Introduction à la virtualisation
LXC & LXD

Lucas Nussbaum

lucas.nussbaum@univ-lorraine.fr

Licence professionnelle ASRALL

Administration de systèmes, réseaux et applications à base de logiciels libres

Virtualisation

Principe : faire fonctionner plusieurs systèmes d'exploitation sur un seul ordinateur comme s'ils fonctionnaient sur des ordinateurs distincts

Objectifs :

- Réduire les coûts
  - Moins de serveurs nécessaires
  - Serveurs mieux utilisés
- Faciliter l'administration
  - Chaque service isolé sur son propre serveur
  - Découpler le cycle de vie du matériel de celui des services
  - Découpler le matériel des services $\rightsquigarrow$ *Software-Defined Infrastructure*

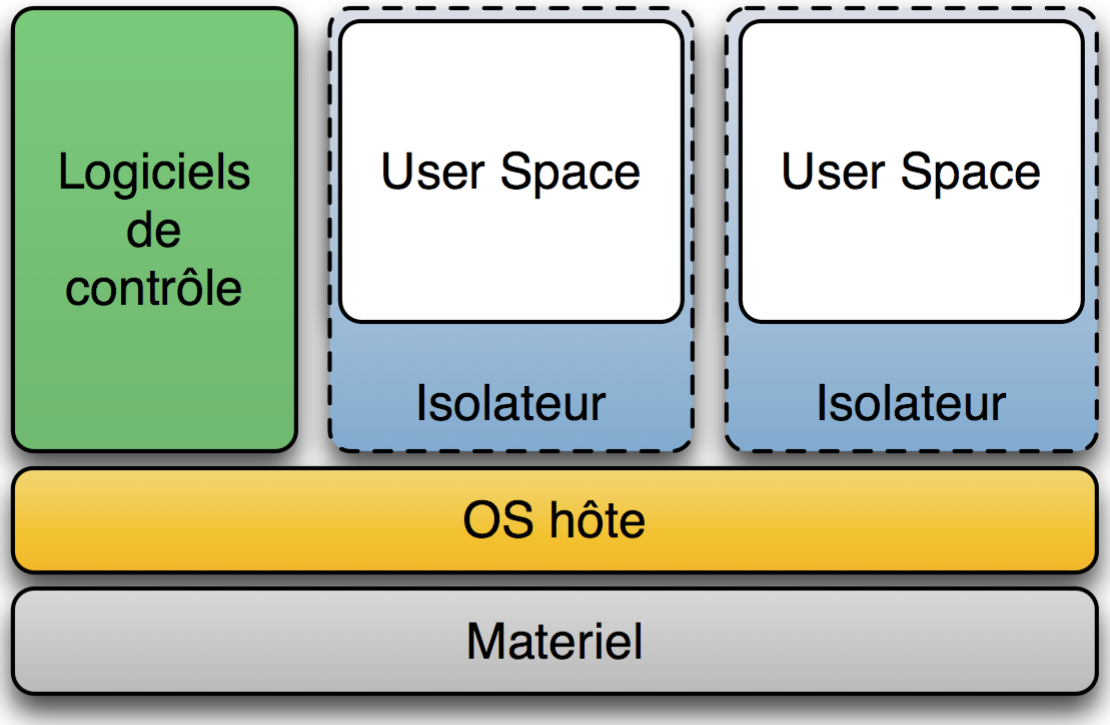
  Mais il faut être capable de gérer de nombreux serveurs !

Deux grandes familles de virtualisation :

- Containers
- Hyperviseurs et machines virtuelles (VM)

Containers (*OS-level virtualization*)

- Un seul noyau, plusieurs espaces utilisateurs (*contextes*)
- *chroot*, Linux-VServer, OpenVZ, Linux Containers (LXC), BSD Jails, Docker



(images Wikipédia)

