R4.08 – Virtualisation

Virtualisation, machines virtuelles, hyperviseurs

Département Informatique

IUT2, UGA

2024/2025

Plan du cours

Plan du cours

Contexte

- Enseignements précédents utilisant la virtualisation
- SAÉ S1.03
  - Installation d'un poste de travail pour le développement
  - Utilisation de 2 machines virtuelles (VM) sur stations Linux
  - OS Debian 10 et Debian 11
- SAÉ S2.03
  - Installation d'un serveur Apache + PHP + PostgreSQL
  - Utilisation d'une VM Debian 11 sur stations Linux
- SAÉ S3.01
  - Utilisation de 2 VM Debian 12 sur serveur `assr`
  - Hébergement d'une appli Web complète

Objectifs de la ressource R4.08 et de la SAÉ S4.01

- Ici nous allons étudier la virtualisation **en elle-même**
- Panorama des différentes techniques de virtualisation
- Approfondissement des 2 techniques principales
  - machines virtuelles
  - conteneurs
- La ressource R4.08 contribue à la SAÉ S4.01
- SAÉ S4.01 parcours A
  - utilisation d'une machine virtuelle contenant le *backend* d'une application mobile et Web
  - travail à réaliser : faire tourner la partie serveur dans des conteneurs
- SAÉ S4.01 parcours B
  - utilisation intensive de machines virtuelles
  - nombreux services réseau à faire tourner dans des VM

Modalités

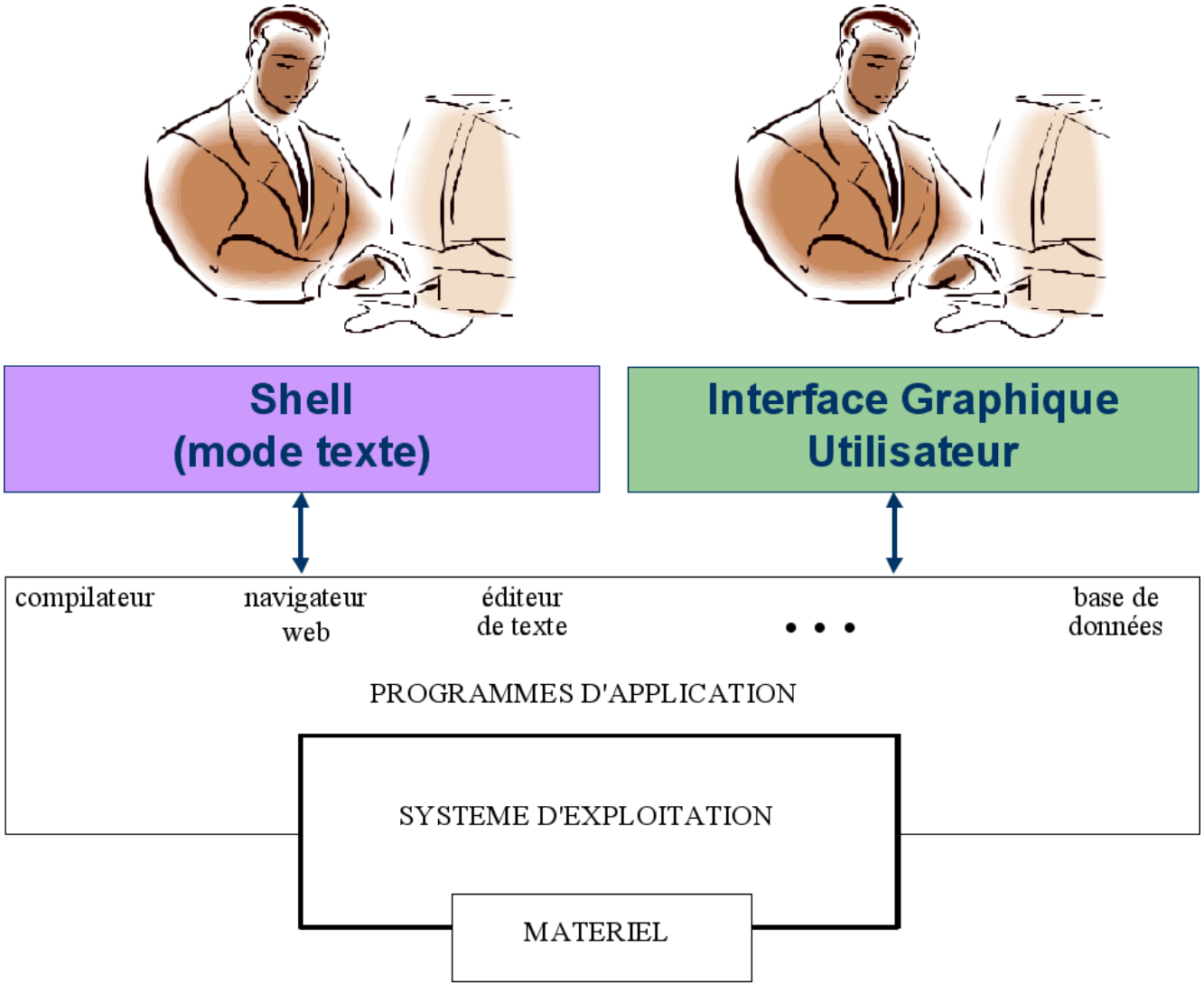
- 4 semaines
- Chaque semaine
  - un cours
  - un TP
- Utilisation de machines virtuelles (VM) hébergées sur le *cloud* E-Cloud de l'UGA

Plan du cours

Définition de «virtualiser»

- Transformer des ressources matérielles en ressources logiques plus faciles à gérer
- Exemple sur un poste de travail
  - installer un 2ème OS dans une machine virtuelle
  - plus simple et pratique que d'installer 2 OS sur une machine physique
  - plus simple
    - l'OS virtualisé est stocké dans une image disque
    - pas besoin de partitionner le stockage du poste
  - plus pratique
    - on peut faire tourner les 2 OS simultanément
    - pas besoin de rebooter sa machine

Rappel sur les OS



OS et virtualisation

- Rappel : notions vues pendant le module sur les OS (R3.05)
- Le CPU est partagé entre les applications, ce qui peut être vu comme de la virtualisation (*time-sharing*, *round-robin*)
- La RAM est virtualisée : chaque processus voit un espace mémoire virtuel, plus grand que l'espace mémoire physique (MMU)
- Le stockage est partagé entre les applications, les fichiers pouvant être vus comme une abstraction logique des secteurs physiques (SGF)

Différents types de virtualisation

- **Machines virtuelles** tournant grâce à un **hyperviseur**
  - virtualisation de serveur
  - virtualisation de poste de travail, d'application
- **Conteneurs** tournant grâce à un **gestionnaire de conteneurs**
  - service réseau tournant dans un conteneur
  - application tournant dans un conteneur
- Virtualisation de stockage
- Virtualisation de réseaux

Virtualisation de stockage I

- Les différents types de stockage
  - DAS : *Direct Attached Storage*
    stockage classique de type SATA/SAS ou NVMe
  - SAN : *Storage Area Network*
    stockage distant mis à disposition sous forme de **blocs**
  - NAS : *Network Attached Storage*
    stockage distant mis à disposition sous forme de **fichiers**
- Virtualisation de stockage dans un OS classique
  - Exemple 1 : Linux et LVM2
    volumes physiques (DAS) $\longrightarrow$ groupes de volumes $\longrightarrow$ volumes logiques
  - Exemple 2 : Windows et *Storage Spaces*

