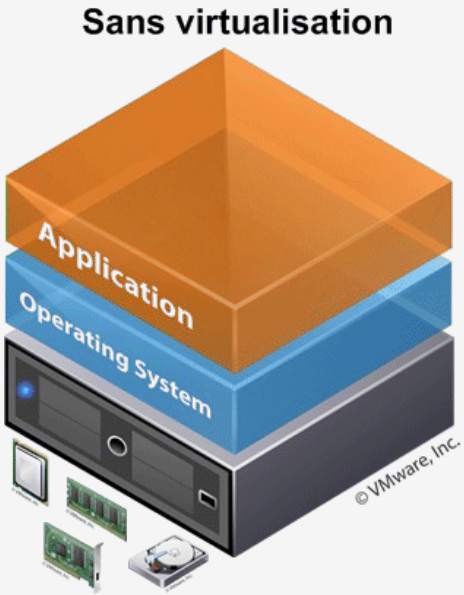
Environnement de travail

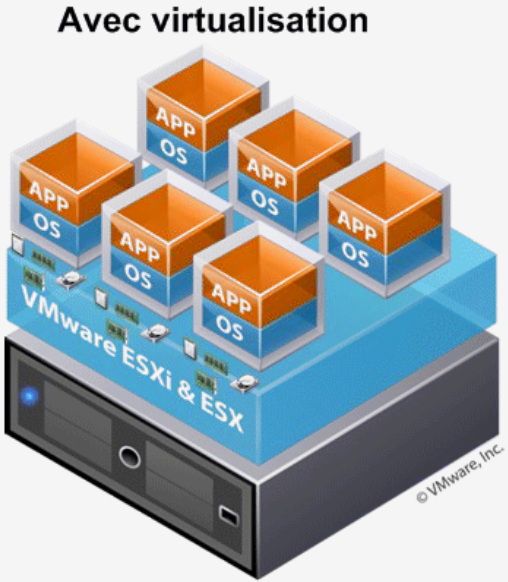
Serveurs, Architecture et Sécurité

Virtualisation

La virtualisation

- Plusieurs OS sur un seul hardware
- Plusieurs types d'OS sur le même hardware (exple: Windows, Linux)
- Optimisation de l'espace et des ressources
- Base du cloud-computing
- Un OS virtualisé => Machine Virtuelle ou VM





Pourquoi virtualiser

- Utilisation optimale des ressources
- Économies (entretien, électricité, espace)
- Facilité de déplacement
- Facilité de test (on peut tout casser et remonter rapidement)
- Avoir plusieurs OS a disposition

Mais attention...

- Si le hardware lâche, plusieurs systèmes sont impactés
- Ajoute de la complexité (installation, connexions)
- Perte de performance

Pourquoi virtualiser

- Tester des nouvelles choses (exple: distributions Linux)
- Tester ses sites sur d'autres systèmes / navigateurs (exple: IE)
- Utiliser des logiciels propres à un OS sur un autre système (exple: Photoshop)

Comment virtualiser

En utilisant un *hyperviseur* (virtualiseur)

