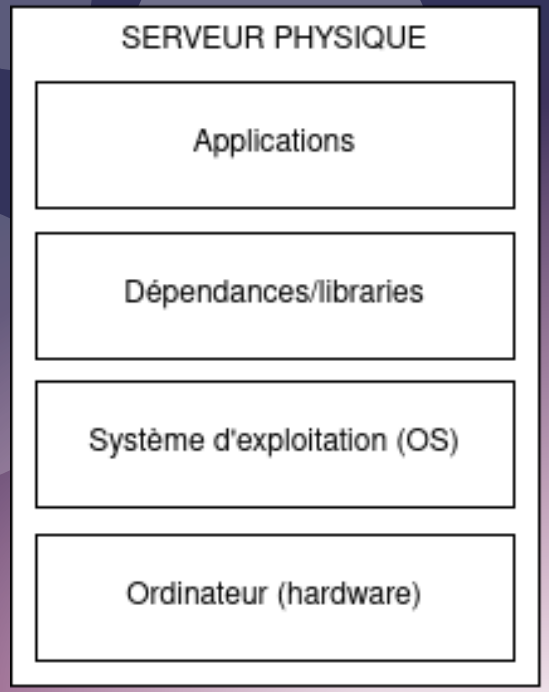
Docker

Introduction à la containerisation

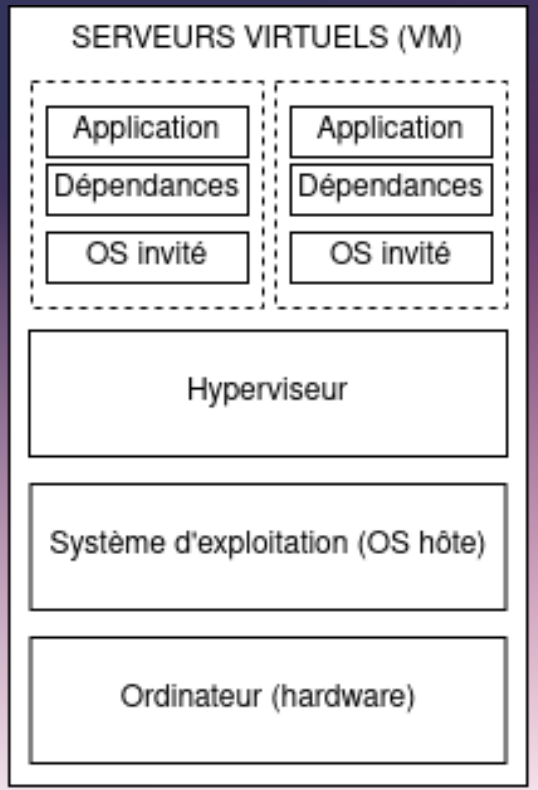
Plan

1. La virtualisation
2. Les containers
3. Docker
4. Le vocabulaire
5. Les volumes
6. Les réseaux

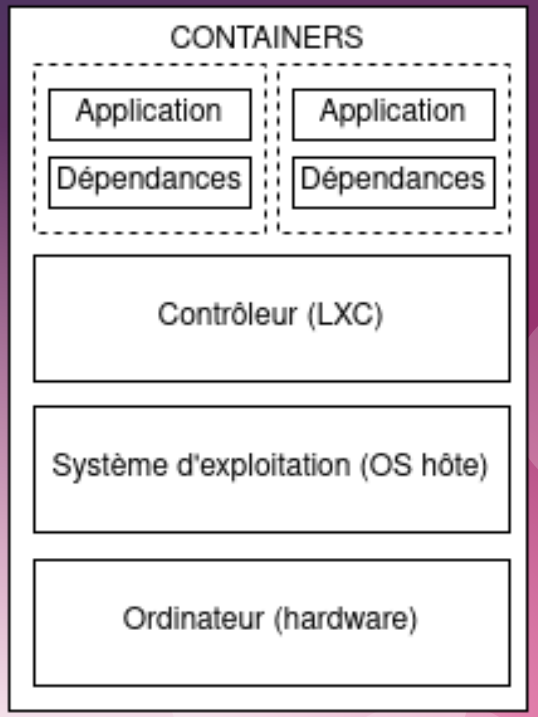
La virtualisation



Architecture simple mais **sous-exploitation des capacités** du serveur car non utilisées



Capacités du serveur utilisées mais non optimisées car espace mémoire, CPU et espace disque surchargés par les nombreux OS hôte