Virtualisation de stockage II

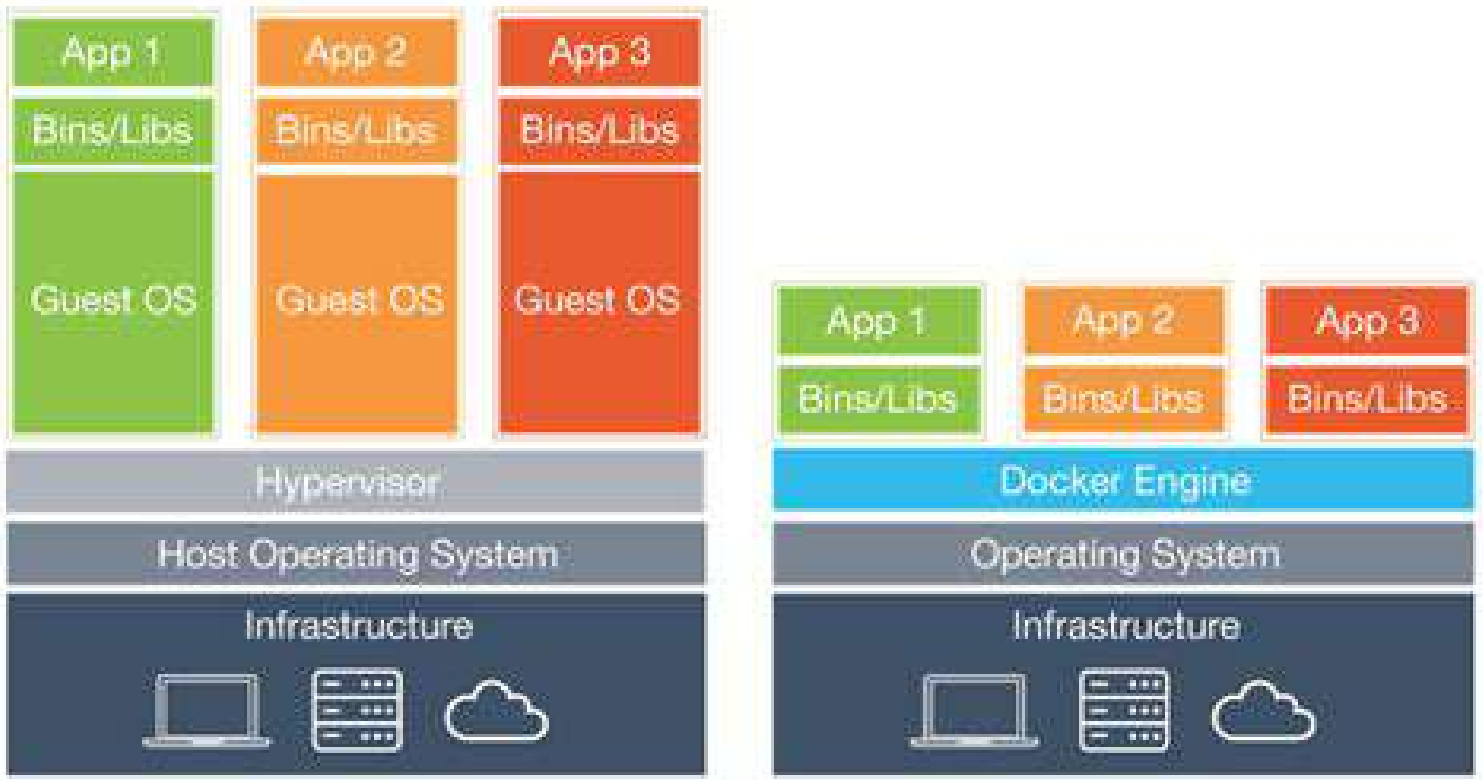
- Exemple de virtualisation avec NAS
  - machine sans stockage (*diskless*) qui boote par réseau sur un SGF NFS (*Network File System*)
- Exemples de virtualisation avec SAN
  - réseau de stockage sur infrastructure de type Fibre Channel, Ethernet ou Infiniband
  - protocole d'accès distant : iSCSI
    - SCSI sur IP
    - SCSI est le précurseur de SAS
  - ce stockage distant est mis à disposition d'un *cluster* de serveur physiques ou d'hyperviseurs

Virtualisation de réseaux

- VPN : *Virtual Private Network*
  - masquage d'adresse IP
    (*privacy*, contournement de restrictions d'accès)
  - travail à distance (domicile $\longrightarrow$ entreprise)
  - liaison sécurisé entre 2 sites d'une entreprise
- VLAN : *Virtual Local Area Network*
  - nombre de VLAN limité à 2^{12} (4096)
  - protocole Ethernet 802.1q
- VXLAN : *Virtual eXtensible LAN*
  - utilisé dans les (très) gros *clouds*
  - nombre de VLAN limité à 2^{24} ($\simeq$16 millions)
  - techniquement : Ethernet encapsulé dans UDP
- SDN : *Software-Defined Networking*
  - commutateurs virtuels (ex : Open vSwitch)
  - routeurs virtuels
  - pare-feu virtuels
  - exemple de protocole : OpenFlow

Plan du cours

Architecture logicielle des VM et conteneurs



Source : NetApp

Vocabulaire

- OS hôte (*host OS*) : l'OS principal de la machine
- OS invités (*guest OS*) : les OS tournant dans des VM

