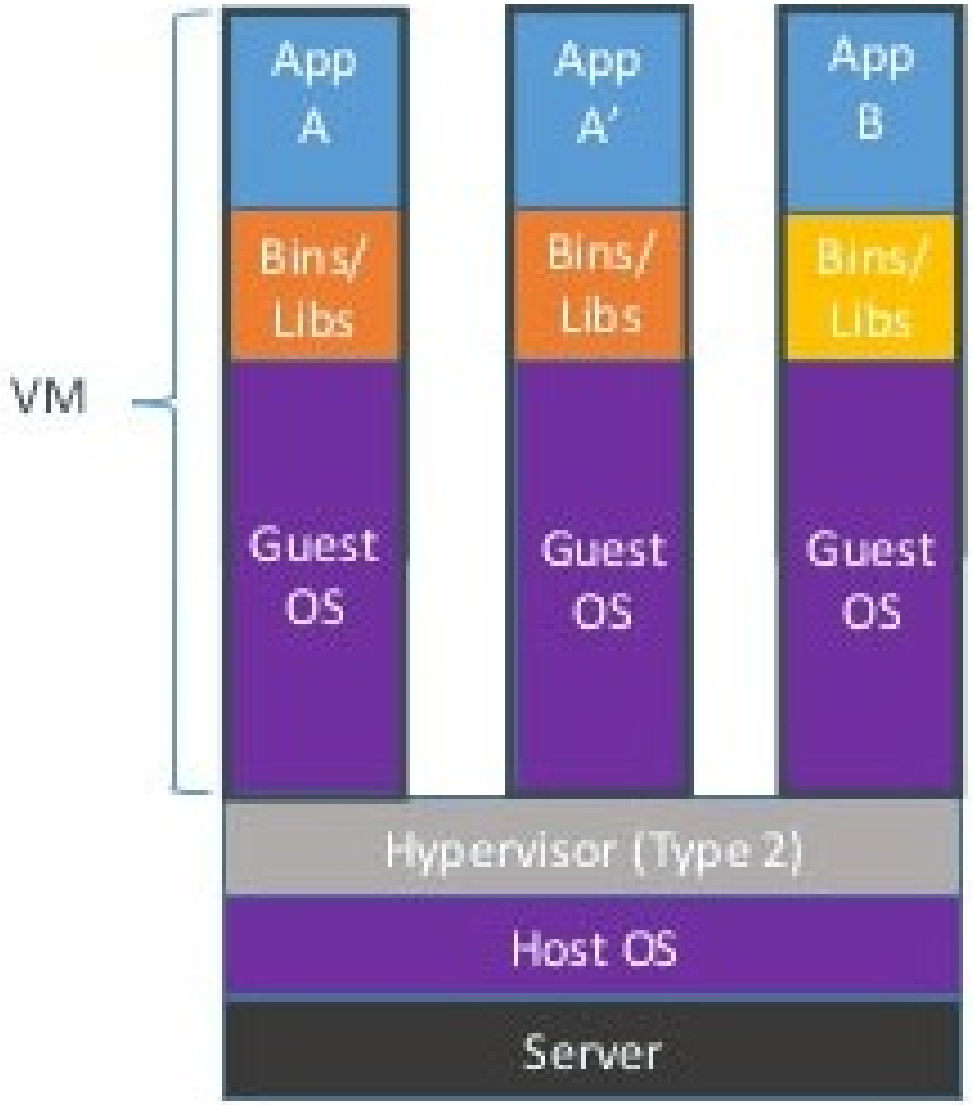
> Les *namespaces* fabriquent la vue, donc virtualisent ; les *cgroups* rationnent sans rien virtualiser.

Avantages des conteneurs

- **Légers** : pas d'OS complet dans chaque instance
- **Démarrage rapide** : quelques millisecondes
- **Scalables** : faciles à déployer en masse
- **Portables** : "ça marche pareil partout"

Limitations

- **Même noyau pour tous** : on n'a pas virtualisé le matériel, donc pas de conteneur Windows sur Linux
- **Isolation moins forte** qu'une VM : la ressource partitionnée est aussi celle qui partitionne
- **Noyau vulnérable = tous les conteneurs exposés**



Containers are isolated, but share OS and, where appropriate, bins/libraries

Pourquoi les oppose-t-on sans cesse ?

