# INF5190 - Virtualisation et conteneurs

Jean-Philippe Caissy

30 octobre 2019

Il y a eu des changements à l'horaire du plan de cours depuis le début de la session.

- La première partie du cours était consacré à la conception d'applications Web :
  - HTTP
  - Fonctionnement d'une application Web
  - Sécurité d'applications Web
  - Modèle MVC
  - REST
  - Formats de sérialisation
- La deuxième moitié de la session se penche sur la gestion et la maintenance d'une application Web :
  - Virtualisation et conteneurs
  - Déploiements
  - Microservices
  - Résilience et performance d'applications Web
  - Maintenance

# Virtualisation

- Virtualisation : création et utilisation d'un environnement virtuel, incluant les périphériques matériels (CPU, carte de son, etc), les engins de stockage (disque dur) et ressources réseaux (carte réseaux).
- Concepts : hôte (*host*) et invité (*guest*)
- Un environnement virtuel est isolé de l'hôte et chacun des invités sont isolés d'eux-mêmes.

# Virtualisation

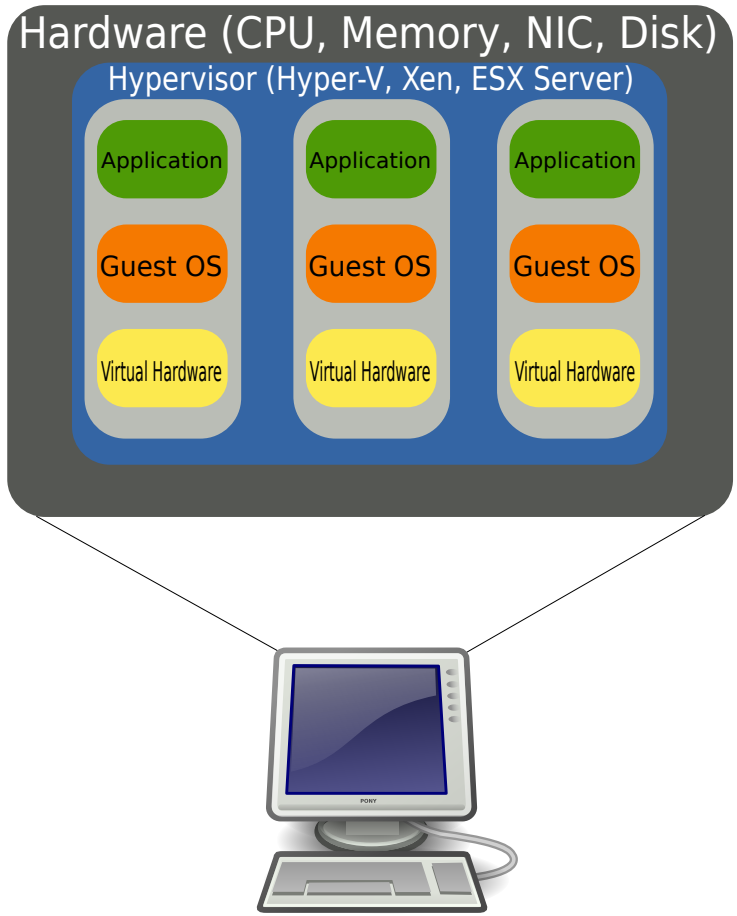



Figure 1: Schéma de virtualisation

Source : John Aplessed [Public domain]