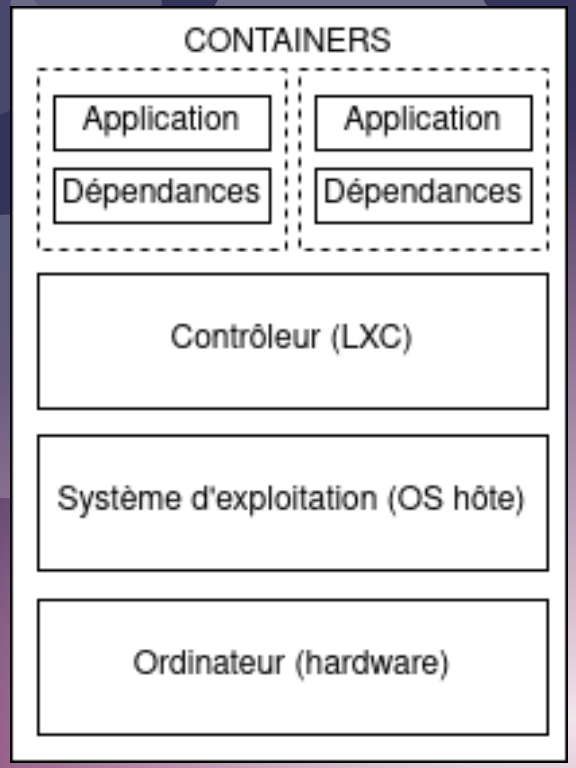


Capacités du serveur utilisées et optimisées car un seul OS

La virtualisation

- **Hyperviseurs Type 2 - hyperviseurs hébergés :** VMWare Fusion, VirtualBox…
- **Hyperviseurs Type 1 - hyperviseurs natifs :** VMWare ESX, Microsoft Hyper-V…
- **Containers :** Docker, Kubernetes (k8s), OpenVZ…

Les containers



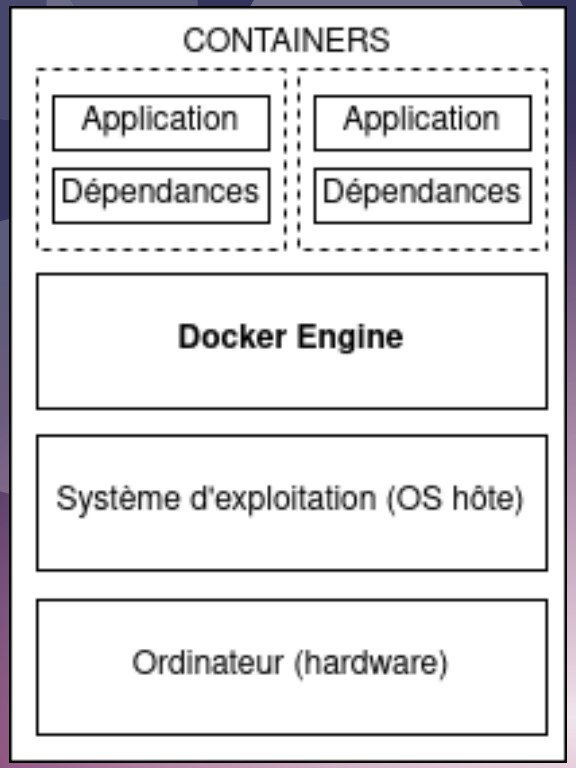
LinuX Containers (LXC) = méthode de cloisonnement au niveau de l'OS basée sur 2 fonctionnalités du noyau Linux :

- **Cgroups** = Control groups pour limiter et isoler les ressources
- **Namespace** = méthode de cloisonnement des espaces de nommages pour rendre inaccessibles (même invisibles !) les ressources d’un groupe à l’autre