- VMware vSphere Hypervisor
- VirtualBox

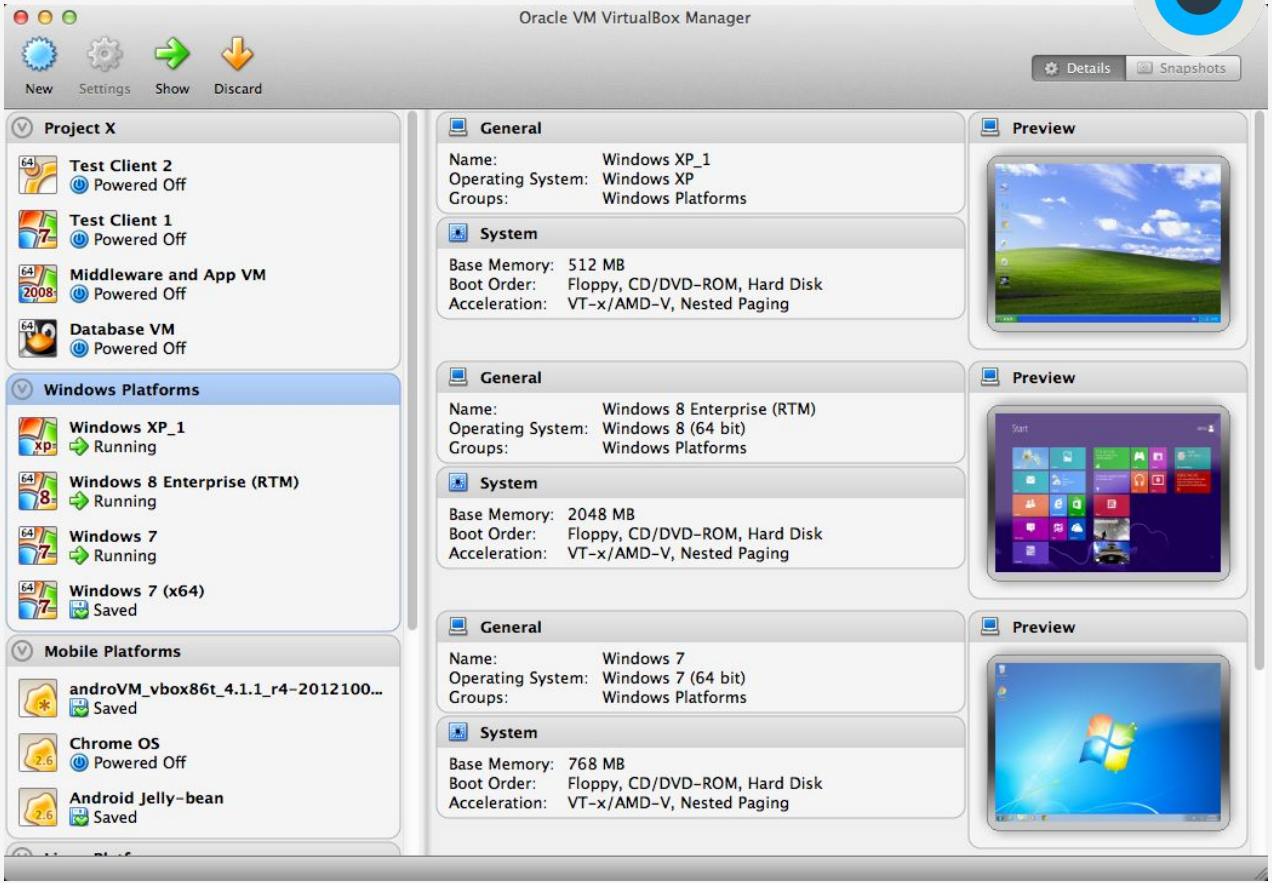
Et une image d'OS (.iso)

- Windows, il faut une licence
- Linux, c'est gratuit
- macOs, ce n'est pas possible

https://www.ubuntu.com/

VirtualBox

- Multi-plateformes
- Simple
- Possibilité de faire des snapshots
- Installer les Guest Additions pour plus d'options



Docker

Docker

- Virtualisation extrêment légère et rapide
- Pas d'interface graphique => ligne de commande
- Introduit le concept de *conteneur*
- Ouvre plein de nouvelles possibilités (DevOps)
- Une application peut être déplacée avec son propre OS
- En grande partie Open-source