LXC & Docker – technique

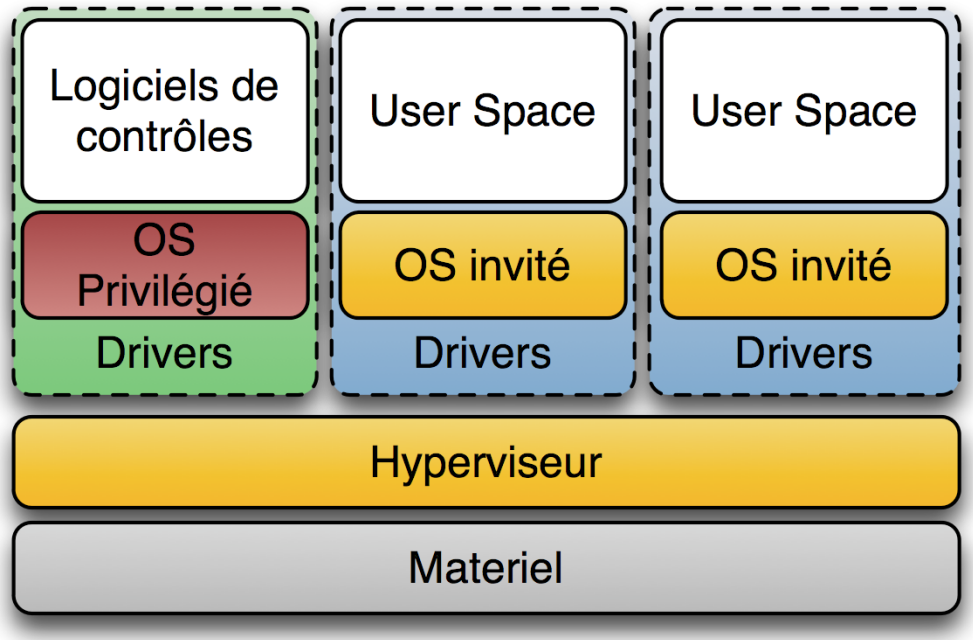
- Base technique : fonctions du noyau Linux
  - *Control groups (cgroups)* : groupes de processus
  - *Namespaces* : limite l'accès et la visibilité des ressources
    - `sudo unshare -npf -mount-proc bash` $\rightsquigarrow$ lancer un shell dans un cgroup avec des nouveaux namespaces PID et NET
    - Namespaces Network, Mount, PID, User, UTS, IPC, Cgroup
  - Containers (Docker, LXC) = *cgroups* + *namespaces* + interface

LXC & Docker – positionnements

- **LXC : conteneurs systèmes**
  - Usage similaire à des machines virtuelles, mais technologie différente
  - Système complet : init, plusieurs services, etc
- **Docker : conteneurs applicatifs**
  - Manière différente de packager, diffuser, exécuter des applications
  - Système très minimal (souvent pas d'init), application (service), dépendances logicielles
  - Interagira avec d'autres services dans d'autres conteneurs
    - Exemple : 1 conteneur pour Apache, 1 autre pour MySQL

Hyperviseurs

- Un noyau par système hébergé
- L'hyperviseur agit comme intermédiaire entre le système invité (*guest*) et le matériel
- Distinction entre hyperviseurs de Type 1 et de Type 2
  - Floue et peu utile en pratique [1]
- Xen, KVM, VMWare, VirtualBox



1. `http://blog.codemonkey.ws/2007/10/myth-of-type-i-and-type-ii-hypervisors.html`

Xen

- Un des projets les plus anciens, la solution libre la plus utilisée
- Vocabulaire :
  - OS privilégié = Dom0
  - OS invité = DomU
- Deux modes d'utilisation
  - Paravirtualisation
    - Noyau spécifique dans le domU
    - Très bonnes performances
  - Virtualisation matérielle
    - Virtualisation transparente pour le système invité
    - Besoin d'un support dans le processeur (AMD-V, Intel VT)
- En cours d'intégration dans le noyau Linux

KVM

- Projet un peu plus jeune (2006), mais de plus en plus populaire
- Basé en partie sur Qemu (émulateur) pour le support des périphériques
- Entièrement intégré dans le noyau Linux $\rightarrow$ facile à utiliser
- Support de la virtualisation dans le processeur obligatoire
  - Ne fonctionne pas sur les processus anciens
- Paravirtualisation (virtio) pour les performances (réseau, disque)