Les containers

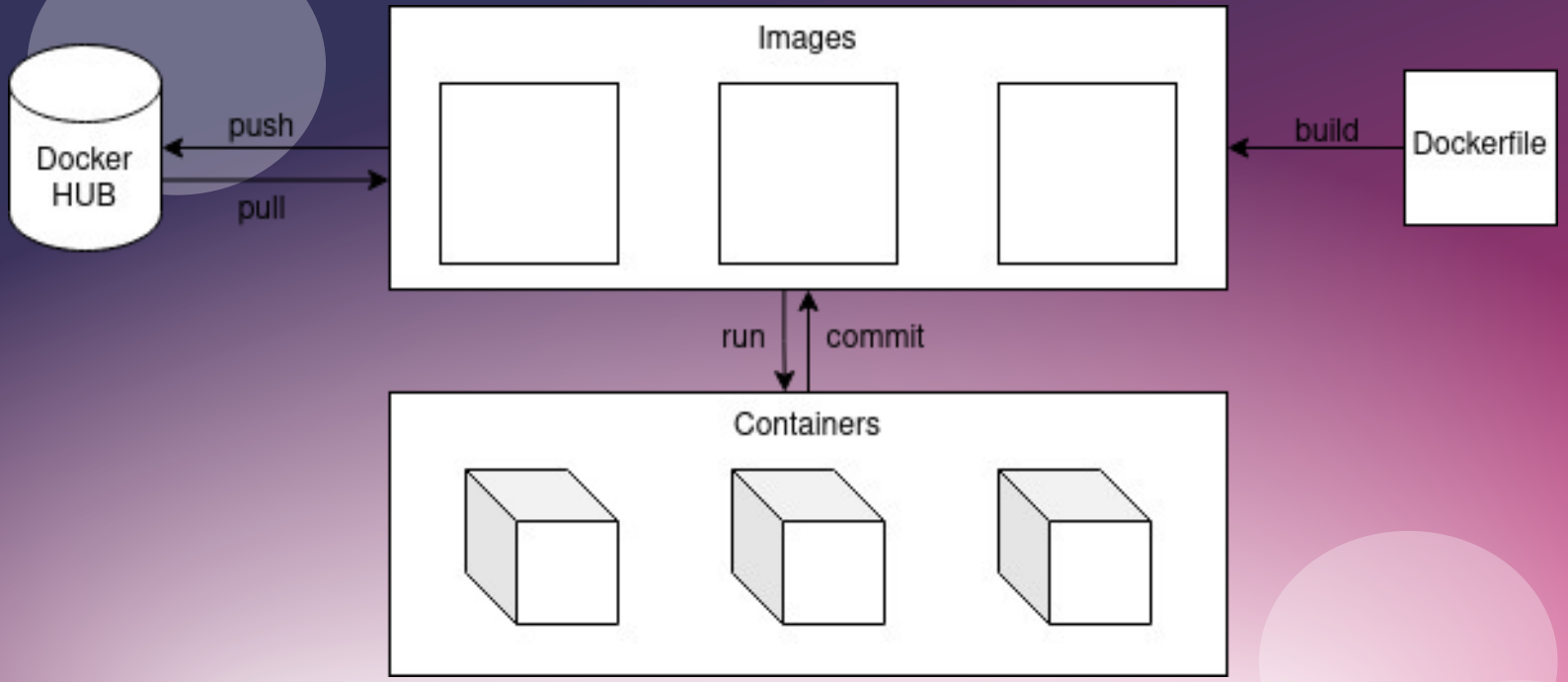
- Les containers permettent de **packager** une application avec **toutes les dépendances nécessaires** et la **configuration**
- Les containers sont **portables** et peuvent être facilement partagés et déplacés

Docker



- Docker est une technologie de virtualisation par conteneurs reposant sur le LXC qui joue le rôle du contrôleur
- Images : templates prêts à l'emploi avec des instructions pour la création de conteneurs
- Container : un empilement d'images (en lecture seules) et une couche accessible en écriture pour la configuration personnelle de votre conteneur

Docker


Images
Docker
HUB
push
pull
build
Dockerfile
run
commit
Containers

Le vocabulaire

HUB

- Registre permettant de télécharger des images docker gratuitement
- Permet d’héberger ses propres images docker publique (gratuit) ou privées (payant)
- https://hub.docker.com

Le vocabulaire

Image

• Contient l’ensemble des fichiers et des paramètres d’exécution par défaut du programme souhaité
• Est « versionnée » avec des tags
• Syntaxe : [registry/][user/]image_name[:tag]
• Tag par défaut : *latest*

Le vocabulaire

Container

• Instance d’une image Docker

• Peut être personnalisé avec différents paramètres (réseau, volumes, …)

• Chaque container est isolé des autres, sauf on les associes explicitement (même réseau, volume partagé)

Les volumes

Par défaut, toutes les données sont stockées dans le container.
Problème : suppression du container => données supprimées

Concept de « volume » pour permettre la persistance : les données sont extériorisées

- Soit sur le système de fichier de l'hôte (comportement par défaut)
- Soit ailleurs, avec des plugins : AWS S3, NFS, …