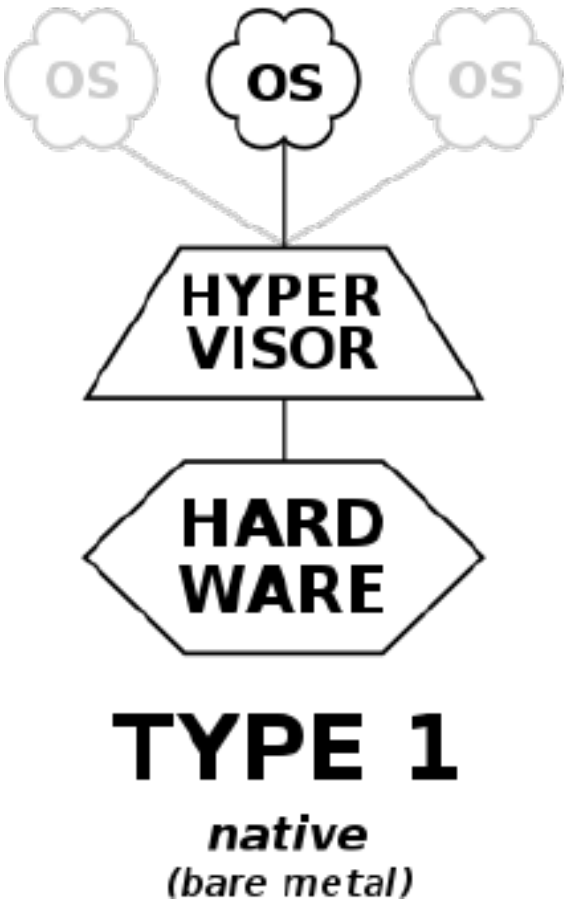
Points communs entre VM et conteneurs

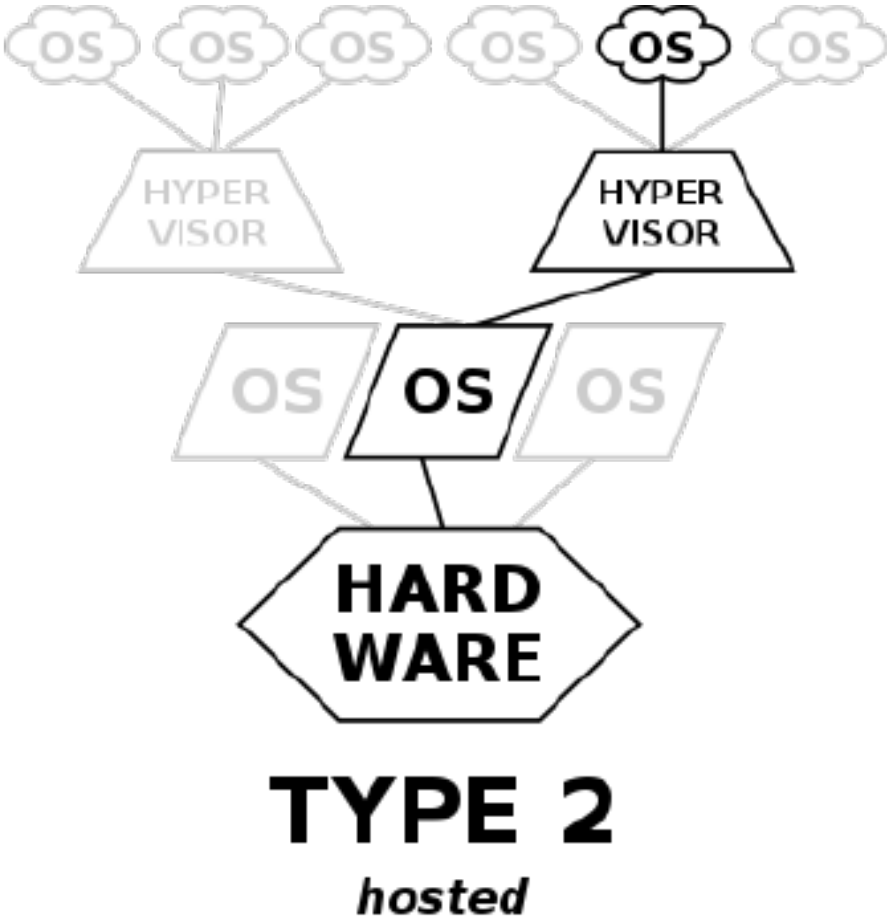
- L'architecture de CPU doit être la même
  x86-64/amd64, arm64, risc-v, ...
- Compartimentation, cloisonnement, isolation, étanchéité
  - Les VM/conteneurs sont isolés entre eux
  - L'OS hôte est isolé des VM/conteneurs
- Remarques sur la sécurité
  - ces propriétés sont des promesses en principe. Des failles de sécurité peuvent remettre en question cette isolation.
  - les VM/conteneurs ne sont pas isolées de l'OS hôte, ce qui est important du point de vue de la sécurité, quand on ne contrôle pas soi-même l'OS hôte (*cloud* public).
- VM et conteneurs contiennent tout ce qui est nécessaire au bon fonctionnement d'une application (logiciels, bibliothèques, ...) sans "polluer" l'OS hôte

Différences entre VM et conteneurs

- Les conteneurs n'utilisent pas d'OS invité
  (pas de noyau et pas de processus)
- Ils ne contiennent que des bibliothèques et des exécutables
- Conteneurs plus "légers"
  - moins de couches logicielles
  - moins de consommation de CPU
    $\longrightarrow$ plus de rapidité d'exécution
  - moins de consommation de RAM
    $\longrightarrow$ plus de conteneurs sur une même machine
  - moins de consommation de stockage
    $\longrightarrow$ plus de conteneurs sur un même stockage
- Conteneurs moins isolés que VM

Les 2 types d'hyperviseurs



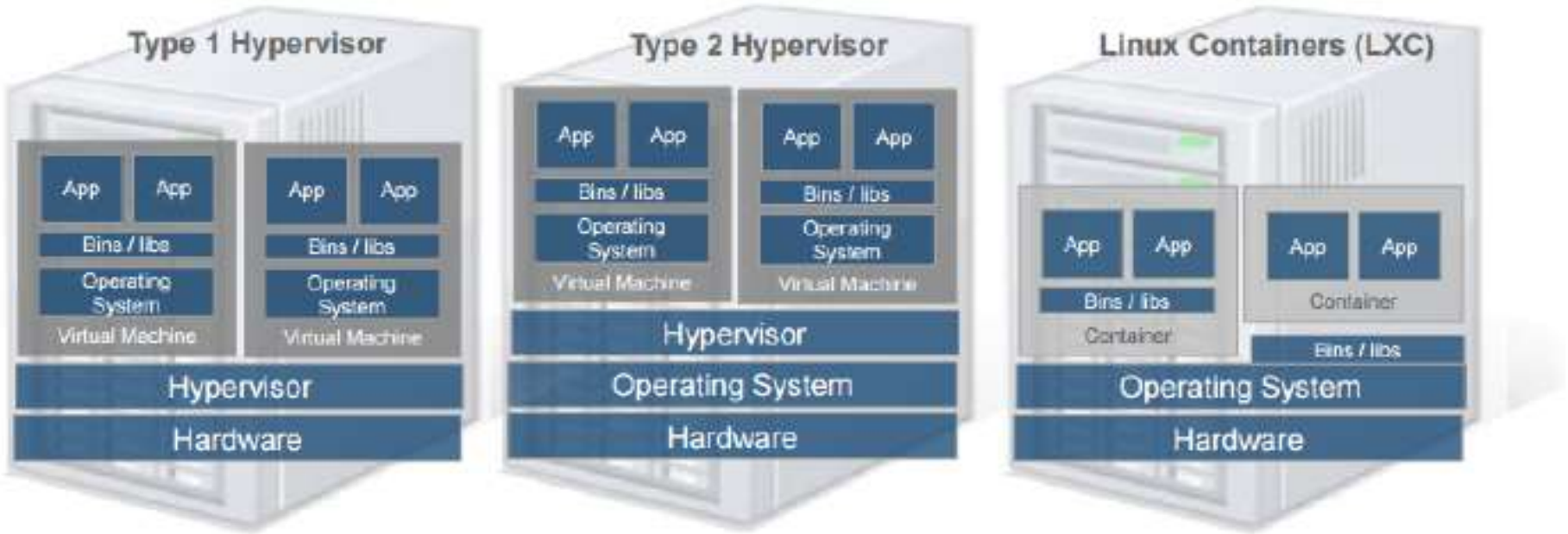


Source : Wikipedia

Les 2 types d'hyperviseurs

- Type 1
  - pas d'OS tournant sur la machine physique
  - l'hyperviseur tourne directement sur la machine physique (*bare-metal*)
- Type 2
  - un OS classique tourne sur la machine physique
  - l'hyperviseur tourne sur l'OS classique