# Virtualisation

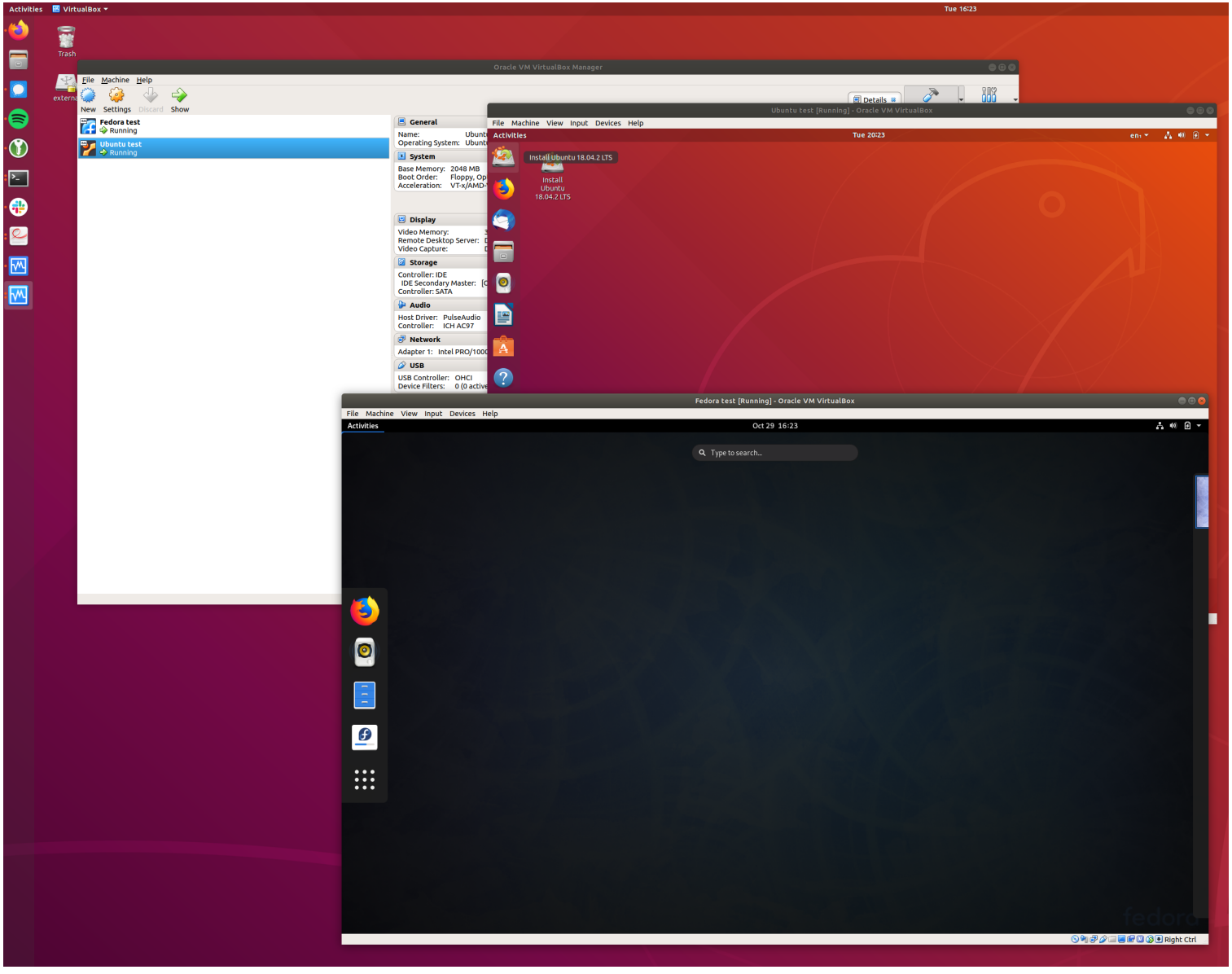



Figure 2: 2 hôtes Linux sous VirtualBox

# Virtualisation

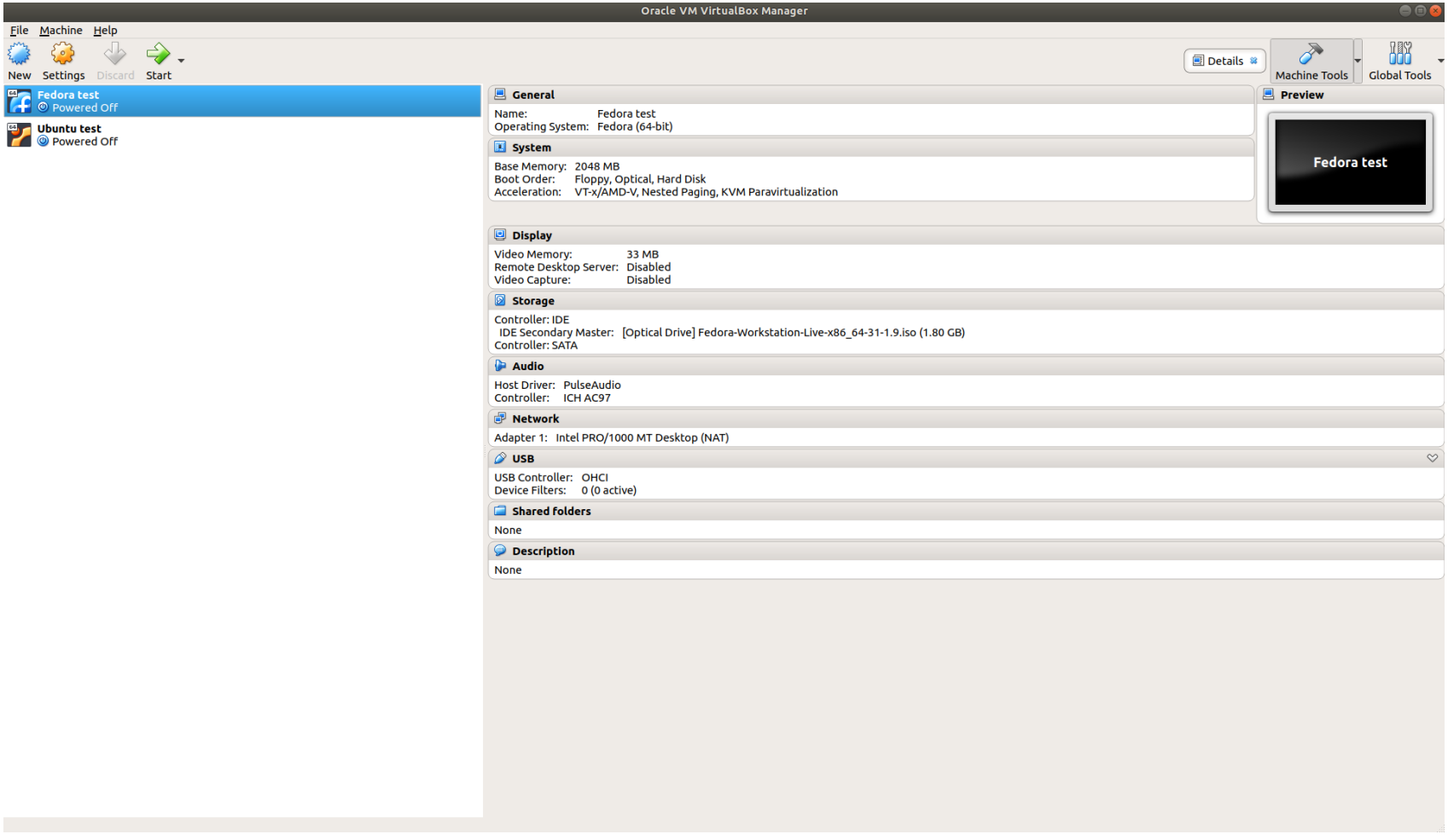



Figure 3: Matériel simulés configurable dans VirtualBox

# Virtualisation

## Avantages

- Environnement reproductible
- Sécurité par la séparation et l'isolation des systèmes
- Meilleur contrôle des coûts
- Mise à l'échelle (*scaling*) plus facile et rapide
- Plus grande flexbilité (IP dynamique, migration, etc)

