

---

# Virtualisation



docker



Thierry Vaira  
LaSalle Avignon BTS SN IR

v0.1 22/05/2020

---

---

# Présentation

La virtualisation consiste à exécuter sur une machine hôte dans un **environnement isolé** :

- des systèmes d'exploitation : virtualisation système
- des applications : virtualisation applicative

Terminologie : serveur privé virtuel (*Virtual Private Server* ou VPS), environnement virtuel (*Virtual Environment* ou VE) ou machine virtuelle (*Virtual Machine* ou VM).

---

---

# Virtualisation système : terminologie

La virtualisation consiste donc à faire exécuter plusieurs noyaux en parallèle de façon isolé sur une même machine.

- Le système d'exploitation accueillant les VM est appelé **hôte**
- Les systèmes virtualisés sont appelés systèmes **invités** ou **VM**
- Le composant du système hôte réalisant la virtualisation se nomme l'**hyperviseur**.

Les systèmes invités et hôtes peuvent être totalement différents les uns des autres.

---

---

# Principe

La virtualisation respecte deux principes fondamentaux :

- Le **cloisonnement** : chaque système a un fonctionnement indépendant et ne peut interférer avec les autres.
- La **transparence** : le système ne change pas son fonctionnement.

---

# Avantages

- ★ la réduction des coûts d'acquisition d'une infrastructure
- ★ l'optimisation des ressources d'une infrastructure
- ★ l'amélioration de la disponibilité, sécurité, performance
- ★ la réduction des coûts de maintenance de cette infrastructure
- ★ la baisse de la consommation énergétique
- ★ la mise à disposition d'un environnement de tests à moindre coût,
- ★ la réutilisation de vieux systèmes incompatibles avec les nouvelles générations de matériel, etc.

---

# Inconvénients

- ❖ performances
- ❖ pannes
- ❖ mise en oeuvre, administration

---

# Historique

- 1960/1970 IBM et MIT
- 1980 Amiga
- 1990 / 2000 Émulateurs, VMware, VirtualBox, Xen, KVM, QEMU

---

# Types de virtualisation

## Virtualisation Complète

L'hyperviseur crée pour l'invité un environnement matériel virtualisé complet.

Le système invité s'installe et s'exécute sur le matériel présenté par l'hyperviseur comme s'il était "réel".

L'invité n'a pas "conscience" d'être virtualisé.

Limitations : Ce type de virtualisation ne permet de virtualiser que des systèmes d'exploitation prévus pour la même architecture matérielle que l'ordinateur hôte.

---



---

# Types de virtualisation

## Para-Virtualisation

L'hyperviseur peut pratiquer la para-virtualisation pour l'invité en lui fournissant un environnement matériel virtualisé complet sans le gérer complètement.

Le système invité peut accéder directement au matériel de l'hôte.

L'invité a donc "conscience" d'être virtualisé.

Avantages : plus performant

Limitations : moins isolé

---

---

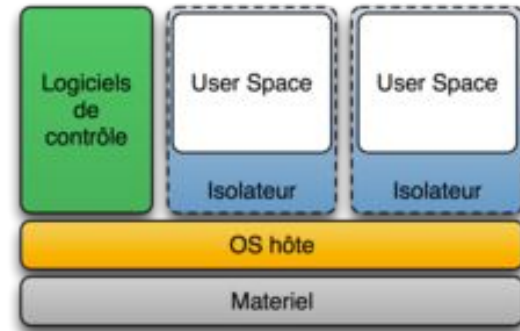
# Types de virtualisation

## Isolateur

Un isolateur est un logiciel permettant d'isoler l'exécution des applications dans ce qui est appelé des contextes, ou bien zones d'exécution.

Très performant et léger car l'isolateur n'embarque pas un système d'exploitation car il repose sur l'OS hôte sur lequel il s'exécute.

L'isolation est partielle.



---

# L'émulation

L'émulation consiste à simuler l'exécution d'un programme en interprétant chacune de ses instructions processeurs.

En théorie, il est possible d'émuler n'importe quel processeur et donc l'environnement complet d'une machine.

L'émulation est la technique qui offre le plus haut niveau d'abstraction de l'infrastructure mais les moins bonnes performances. Les autres techniques de virtualisation nécessite que l' "invité" soit compatible avec l'infrastructure.

QEMU est une solution open source de virtualisation par émulation.