- Parce que « virtualisation » au sens courant désigne **les seules VM**
- Ce schéma compare deux **niveaux** différents de l'axe 1, pas deux concurrents
- La question n'est pas « VM ou conteneur ? » mais **« de quelle virtualisation ai-je besoin ? »**

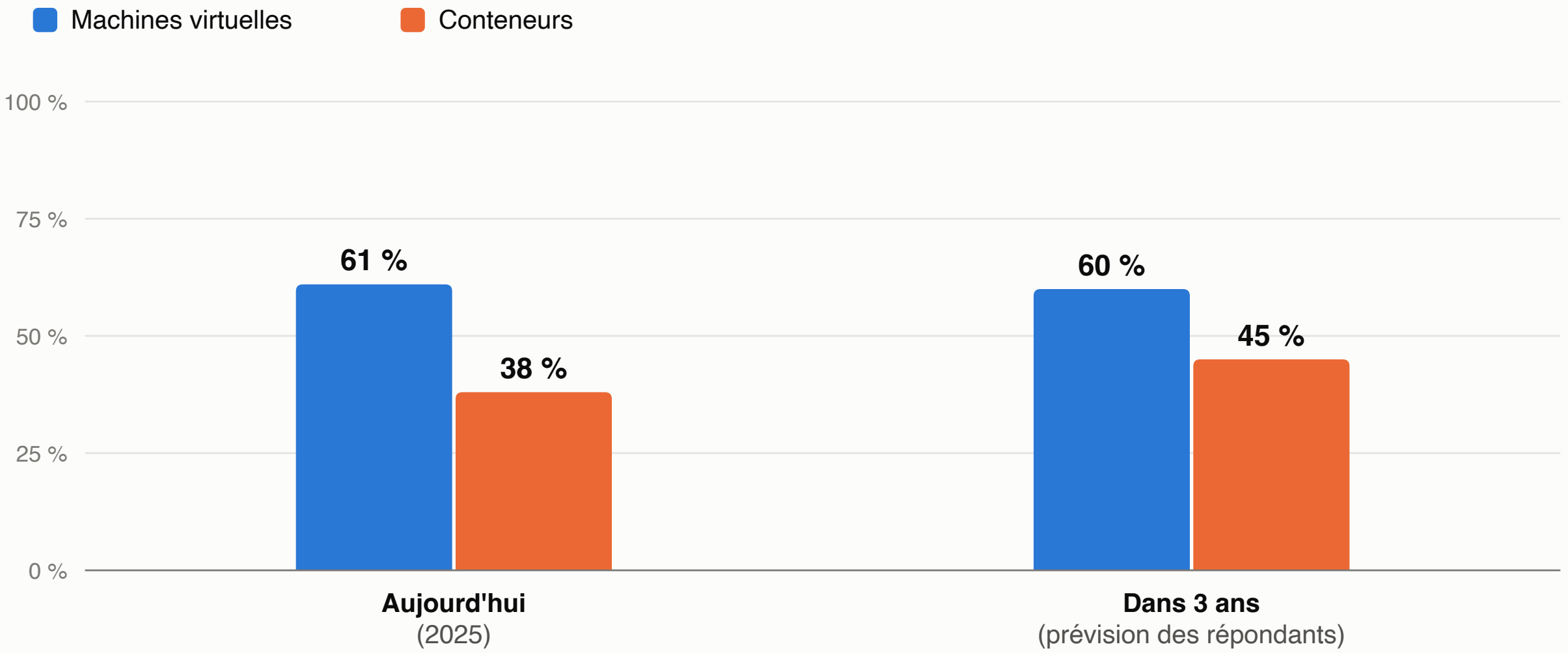
Utilisation

Et dans la vraie vie ?

- On a vu trois familles et leurs mécanismes
- Laquelle utilise-t-on vraiment, et dans quelles proportions ?