# Virtualisation

## Désavantages

- En raison de la simulation des environnements, demande plus de ressources
- Ressources physiques (RAM, disque dur, réseau) partagées, donc plus difficile à gérer
- Redondance entre chaque invité (N systèmes d'exploitation complet)

# Virtualisation

## Exemples de scénarios

- Rouler une application qui n'est pas supporté par l'OS de l'hôte
- Tester et évaluer d'autres OS
- Virtualisation de serveurs : plusieurs serveurs virtuels roulent sur le même matériel physique
- Dupliquer des environnements
- Protéger des environnements

# Virtualisation

## Types de virtualisation

### Virtualisation complète

L'invité n'est aucunement conscient qu'il roule sur un environnement virtuel. Aucune modification à l'hôte n'est nécessaire. Le matériel physique est complètement simulé par l'hôte et le système invité croit faire des appels directement sur le matériel physique.
Ex : VirtualBox

### Paravirtualisation

Le système invité qui est virtualisé est au courant qu'il roule dans un environnement virtuel et utilise des pilotes (*drivers*) spécifiques au lieu d'accéder au matériel physique (disque, RAM, etc).
Ex : Xen

# Virtualisation

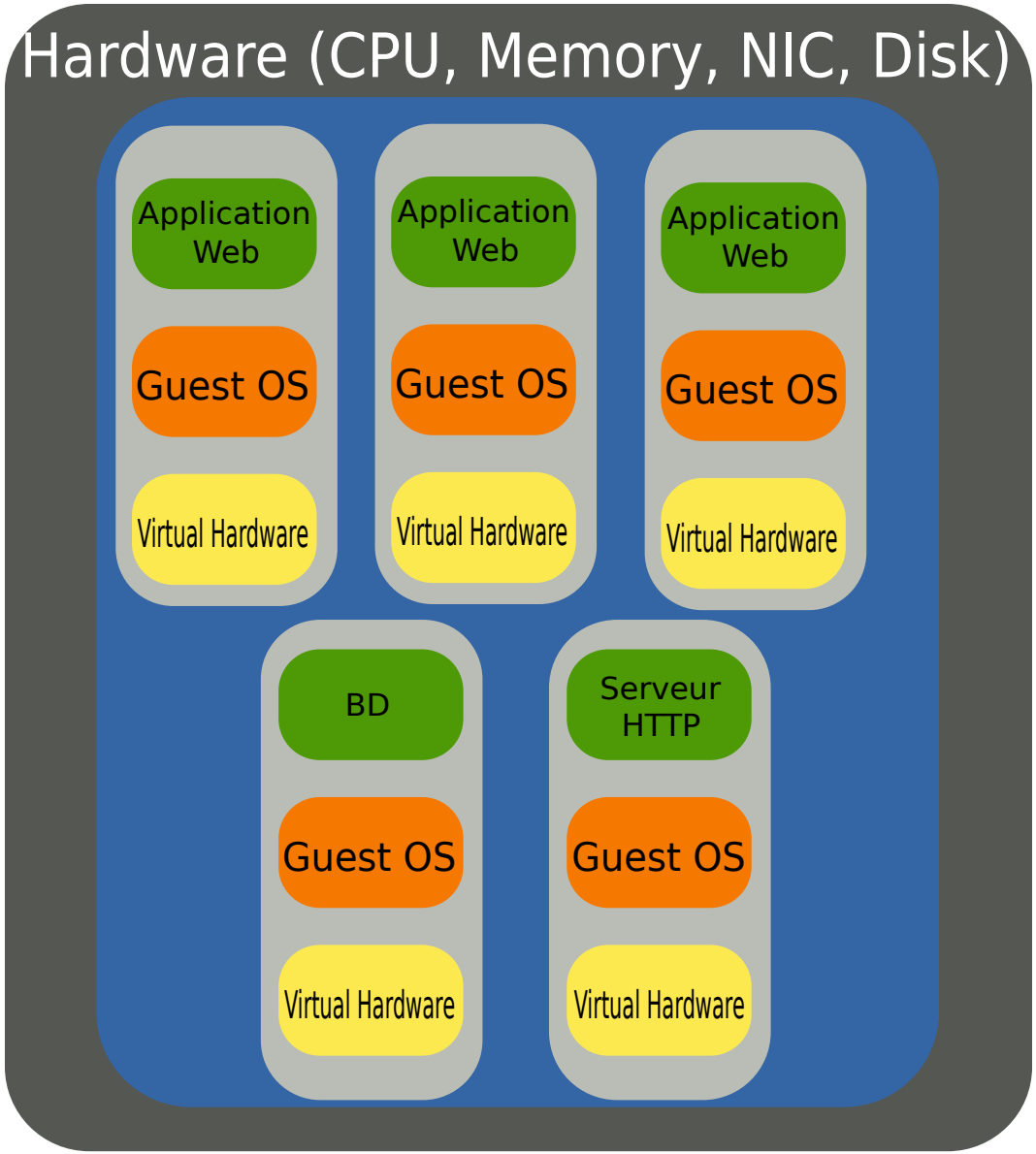
## Application Web

On peut utiliser une application Web pour virtualiser et isoler les composantes :