---

---

---

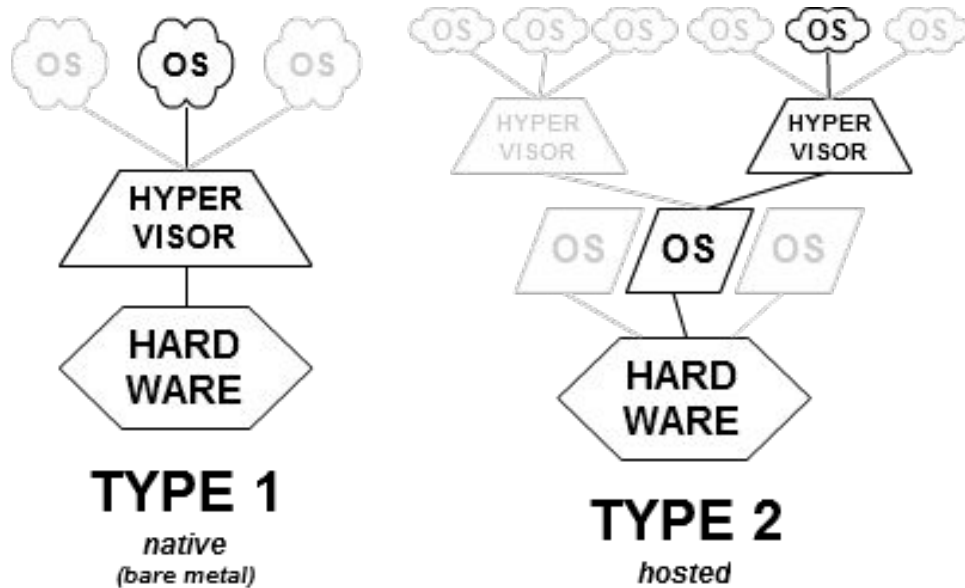
# Hyperviseur

Un hyperviseur est une plate-forme de virtualisation qui permet à plusieurs systèmes d'exploitation de travailler sur une même machine physique en même temps.

Les hyperviseurs sont classés actuellement en deux catégories : type 1 et type 2.

---

# Types d'hyperviseur



---

# Hyperviseur de type 1

Un hyperviseur de type 1 ou natif (voire « *bare metal* »), est un logiciel (un noyau allégé) qui s'exécute directement sur une plateforme matérielle.

Les ressources de l'hôte sont directement gérées par l'hyperviseur et non plus par l'OS. La totalité des ressources est donc dédiée aux VM.

Un seul hyperviseur par machine.

*Certains hyperviseurs de type 1 s'installent sur un OS qu'ils modifient pour qu'il devienne un VM.*

---

---

# Hyperviseur de type 2


Un hyperviseur de type 2 (ou *hosted hypervisor*) est un logiciel qui s'exécute à l'intérieur d'un autre système d'exploitation (hôte).

Un système d'exploitation invité s'exécutera donc en troisième niveau au-dessus du matériel. Les systèmes d'exploitation invités n'ayant pas conscience d'être virtualisés, ils n'ont pas besoin d'être adaptés.

---

# En résumé

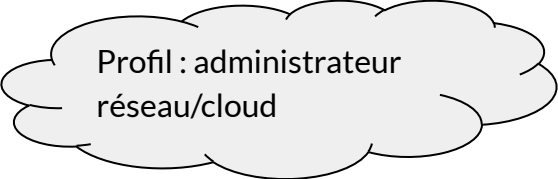
L'hyperviseur de type 2 : ◦ ◦ ◦



Profil : développeur,  
administrateur réseau

- Tester un OS
- Créer un environnement de test
- Développer une application et la tester sur différents systèmes

L'hyperviseur de type 1 : ◦ ◦ ◦



Profil : administrateur  
réseau/cloud

- Créer des serveurs
- Remplacer des machines physiques par des VM (réduction des coûts)
- Tests en environnement de pré-production



---

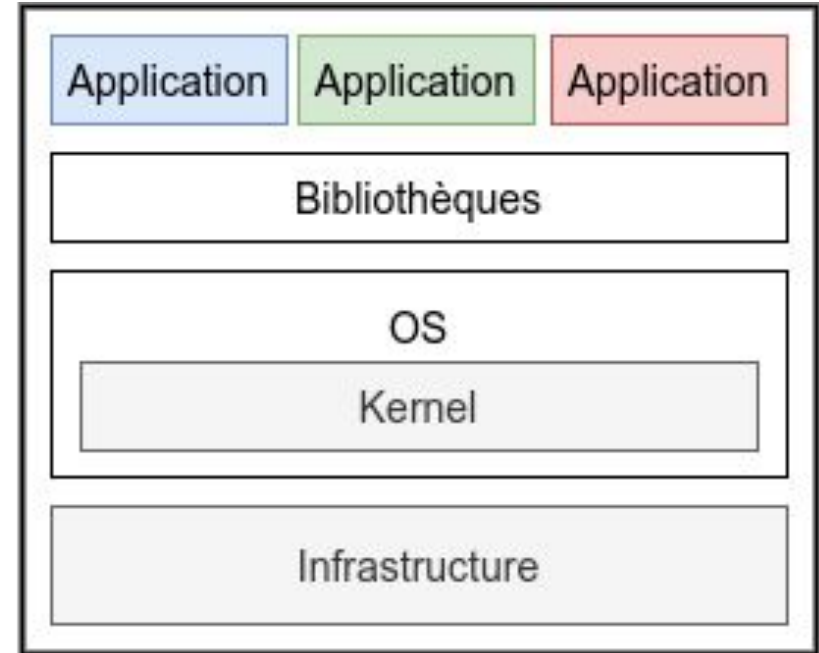
# Architecture traditionnelle

Dépendances entre les applications ?