Les 2 types d'hyperviseurs et les conteneurs



Source : supereon.co.uk

Virtualisation et Para-virtualisation

- Virtualisation complète
  - l'hyperviseur est capable de faire tourner n'importe quel OS
  - l'OS invité tourne sans modification
- Para-virtualisation
  - l'OS invité est modifié pour être adapté à un ou plusieurs hyperviseurs
  - cela permet d'améliorer les performances
  - dans le cas de Linux, le même noyau Linux peut être utilisé

Virtualisation et Para-virtualisation

Illustration avec système Linux

- Linux dans VM des SAÉ

```
root@debian:~# dmesg | grep virt
[    0.006658] Booting paravirtualized kernel on KVM
```

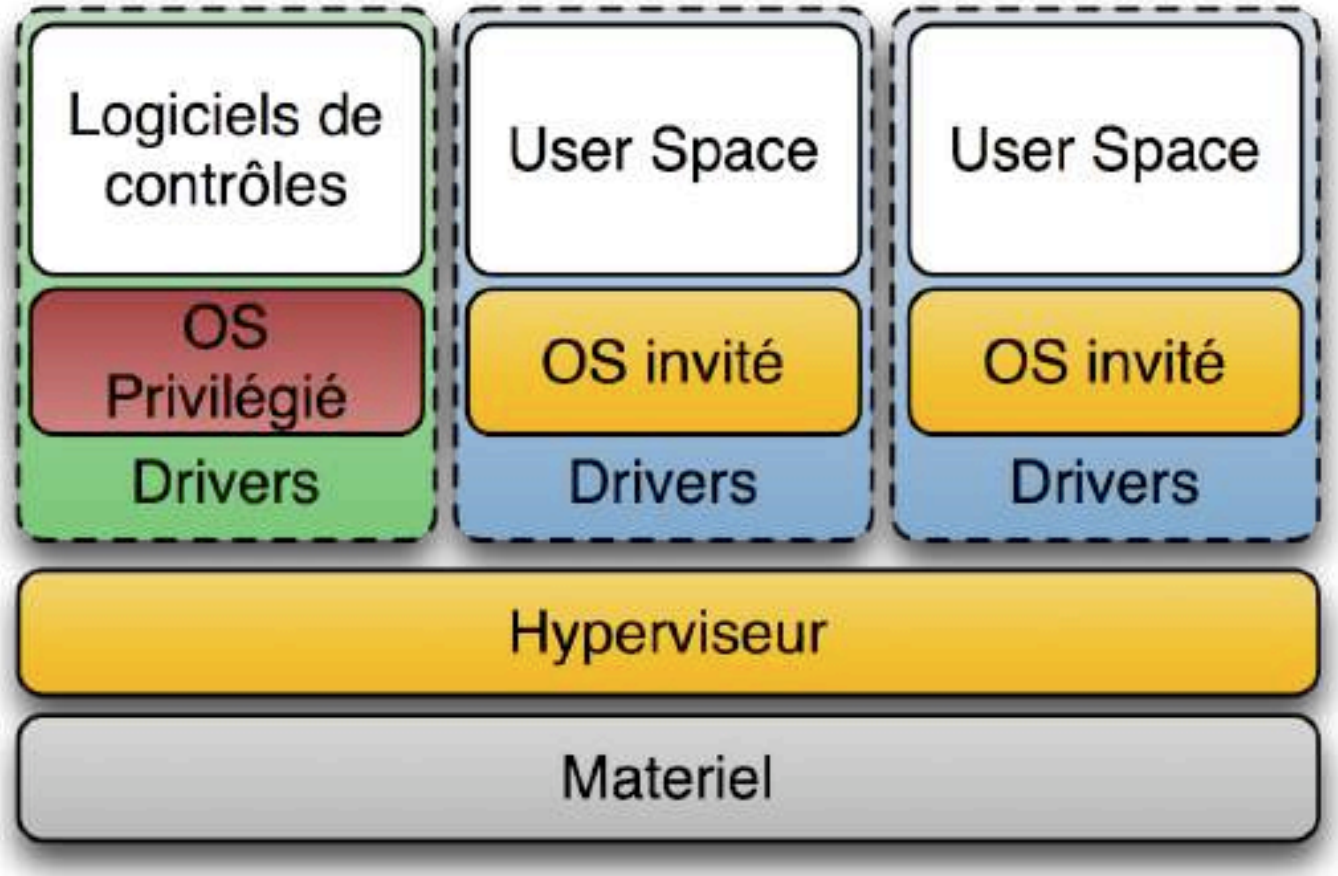
- Linux sur `transit`

```
root@transit:~# dmesg | grep virt
[    0.034035] Booting paravirtualized kernel on VMware hypervisor
```

- Linux sur station Linux

```
root@iut2-dgxxx:~# dmesg  | grep virt
[    0.015337] Booting paravirtualized kernel on bare hardware
```