- Application Web
- Serveur HTTP
- Base de donnée
- etc

# Virtualisation

## Application Web



# Conteneurs

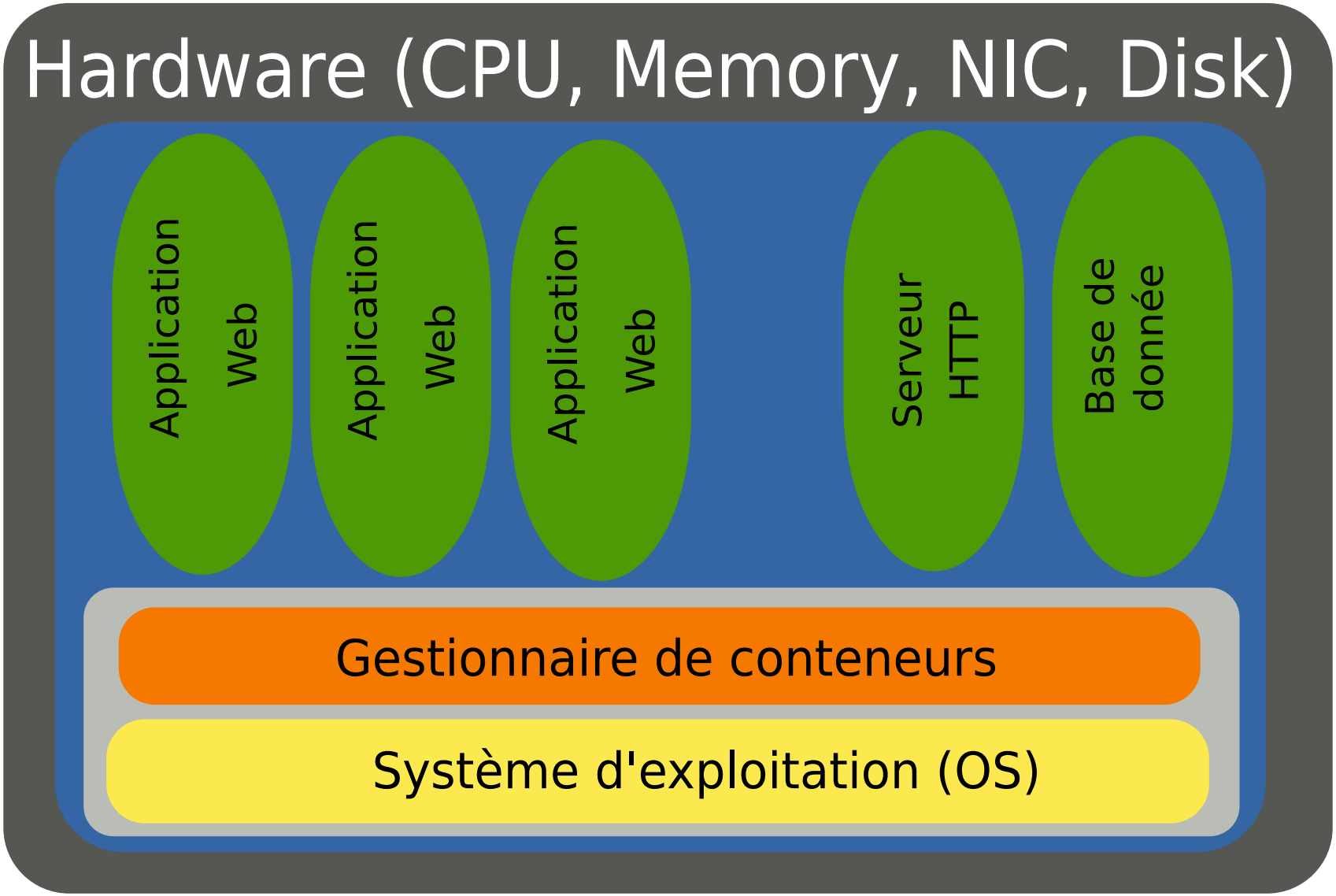



Figure 5: Conteneurs

# Conteneurs

- Paravirtualisation : le matériel n'est pas virtualisé
- Le conteneur partage le système d'exploitation hôte
  - Puisque le OS hôte est partagé : souvent plus léger (RAM, espace disque)
- Mêmes avantages que la virtualisation complète :
  - Isolation, environnement reproductible, etc

# Conteneurs

## Différence entre virtualisation complète

- Virtualisation : rouler des applications qui nécessitent toutes les ressources et fonctionnalités d'un système d'exploitation
- Conteneurs : maximiser les ressources disponible d'un système d'exploitation en roulant le plus d'applications possible

| Virtualisation | Conteneurs |
|---|---|
| Lourd | Léger |
| Perte de performance | Performance native |
| OS complet simulé | Partage de l'OS de l'hôte |
| Périphériques virtualisés | OS virtualisé |