Containers vs VM : avantages & inconvénients

- Partage facile (plus fluide) de ressources entre containers : espace disque, mémoire, caches
  - Plus de containers par serveur physique (densité)
- Couche de virtualisation plus légère
  - Meilleures performances, quasi équivalentes à une machine physique
- Mais isolation plus faible qu'avec des VM
  - Plus difficile de contrôler l'utilisation des ressources
  - Surface d'attaque plus importante $\rightsquigarrow$ sécurité
- Démarrage et instanciation des containers plus rapide (quelques secondes vs quelques dizaines de secondes)
- Forte adhérence entre l'OS du container et l'OS hôte
- La migration en cours d'exécution fonctionne moins bien

Plusieurs échelles pour la virtualisation

- Un seul serveur
- Quelques serveurs
  - Géré avec des solutions légères comme `libvirt` (`virsh`, `virt-manager`)
- Toute une infrastructure
  - Y compris le stockage (centralisé ou distribué) et le réseau
    $\rightsquigarrow$ *infrastructure hyper-convergée*
  - Solutions : Ganeti, oVirt, Proxmox
  - Gestion centralisée, haute disponibilité intégrée, etc.
- Cloud (virtualisation multi-*tenants*)
  - Privés (internes à l'organisation) : OpenStack, CloudStack
  - Publics : Amazon Web Services (pdm : 41.5%), Microsoft Azure (29.4%), Google Cloud Platform (3.0%), OVH
    - ★ Crédits gratuits pour étudiants (par exemple `http://www.awseducate.com/`)

Connexion au réseau

La machine virtuelle exporte une interface virtuelle sous forme de *tap* dans l'hôte, puis différents modes de fonctionnement possibles :

- Le système hôte agit comme un routeur (niveau 3)
  - Éventuellement avec du NAT
- Le système hôte agit comme un bridge/pont (niveau 2)
  - Les machines virtuelles sont directement visibles sur le réseau physique
- Disponible dans KVM uniquement :
  NAT en mode utilisateur

Cloud Computing

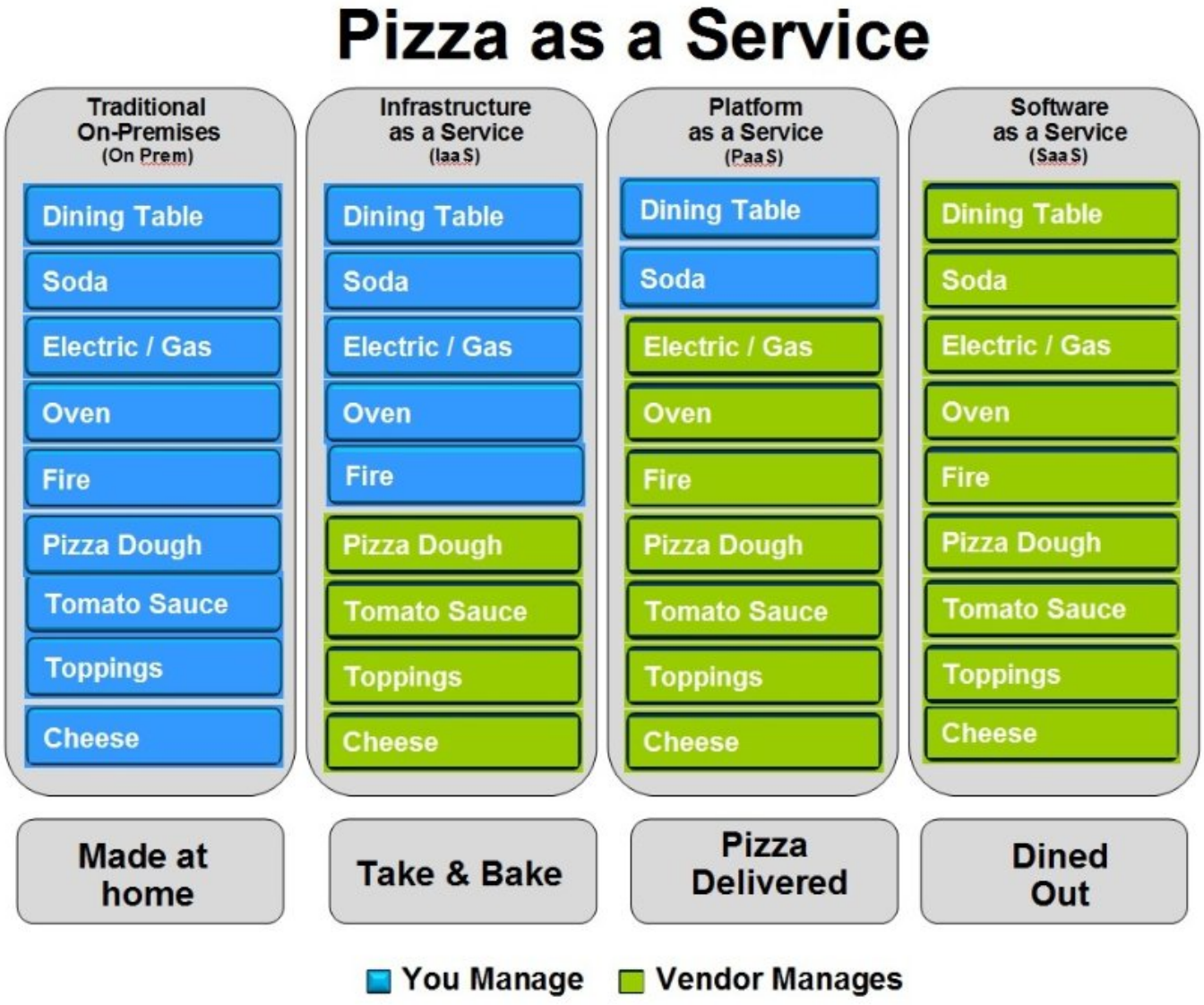
Principe de base : **l'informatique comme un service**