Dépendances entre les applications et les bibliothèques ?

Dépendances entre les applications et le système d'exploitation ?

Accès aux ressources ?

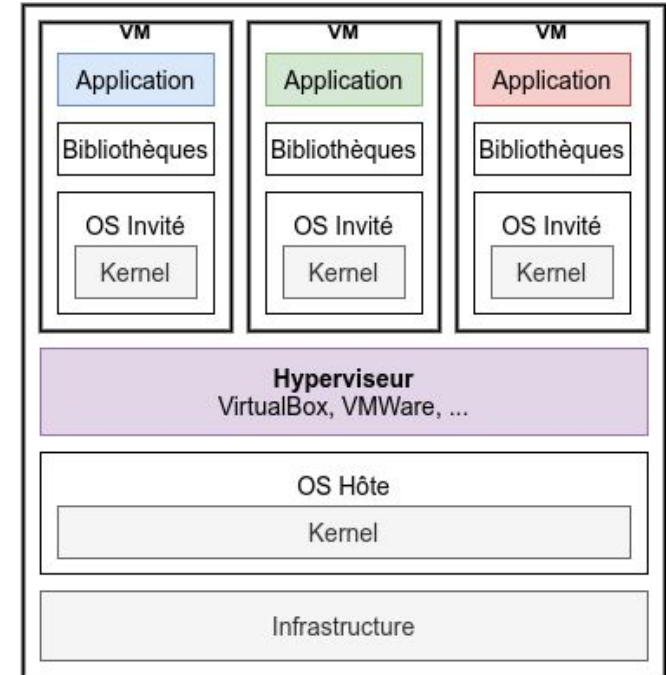


---

# Architecture Machine Virtuelle

Poids de la VM ? Performances ?

Configuration de la VM ?



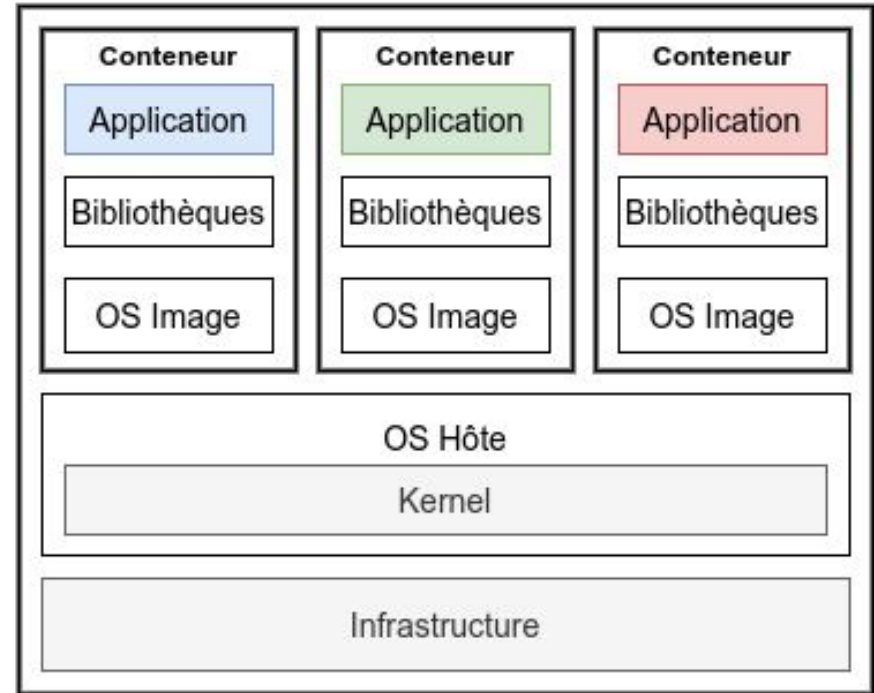
---

# Architecture Conteneur

Solution de compromis :

isolation avec les dépendances dont il a besoin

Très léger et performant vs VM



---

# Conteneur

Un **conteneur** est tout simplement un système de fichiers sur lequel s'exécutent des processus (de préférence un par conteneur) de manière :

- **contrainte** : on spécifie les limites en termes de ressources (cf. [cgroups](#))
- **isolée** : les conteneurs ne se “voient” pas les uns les autres (cf. [namespaces](#))

→ Travaux pratiques : [TP Docker](#)

---