| Démarrage dans les minutes | Démarrage dans les millisecondes |
| Allocation complète de la RAM | Allocation dynamique de la RAM |
| Isolation complète | Isolation au niveau de l'OS |

# Conteneurs

## Types

- **Linux Containers (LXC)** : Système qui permet de virtualiser le noyau de Linux pour permettre de rouler simultanément plusieurs systèmes Linux isolés sous un même hôte.
- **Docker** : Initialement une solution pour faire rouler une seule application avec LXC pour le rendre plus portable et flexible. Maintenant c'est une solution complète de gestion de conteneurs.

# Conteneurs

## Docker

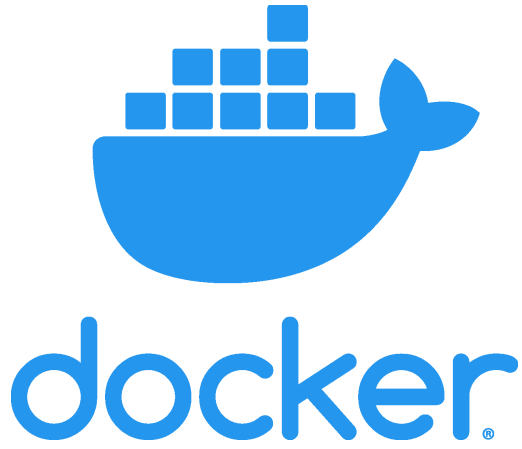



Figure 6: Logo de Docker

- Open source
- Support natif pour Linux et Windows (beta pour Mac)
- Norme de facto pour les applications Web

# Conteneurs

## Docker

## Concepts

- Logiciel (daemon)
  - Installé sur l'hôte
  - Gestionnaire des conteneurs
  - Roule en arrière plan
  - `$ docker -v`
    `Docker version 19.03.4, build 9013bf583`