Schéma détaillé d'un hyperviseur de type 1



Source : Wikipedia

Schéma détaillé d'un hyperviseur de type 2

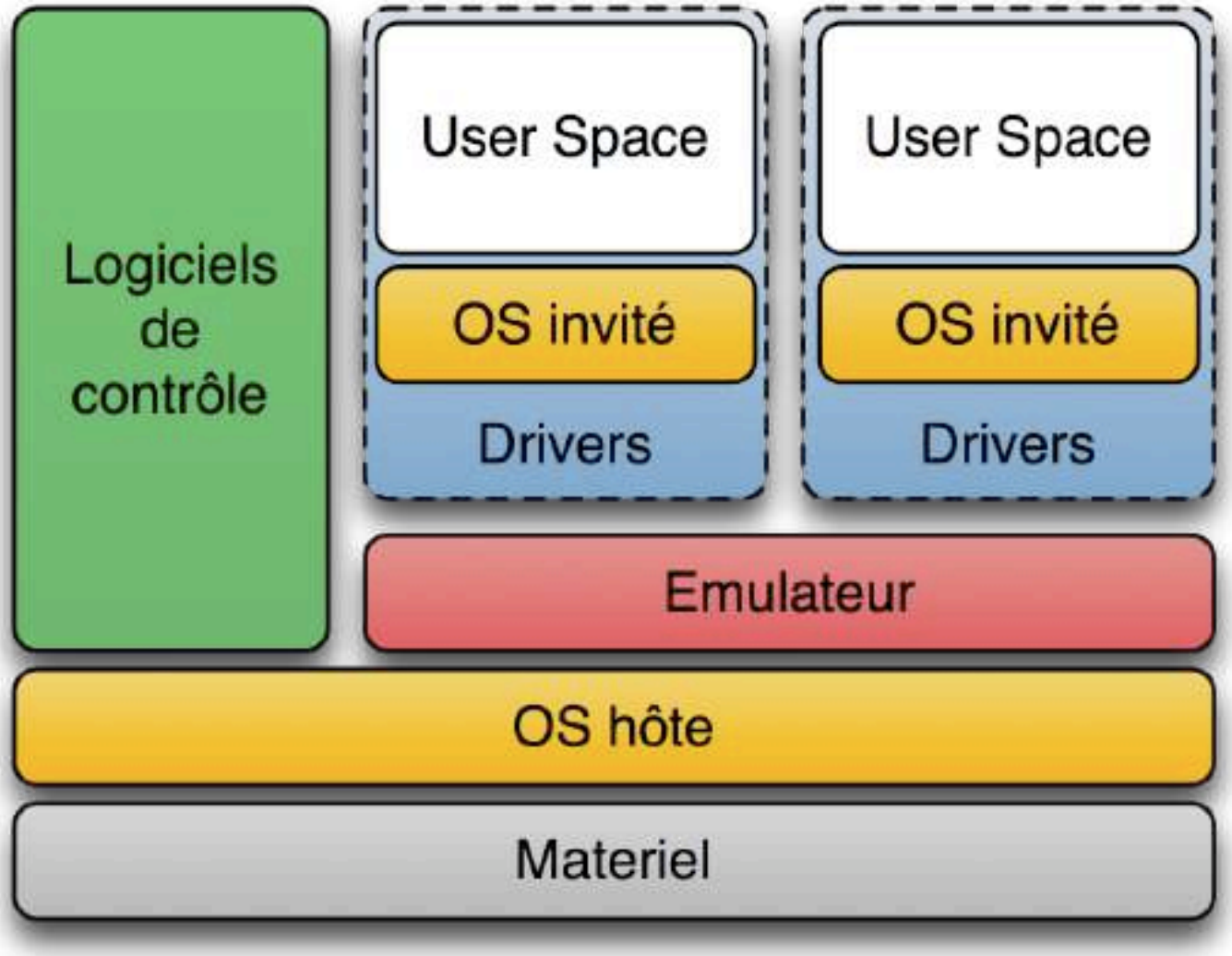
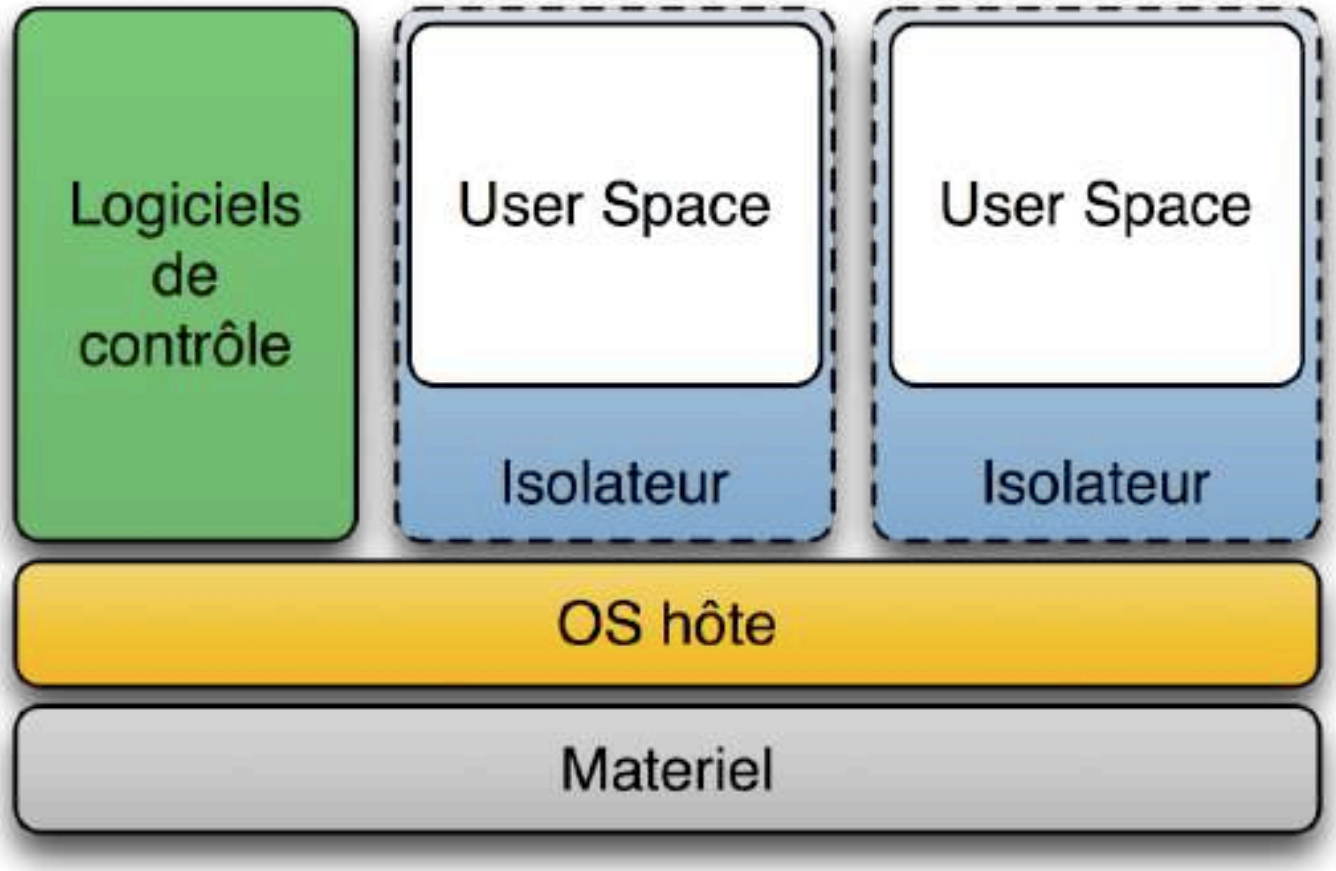


Schéma détaillé d'un gestionnaire de conteneurs



Source : Wikipedia

Plan du cours

Flexibilité dans l'allocation des ressources

- Certains serveurs ont de gros besoins en
  - CPU (cœurs) : serveur d'exécution
  - RAM : application gourmande ou mal optimisée
  - stockage : serveur de fichiers
- Dimensionner des serveurs physiques est assez complexe
- Avec des serveurs virtuels, on peut le faire à la demande
- On installe ses serveurs avec le minimum de ressources, puis on les fait grossir à la demande
- Upgrader un serveur physique est faisable, mais complexe
- Upgrader un serveur virtuel, se fait en quelques clics, ou changement de fichier de configuration

Moindre utilisation des ressources

- Pour une VM, on peut utiliser moins de CPU, RAM, stockage que le minimum existant sur un serveur physique
- Un serveur physique avec 1 seul cœur n'existe pas
- Un serveur physique avec 1 Go de RAM n'existe pas
- Un serveur physique avec 2 Go de stockage n'existe pas
- Alors que ces valeurs peuvent suffire pour une VM

⟶ Économie de ressources

Partage/Mutualisation des ressources

- n serveurs virtuels ont besoin de moins de matériel que n serveurs physiques
- On utilise moins de
  - cartes mères
  - boîtiers physiques, alimentations électriques
  - espace d'hébergement (nombre de "U" dans un rack)
  - énergie électrique

$\longrightarrow$ Optimisation de l'utilisation des ressources

$\longrightarrow$ Réduction des coûts

Sécurité par isolation

- Comparaison
  1. 1 serveur physique avec n services réseau ou applications (ex : *Virtual Hosting* pour le Web)
  2. n serveurs virtuels avec chacun 1 service ou application
  3. n serveurs physiques avec chacun 1 service ou application
- On augmente la sécurité, mais aussi le coût
- Des serveurs virtuels peuvent être considérés comme un bon compromis entre coût et sécurité

Haute disponibilité

- Une VM peut être transférée d'un hyperviseur à un autre

⟶ Meilleure disponibilité qu'une machine physique
Une VM peut être considérée comme "immortelle"

- Le transfert peut se faire sans arrêter la VM

⟶ Excellente disponibilité

- Une VM boote beaucoup plus vite ($\sim$5 secondes) qu'un serveur physique ($\sim$5 minutes)

⟶ Minimisation des temps d'arrêt

Sauvegarde et restauration

Machine physique

- Sauvegarder un OS physique nécessite quelques précautions (SGF virtuels, fichiers en cours d'écriture, ...)
- Restaurer nécessite pas mal de connaissances
  - recréer la table des partitions
  - recréer les volumes LVM et/ou RAID
  - recréer les SGF
  - rendre le système bootable
  - ...