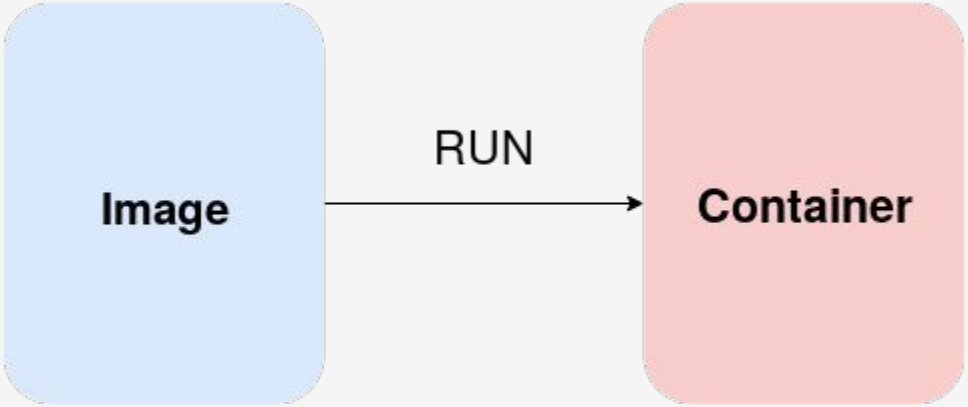
docker

Pourquoi Docker

- Un Linux à disposition en 5s
- Environnement de travail facile à installer et configurer
- Pas besoin de chambouler tout son système hôte
- Développement facilité
- Environnement identique pour tout le monde
- Plus simple à mettre à disposition sur Internet

Comment ça marche

Une image qui contient le minimum de fichiers constituant l'OS



Un container est un OS léger auquel on peut accéder en ligne de commande

- Espace de stockage
- Adresse IP
- Processus
- Partage le CPU et la RAM de l'hôte

Les images disponibles

Le site Docker Hub (Docker Store) rassemble des images de base (Debian, Ubuntu, Windows server...) ainsi que des images créées par la communauté pour faire à peu près tout.

```
docker pull ubuntu
```

```
docker pull ubuntu:16.04
```

https://store.docker.com/

Créer un container

Quand on a une image, on peut créer un container en un instant :

```
docker run -it ubuntu
```

La commande `run` possède beaucoup d'options :

```
docker run -d -v /home/sika:/root -p 80:80 --name testUbuntu ubuntu:16.04
```

https://docs.docker.com/engine/reference/run/

Installation de Docker

=> https://docker.com

- Windows: avec la console PowerShell (et pas cmd)
- macOs: avec le Terminal

Il y a une édition *Entreprise* (payante) et une édition *Community*

Si la commande `docker info` fonctionne, vous avez Docker

Utilisation

- Télécharger une image : `docker pull ghost`
- Lister les images téléchargées : `docker images`
- Lancer un container : `docker run -d -p 8080:2368 ghost`
- Lister les containers en cours : `docker ps`
- Lister tous les containers : `docker ps -a`
- Supprimer un container : `docker rm -f nom_container`
- Supprimer une image : `docker rmi ghost`
- Voir ce qu'affiche un container : `docker logs -f nom_container`

Se connecter à un container

La commande suivante permet de se connecter sur un container déjà en route :

```
docker exec -it nom_container bash
```

- Comme si on était en SSH
- En général *bash* mais selon l'image utilisée, cela peut être juste *sh*
- En général, on est connecté avec l'utilisateur root mais cela est limité au container

Les variables d’environnement

- Sur Unix, chaque chaque session possède un ensemble de variables d’environnement
- Ces variables sont accessibles dans tous les terminaux et depuis les applications
- Docker utilise énormément ces variables pour la configuration
- Pour voir les disponibles : `env`

Connecter une base de données

Un container = Un serveur

Mais comment faire pour avoir Wordpress (serveur Apache + serveur MySQL) dans Docker ?

- On crée un container MySQL
- On crée un container Apache et on le **lie** avec le container MySQL

Créer un Wordpress

Lancement et configuration de la base de données :

```
docker run -d --name srv-mysql -e MYSQL ROOT PASSWORD= rootpass -e
MYSQL_USER=dbuser -e MYSQL_PASSWORD=userpass -e MYSQL_DATABASE=wordpress
mysql
```