# Conteneurs

## Docker

## Concepts

- Image
  - Application/binaire unique
  - Statique, tous changements à l'image durant l'exécution sont perdu
- Conteneur
  - **Instance** d'une image
- Registre
  - Répertoire d'images
  - Publique ou privé
  - Exemple : Docker Hub
- Volumes
  - Permet de persister les données d'un conteneur

# Conteneurs

## Docker

## Image

- Disponible à partir de régistre distant (e.g. : Docker Hub) ou localement
- Contient les librairies ou applications de base d'un OS
  - e.g. : Ubuntu, Apache, MySQL

# Conteneurs

## Docker

## Image

- Les images représente une entité qui peut être instancié dans un conteneur
  - Les couches intermédiaires sont en mode lecture seule et représente la différence entre la couche du dessous

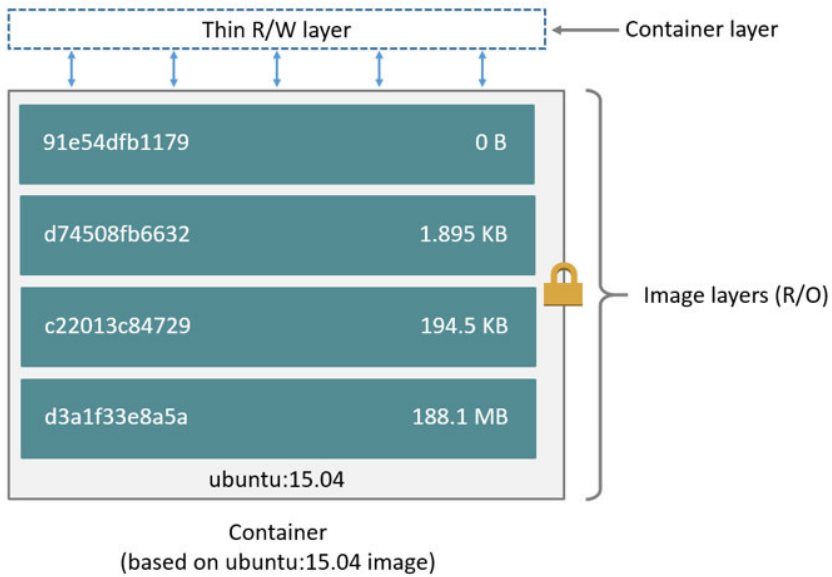



Figure 7: Système de couche des images Docker

# Conteneurs

## Docker

## Registre

- Héberge un ou plusieurs images Docker
- Régistre publique : https://hub.docker.com/
- Régistre privé propre à une entreprise

# Conteneurs

## Docker

## Conteneur

- Un conteneur est lancé à partir d'une image
- L'instantiation de l'image peut être configuré avec des options
- Ne contient qu'une seule application (pid 1 = application)
- Attention : **un conteneur est éphémère**
  - Toutes les données sont perdues lorsque le conteneur est terminé
  - Peut être remplacé par un autre conteneur de la même image
  - Peut être déplacé sur un autre hôte

# Conteneurs

## Docker

## Volumes

- Persistance de données entre les conteneurs
- Un volume Docker est analogue à une clé USB.
- Peut être partagé entre plusieurs conteneurs en même temps
- Exemple :
  - Données d'une application
  - Logs d'une application
  - Fichiers d'une base de donnée

# Conteneurs

## Docker

## Réseau

- Par défaut, les conteneurs utilisent un réseau docker interne
- Communication entre les conteneurs permises grâce au *bridge* réseau
- L'image définit explicitement les ports qui peuvent être exposés
- Docker permet de transférer un port de l'hôte vers un port d'un conteneur
  - Exemple :
    - 8080 -> conteneur nginx 80
    - 54321 -> conteneur postgres 5432

# Conteneurs

## Docker

### Construire une image

Une image est construite avec un `Dockerfile`

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```