- Hébergement local $\rightsquigarrow$ externalisation
- Automatisation, élasticité

Différents types de Cloud :

- **IaaS** (Infrastructure as a Service) : machines virtuelles, stockage
  - Amazon Web Services, Google Cloud Platform, Microsoft Azure
  - Anciennement : Dedibox, Linode
- **PaaS** (Platform as a Service) : déploiement facilité d'applications pour une certaine plate-forme. intégration avec serveur de base de données.
  - Google AppEngine (Python), Heroku (Rails), Azure (ASP.NET)
  - Anciennement : mygale.org ☺, pages perso de Free ?
- **Saas** (Software as a Service) : application hébergée
  - Google Apps (Docs), Salesforce.com
  - Anciennement : Wordpress.com ?

Cloud & Pizzas



source : `https://www.linkedin.com/pulse/20140730172610-9679881-pizza-as-a-service`

Cette séance : LXC, puis LXD

- Installez LXC : `apt-get install lxc`
- Vérifiez que votre système a bien le support nécessaire pour LXC : `lxc-checkconfig`
- Créez un conteneur : `lxc-create -n monconteneur -t debian`
- Consultez le résultat, dans `/var/lib/lxc/`, et en particulier le fichier de configuration du conteneur.
- Consultez la documentation (par exemple `http://wiki.debian.org/LXC`), et
  - Démarrez le conteneur en arrière-plan (détaché)
  - Connectez-vous à la console du conteneur
  - Vérifiez que vous pouvez vous connecter en *root*
  - Déconnectez-vous de la console
  - Arrêtez le conteneur
- Même s'il est plus basique que celui de Docker par exemple, LXC propose aussi un annuaire d'images, accessible avec `lxc-create -t download`. Créez un conteneur Centos 7.

LXD

- LXD : solution de gestion de conteneurs construite au-dessus de LXC (développé par la même équipe)
- Documentez-vous sur LXD
- Essayez LXD :
  - Soit en ligne sur `https://linuxcontainers.org/lxd/try-it/` (attention : fonctionne mal avec Chromium) – le plus facile
  - Soit dans une VM Vagrant Ubuntu – difficulté moyenne
  - Soit dans une VM Vagrant Debian, en installant LXD avec `snap` – difficulté un peu plus élevée
    Voir `https://stgraber.org/2017/01/18/lxd-on-debian/`

- Question subsidiaire : reprenez votre environnement LXC, et configurez LXC pour que les conteneurs aient accès au réseau