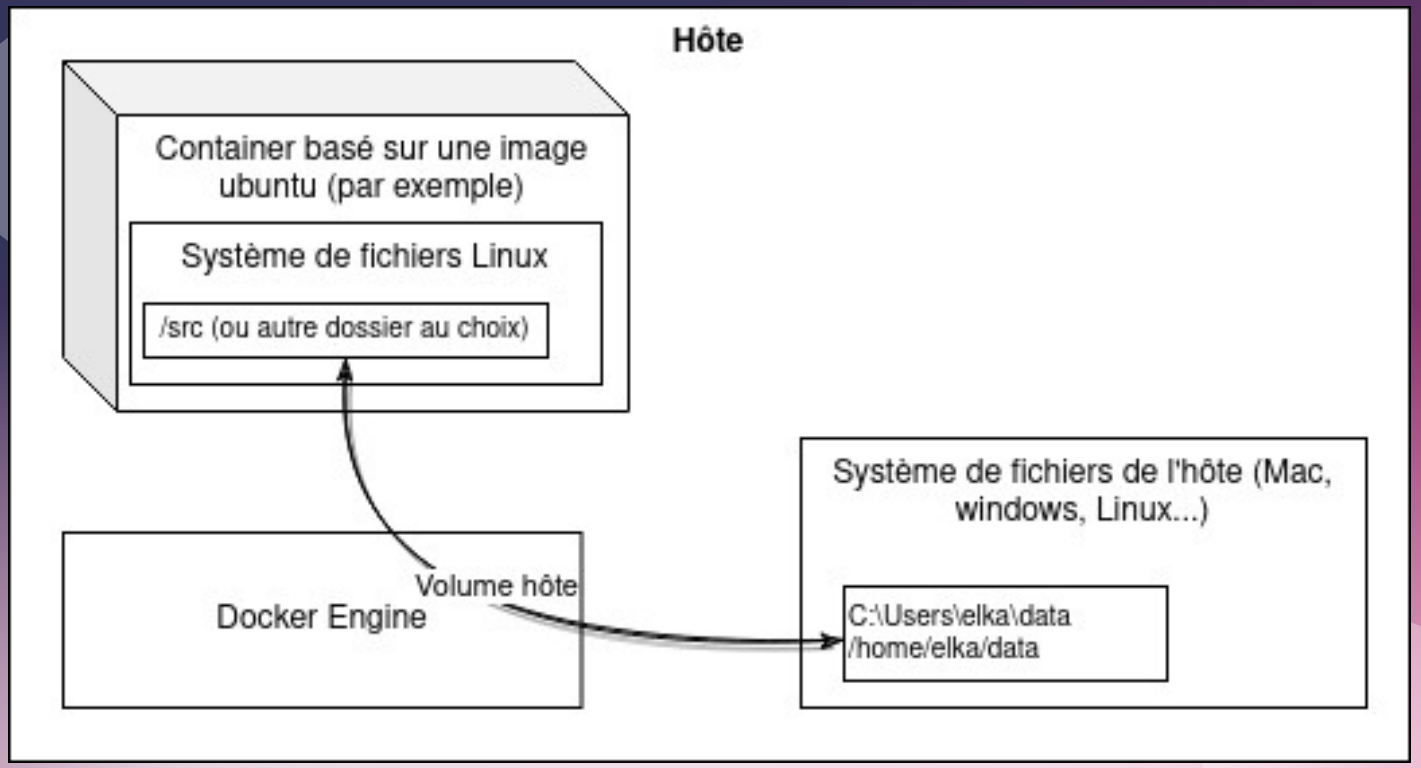
Les volumes

- **Volumes « anonymes » :**
  Créés par défaut si définis dans l'image mais non référencés lors de l'exécution

- **Volumes hôtes :**
  $ docker run -v "/home/username/data:/var/data" […] image_name
  $ docker run -v "C:\Users\Username\data:/var/data" […] image_name

- **Volumes nommés :**
  $ docker volume create volume_name
  $ docker run -v "volume_name:/var/data" […] image_name