Grandes entreprises : VM et conteneurs, aujourd'hui et dans 3 ans

Répartition déclarée des workloads entre VM traditionnelles et conteneurs

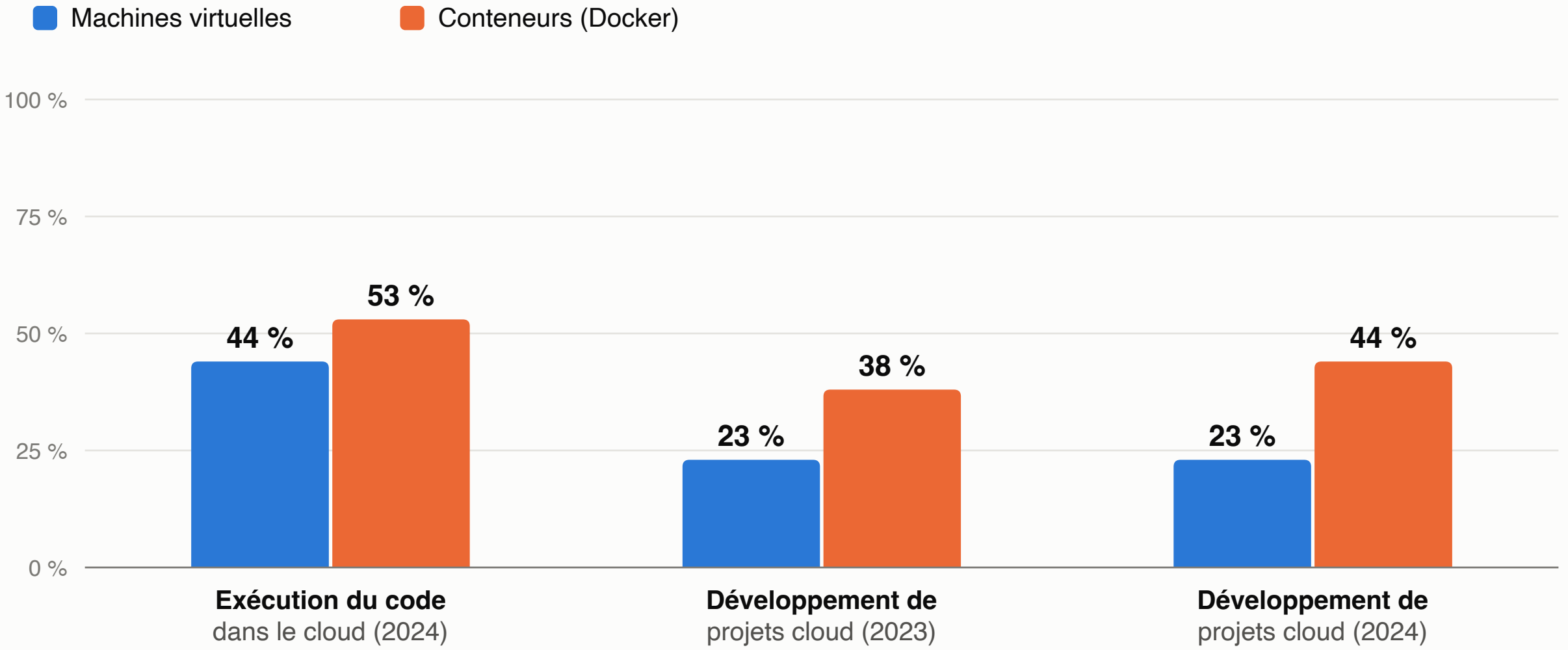


Étude commanditée par Red Hat, éditeur d'une plateforme VM + conteneurs (OpenShift).

Source : Red Hat / Illuminas, « The state of virtualization », 2025 — 1 010 décideurs IT et praticiens (US, UK, APAC), mars-avril 2025
https://www.redhat.com/en/engage/state-of-virtualization-ebook

Développeurs Python : VM et conteneurs, en production cloud et en dev

Part des répondants concernés utilisant chaque technologie (plusieurs réponses possibles)



Questions « How do you run code in the cloud? » et « How do you work on cloud projects? ». Population : développeurs Python.