Machine virtuelle : il suffit de

- faire un *snapshot* de la VM
- sauvegarder un seul fichier : l'image disque

Autres avantages

- Gestion centralisée
  Un hyperviseur permet de gérer les machines virtuelles de manière centralisée, ce qui facilite la gestion et la maintenance des VM.
- Support multi-OS
  - Un hyperviseur permet de faire fonctionner plusieurs OS sur un seul ordinateur physique.
  - Des conteneurs permettent d'utiliser plusieurs distributions Linux, ou plusieurs installations d'une même distribution.

Quelques inconvénients de la virtualisation

Performances

- Serveur virtuel en général plus lent que serveur physique
- Mais les processeurs ont maintenant des fonctionnalités et extensions de leur jeu d'instruction pour accélérer matériellement l'exécution des VM
  - Intel : VT-x/VMX (Virtual Machine eXtensions), VT-d (IO-MMU)
  - AMD : AMD-V

⟶ Performances en calcul quasiment identiques

- Faible différence de performances sur les entrées/sorties

Sécurité

- On est soumis aux failles de sécurité dans les hyperviseurs
- Impact d'une faille dans un hyperviseur : catastrophique !

Plan du cours

Principaux hyperviseurs existant I

- VMWare : server, workstation, ESXi, VSphere, ...
  - type 1, type 2
  - utilisé par certains ”petits” fournisseurs de *cloud* (OVH)
- Linux Foundation : Xen
  - type 1
  - utilisé dans les *clouds* d'Amazon (AWS), Alibaba, ...
- Microsoft : Hyper-V
  - type 1, type 2, conteneurs
  - utilisé dans le *cloud* de Microsoft (Azure)

Principaux hyperviseurs existant II

- Qemu/KVM
  - type 2
  - Qemu fonctionne sur Linux, MacOS, Windows
  - KVM intégré au noyau Linux
  - KVM spécifique au noyau Linux
  - KVM utilisé dans le *cloud* de Google (GCE/GCP)
- VirtualBox
  - type 2
  - fonctionne sur Linux, MacOS, Windows
  - driver Linux développé en dehors du noyau Linux
- ...

Critères de choix d'un hyperviseur

- Architecture de l'hyperviseur
  - type 1 : plutôt pour une machine dédiée à la virtualisation
  - type 2 : convient mieux pour un poste de travail
- Sécurité : faire des statistiques sur les failles CVE
- Licence libre ou propriétaire
- Facilité d'utilisation
- Maintenance
- Performances : lire ou faire des *benchmarks*
- ...

Plan du cours

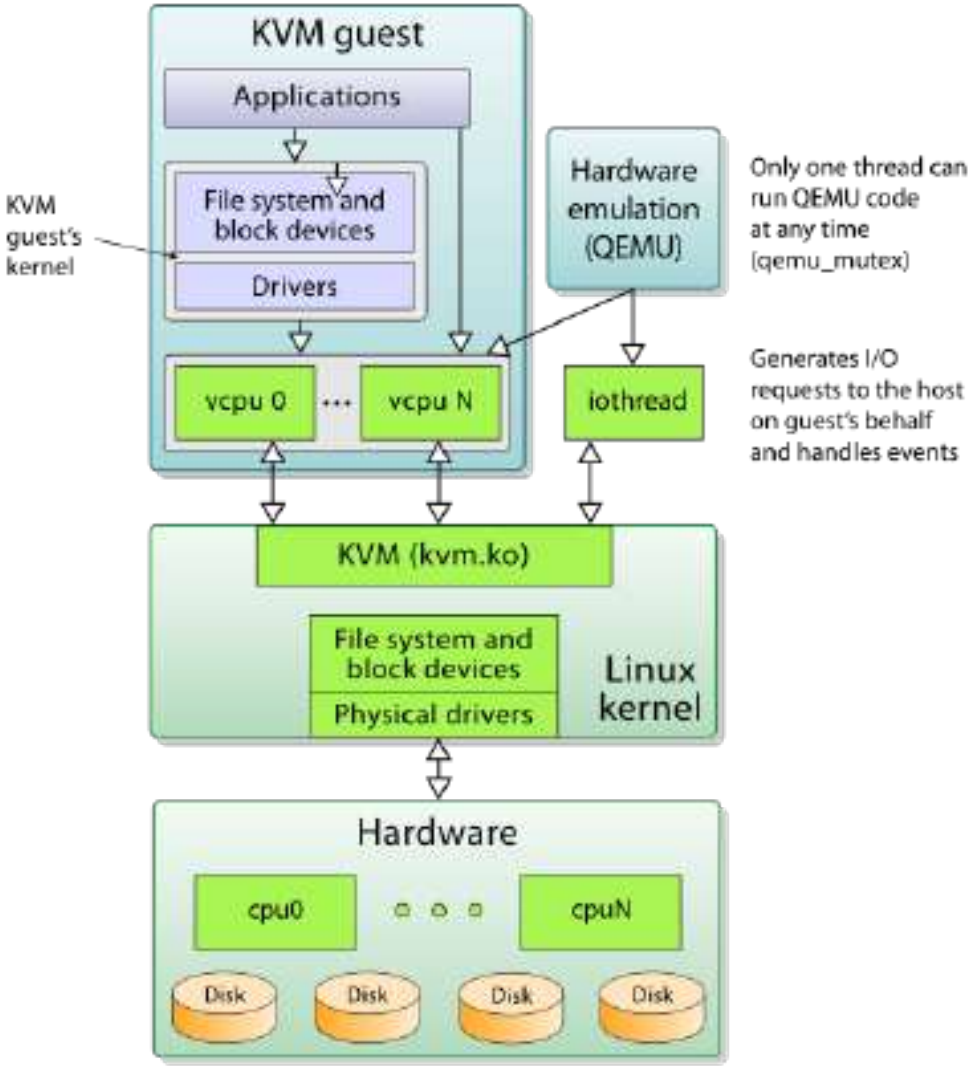
Présentation de Qemu

- À l'origine, c'est un émulateur de processeur
- Principe
  - prendre un OS (ou un exécutable) compilé pour une architecture X
  - le faire tourner sur un CPU d'architecture Y
- Émule aussi un grand nombre de composants matériels classiques
  - cartes réseau
  - cartes graphiques
  - ...
- Capable d'utiliser la virtualisation matérielle pour accélérer l'émulation (dans le cas où X=Y)

Présentation de KVM

- KVM : *Kernel Virtual Machines*
- Attention à ne pas confondre avec les autres KVM !
- Lien avec Qemu
- Fait partie du noyau Linux
- Démonstration
  - modules `kvm` sur disque
  - modules `kvm` chargés dans le noyau Linux
- Démonstration d'OS virtualisés : Windows, MacOS