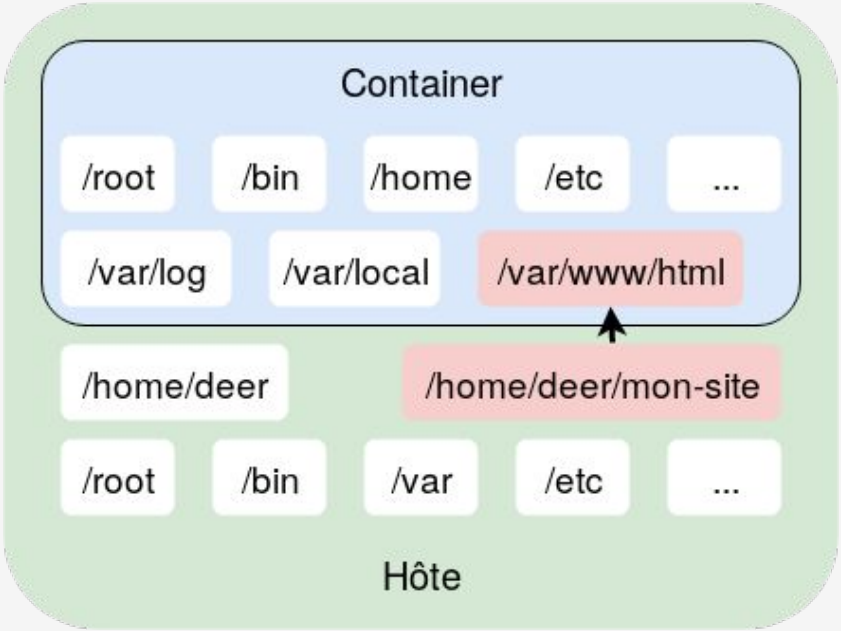
Lancement d'un serveur Apache avec Wordpress déjà prêt :

```
docker run -d --name srv-wp  --link srv-mysql:db -p 8080:80 wordpress
```

Puis installation directement dans la navigateur sur http://localhost:8080

Mettre des fichiers dans le container

- Le container possède ses propres fichiers
- Par défaut, il n'a pas accès aux fichiers de l'hôte
- L'option *-v* permet de *mapper* un dossier de l'hôte dans le container
- Modifier l'un, modifie l'autre
- Écrase si le dossier existe déjà dans le container




```
docker run -v /home/deer/mon-site:/var/www/html wordpress
```

Créer une image Docker

- Une image doit être la plus légère possible et ne contenir que le strict minimum
- Une image est toujours basée sur une autre image par dessus laquelle elle rajoute des *couches*.

Exemple: L'image *Wordpress* se base sur l'image *Apache* et ajoute les fichiers d'installation Wordpress + les dépendances nécessaires

Le Dockerfile

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y apache2

COPY apache.config /etc/apache2/apache2.conf

WORKDIR /var/www/html

CMD /etc/init.d/apache2 start && tail -f /var/log/apache2/*.log
```

Créer une image

À partir d'un fichier Dockerfile et d'éventuels scripts, on peut créer une image Docker :

```
docker build -t mon-image -f Dockerfile .
```

Docker va lire chaque ligne une à une et effectuer les actions. Une fois terminé, la nouvelle image peut être utilisée pour lancer des containers.

Les tags

- Une image peut avoir plusieurs *tags*
- Par défaut, elle prend le tag *latest*
- Cela permet d'avoir plusieurs version d'une image => plusieurs versions d'un programme
- exple: php:latest (php 7.2), php:7.1, php5.6 ...
- Il est possible de voir les différents tags d'une image sur Docker store
- Pour utiliser un autre que tag que *latest*

```
docker tag nom-image nom-image:alpine
```

```
docker build -t mon-image:alpine .
```

Exercice de cours - Pratique 3

1. Lancer un container avec Apache pour publier un fichier *index.html* que vous créez sur votre ordinateur (peu importe son contenu)
2. Utiliser Docker pour installer un Wordpress
3. Qu'est-ce que l'image Docker *alpine* et en quoi est-elle utile ? Notez bien où vous avez trouvé des informations
4. Comment faire pour avoir une image Docker permettant de lancer en une commande simple votre site web ?

Votre rapport est à rendre avant le 26 novembre minuit sur
https://www.dropbox.com/request/qxMqxOh2NJCGUs2U5tI2

Exercice de cours

Si vous voulez aller plus loin:

- Documentez vous sur *docker-compose*
- Faites un fichier docker-compose.yml permettant de lancer un Wordpress avec un serveur MySQL associé en une commande

À faire uniquement **si vous voulez** découvrir Docker plus en profondeur. Ceci ne sera pas évalué.