Les volumes

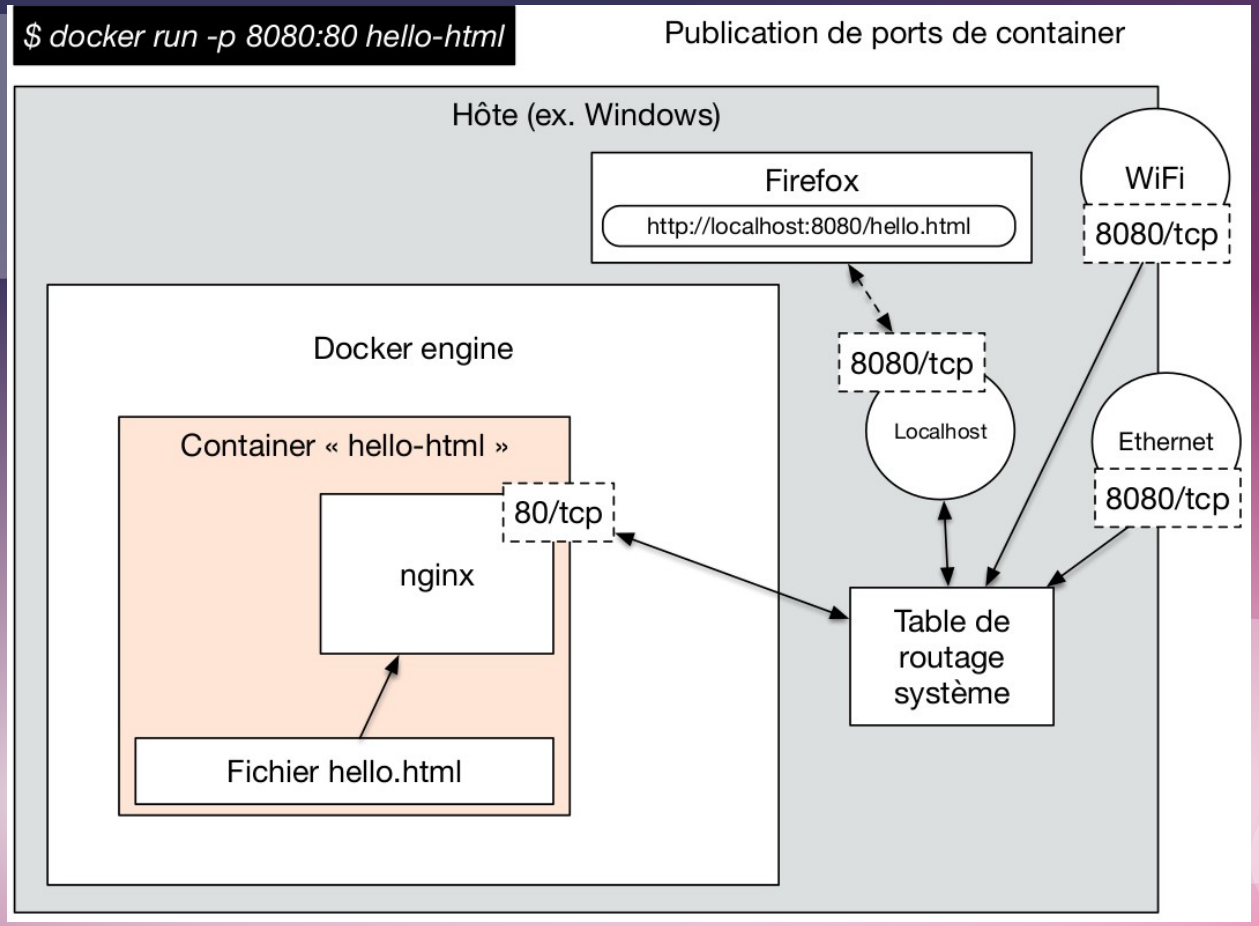


Docker run -v "/home/elka/data:/src" imageName [commande]

Les réseaux

- Docker créé des **réseaux virtuels**
  - Plusieurs containers sur un même réseau peuvent interagir ensemble
  - Condition : avoir le port exporté dans l’image
  - $ docker network create network_name
  - $ docker run --network network_name image_name
- **Publication de ports**
  - Par défaut, les ports sont « fermés » depuis l’extérieur
  - Il faut « publier » le port
  - $ docker run -p "port_externe:port_interne" image_name

Les réseaux


$ docker run -p 8080:80 hello-html
Publication de ports de container
Hôte (ex. Windows)
Firefox
http://localhost:8080/hello.html
WiFi
8080/tcp
Docker engine
8080/tcp
Localhost
Ethernet
8080/tcp
Container « hello-html »
80/tcp
nginx
Table de routage système
Fichier hello.html