Architecture complète Qemu/KVM



Source : Wikipedia

Mise en œuvre manuelle

- Packages à installer : `qemu-system-*`
- Autorisation : permissions de `/dev/kvm`
  - ajout au groupe `kvm` (utiliser `adduser`)

    ```
    crw-rw----+ 1 root kvm 10, 232 Feb  5 20:08 /dev/kvm
    ```
  - changer les permissions Unix (fait sur les stations Linux)

    ```
    crw-rw-rw-+ 1 root kvm 10, 232 Feb  5 20:08 /dev/kvm
    ```
  - ACL (*Access Control List*) à vérifier avec `getfacl`
- Gestion des images disque : `qemu-img`
- Lancement des VM : `qemu-system-x86_64` (+ options)

Plan du cours

Surcouches de Qemu/KVM

- Qemu/KVM peut être utilisé en ligne de commande (CLI)
- Des surcouches fournissent des interfaces de type GUI, Web ou CLI de plus haut niveau

Principales surcouches

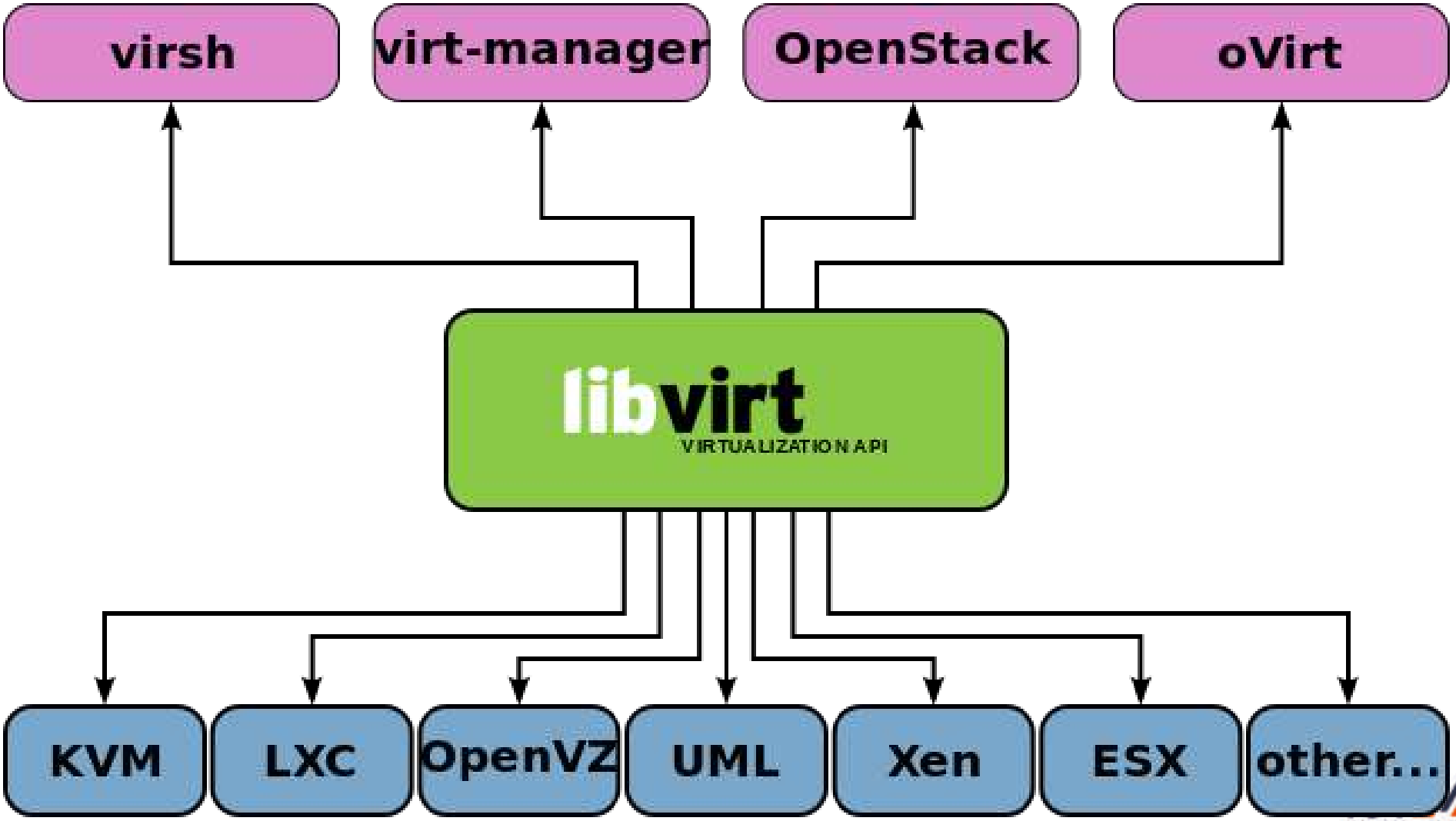
- Libvirt
  - virt-manager : GUI
  - virsh : CLI
- Proxmox : Web + CLI
- OpenStack : Web + CLI
- oVirt : Web + API REST + CLI
- Ganeti : CLI, Web (fourni par des tiers : Ganeti Web Manager)
- ...

Plan du cours

Présentation de `Libvirt`

- API et *middleware* entre
  - interfaces graphiques (GUI) ou ligne de commande (CLI)
  - hyperviseurs et/ou conteneurs
- Facilite
  - gestion des disques virtuels
  - mise en réseau des VM
  - accès aux consoles des VM
  - accès transparent à un hyperviseur distant à travers SSH
  - migration à chaud des VM entre plusieurs hyperviseurs
  - ...

Architecture de `Libvirt`



Source : Wikipedia

Présentation de virt-manager

- Interface graphique pour `Libvirt`
- Donc en particulier pour Qemu/KVM
- Démonstration
  - accès facile aux consoles
  - monitoring CPU, RAM, E/S stockage, trafic réseau

Présentation de `virsh`

- Interface CLI *(shell)* pour `Libvirt`
- Permet d'automatiser par des scripts toute la gestion des VM
  - création
  - installation
  - mise en réseau
  - démarrage et arrêt
  - mise en pause et reprise
  - ...

Mise en œuvre de Libvirt

- Packages à installer
  - `libvirt-*`
  - `virt-manager`
- Démon `libvirtd` tournant sur les machines hôte
- 2 modes de fonctionnement
  - mode privilégié (appelé ”QEMU/KVM”)
  - mode non-privilégié (appelé ”QEMU/KVM User session”)
- Mode privilégié
  - donne accès à des mises en réseau qui nécessitent des privilèges root
  - ne doit être autorisé qu’aux personnes de confiance (équivalent à un accès root)
  - autorisation par ajout au groupe `libvirt`