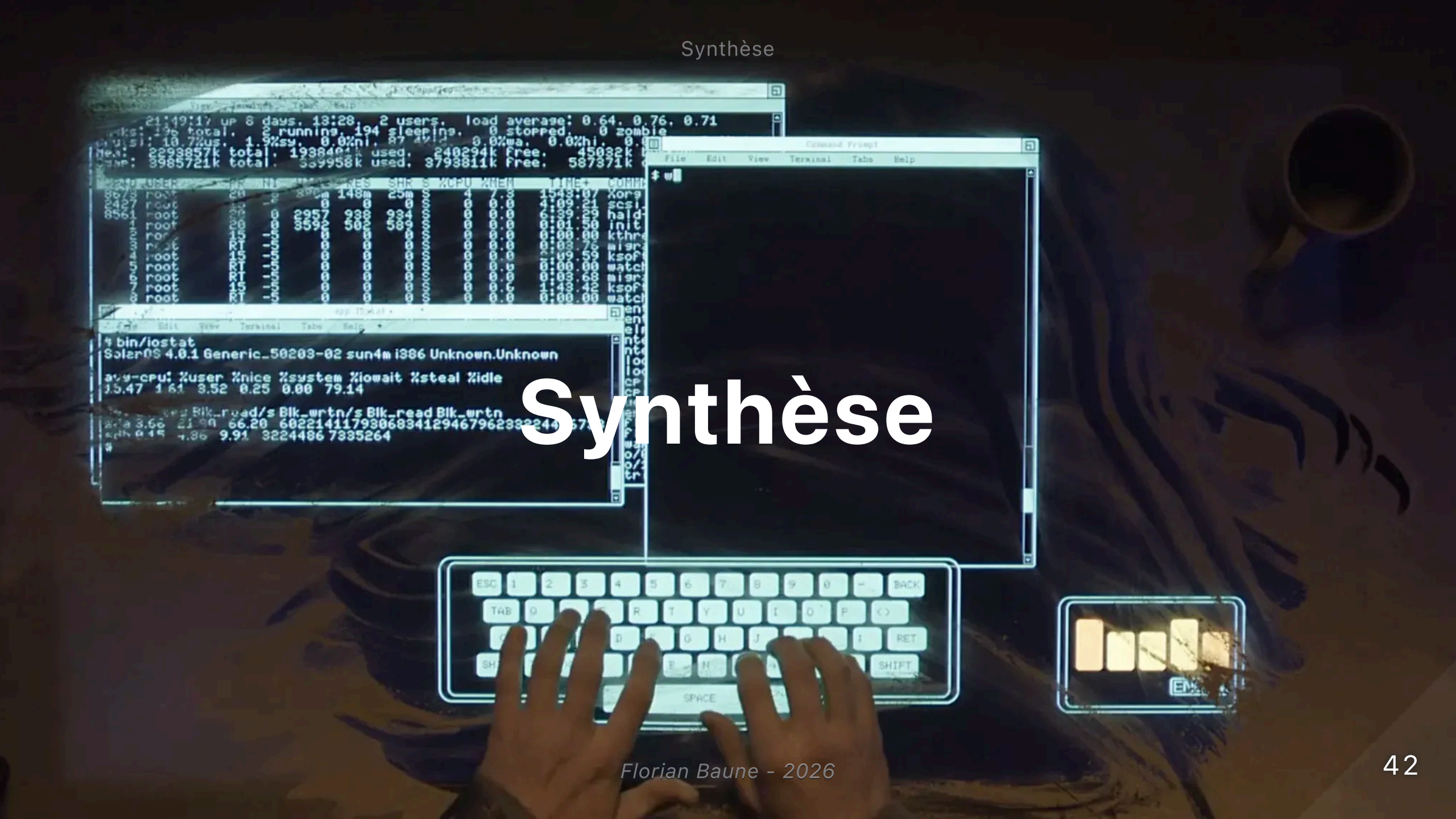
Source : Python Software Foundation & JetBrains, Python Developers Survey 2024 — 25 000+ réponses retenues, oct.-nov. 2024
https://lp.jetbrains.com/python-developers-survey-2024/

Ce que disent les chiffres

- Les deux familles **coexistent** : aucune n'a chassé l'autre
- Le conteneur s'est installé **très vite**, sans faire disparaître la VM
- L'émulation, elle, **n'apparaît pas** : ce n'est pas une technique d'infrastructure

Alors, lequel choisir ?

- **Autre architecture ?** il faut émuler, pas le choix
- **Autre OS, ou besoin d'isolation forte ?** machine virtuelle
- **Même OS, beaucoup d'instances, démarrage rapide ?** conteneur


Synthèse

Ce qu'il faut retenir

- Virtualiser : présenter une ressource **logique** à la place d'une ressource **physique**, via une couche d'indirection
- Deux questions rangent n'importe quelle technique : **quoi, à quel niveau ?** et **quelle relation physique / logique ?**
- **Même interface** que la ressource physique — sinon ce n'est qu'une abstraction

Trois grandes familles

- **Émulation** : le jeu d'instructions — on traduit, et ça se paie
- **Machine virtuelle** : la machine matérielle — on partitionne, overhead quasi nul
- **Conteneur** : la vue du processus sur l'OS — on partitionne, en plus léger

À vous de jouer