Plan du cours

Présentation de Proxmox

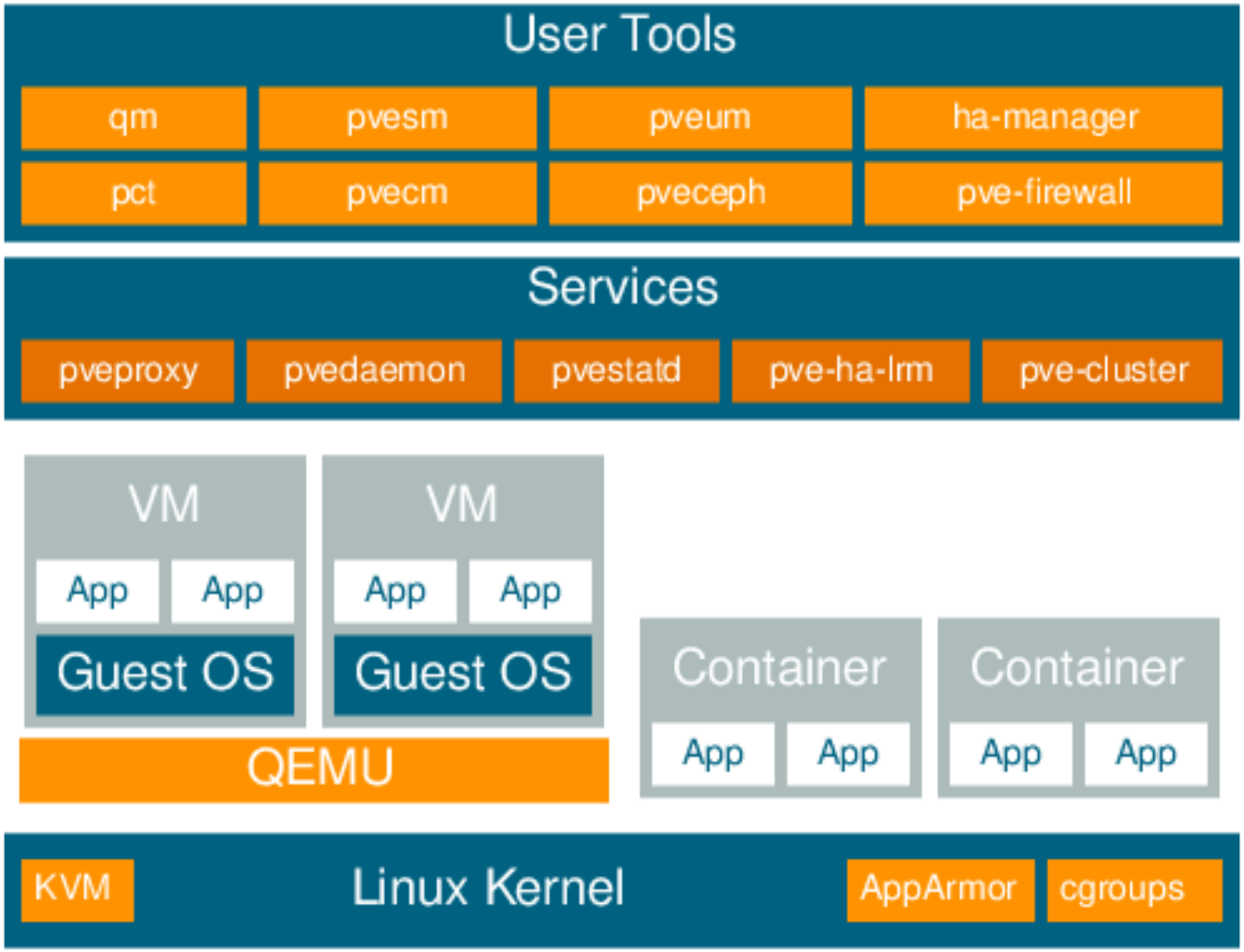
- OS complet basé sur Debian et intégrant Qemu/KVM
- Interface Web
- Gestion de *clusters* d'hyperviseurs
- SGF distribué (ceph)
- Gestion de conteneurs (LXC)
- ...

Mise en œuvre de Proxmox

Installation : 2 possibilités

- On part d'une image ISO intégrant tout ce qui est nécessaire
- On part d'un système Debian et on ajoute des packages supplémentaires

Architecture de Proxmox



Source : documentation de Proxmox

Plan du cours

Principaux modes

- Mode *bridge*
  - VM sur le même réseau physique que l'hôte
  - communications VM $\longleftrightarrow$ extérieur naturelles
  - mode utilisé pour les serveurs virtuels de S3.01
- Mode NAT (*Network Address Translation*)
  - VM sur un réseau virtuel (privé) isolé
  - communications VM $\longrightarrow$ extérieur par routeur NAT
  - communications extérieur $\longrightarrow$ VM par redirection de ports ou SNAT
  - mode utilisé pour
    - postes de travail virtuels de S1.03
    - serveurs virtuels de S2.03
  - NAT effectué soit par Qemu, soit par le noyau Linux de l'hôte

Plan du cours

Mode bridge : configuration IP des VM

2 modes

- Configuration manuelle
  - adresse IP fixe
  - dans fichier de configuration de la VM
- Configuration par serveur DHCP
  - serveur DHCP à faire tourner sur une autre machine
  - machine physique ou virtuelle

Mode bridge : interfaces de la machine hôte

Sur la machine hôte, un *bridge* Ethernet (`br0`) relie

- l'interface physique `eno1`
- les interfaces virtuelles `tap???`

```
root@assr:~# brctl show
bridge name     bridge id               STP enabled     interfaces
br0             8000.162222f109c7       no              eno1
                                                        tap210
                                                        tap211
                                                        tap212
[...]
```

Mode bridge : configuration IP de la machine hôte

L'interface physique n'a plus d'adresse IP

```
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        ether 4c:d9:8f:8a:e1:58  txqueuelen 1000  (Ethernet)
```

Le bridge récupère l'adresse IP

```
br0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.14.3  netmask 255.255.255.0  broadcast 19
        inet6 fe80::1422:22ff:fef1:9c7  prefixlen 64  scopeid
        ether 16:22:22:f1:09:c7  txqueuelen 1000  (Ethernet)
```

Plan du cours

Mode NAT par défaut de Qemu

- Chaque VM a son propre réseau privé (`10.0.0.x`)
- Les VM sont isolées entre elles
- Le serveur DHCP est intégré à Qemu
- Les VM ont accès au réseau extérieur par NAT
- NAT assuré par Qemu
- Une VM peut être joignable depuis la machine hôte par redirections de ports
- Redirections de ports à configurer manuellement
  (options sur la ligne de commande Qemu, vues dans S2.03)

Mode NAT par défaut de `Libvirt` en mode privilégié

- Toutes les VM sont dans un même réseau privé (`192.168.122.x`)
- Les VM peuvent communiquer entre elles
- Le service DHCP est assuré par le logiciel `dnsmasq`
- `dnsmasq` est lancé automatiquement par `libvirt`
- Les VM ont accès au réseau extérieur par NAT
- NAT assuré par le noyau Linux de la machine hôte (`netfiler`)
- Les VM sont joignables depuis la machine hôte par redirections de ports
- Redirections de ports configurées automatiquement par `libvirt`

Mode SDN de Proxmox

- Pas de réseau par défaut : il faut en configurer un ou plusieurs
- Pas d'adresses par défaut : il faut les choisir
- Le serveur DHCP est assuré par le logiciel `dnsmasq`
- `dnsmasq` est lancé par Proxmox
- Les VM ont accès à l'extérieur par NAT
- Les VM sont joignables depuis la machine hôte par SNAT
- SNAT assuré par le noyau Linux de la machine hôte (`netfiler`)

Plan du cours

Résumé

- Différences entre VM et conteneurs
- Techniques puissantes avec de nombreux avantages
- Nombreux logiciels existant
- Différentes possibilités de mise en réseau
- Domaine très riche
- Nécessite des connaissances en système et en réseau