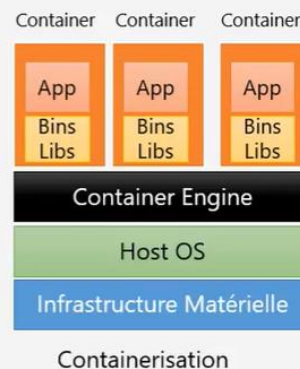


Conteneurisation

2- Conteneurisation :

Container

- Enveloppe permettant de packager une application avec juste ce dont elle a besoin pour fonctionner
- Peut être déployé tel quel dans n'importe quelle machine disposant d'un Container Engine avec différents environnements (Dev, Test, Prod)
- Utilise le Kernel de l'OS Hôte
- A son propre espace de processus et sa propre interface réseau.
- Isolé de l'hôte, mais exécutée directement dessus.
- Permet de décomposer l'infrastructure applicative en petits éléments légers facile à déployer et à réutiliser



Un seul OS partagé pour un ensemble des conteneurs.

On peut dire que c'est presque une virtualisation des applications .

Pourquoi utiliser les Containers

- Meilleures performances que les VM (Démarrage instantané)
- Portabilité d'un environnement à l'autre (Multi cloud)
- Cohérence entre les environnements Dev, test et prod
- Permet de modulariser facilement l'application
- Gérer l'héritage technique (Ancienne application) Grâce à l'isolation

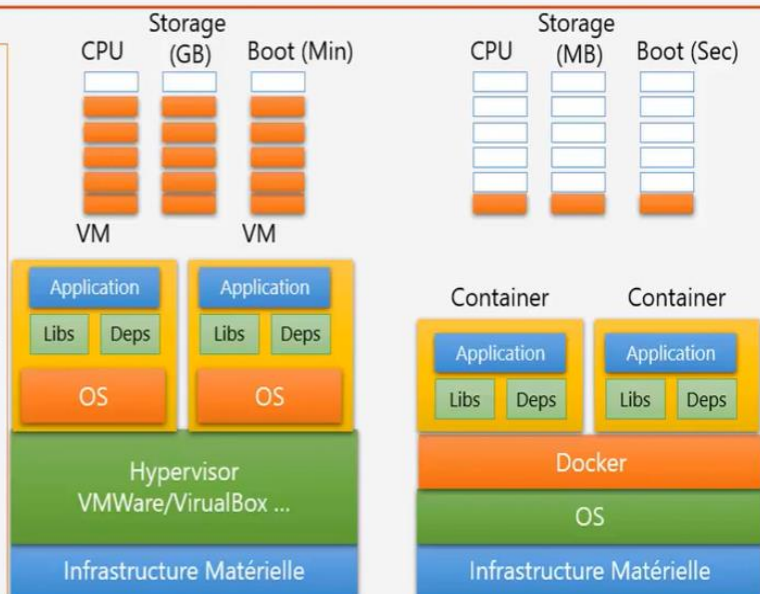
Containers vs Virtual Machines

- Machine Virtuelle :

- Permet de virtualiser une machine physique.
- Chaque VM a son propre OS
- Une VM consomme bcp de ressources (CPU, Stockage) et prend assez de temps pour booter (qq minutes).

- Conteneur :

- Permet de créer un environnement d'exécution des applications
- Les conteneurs utilisent le même OS
- Tous les conteneurs utilisent le même kernel OS (Linux), Consomme peu de ressources, boot rapide (qq secondes)



Un conteneur consomme moins de ressource et il est plus rapide et performant qu'une VM.

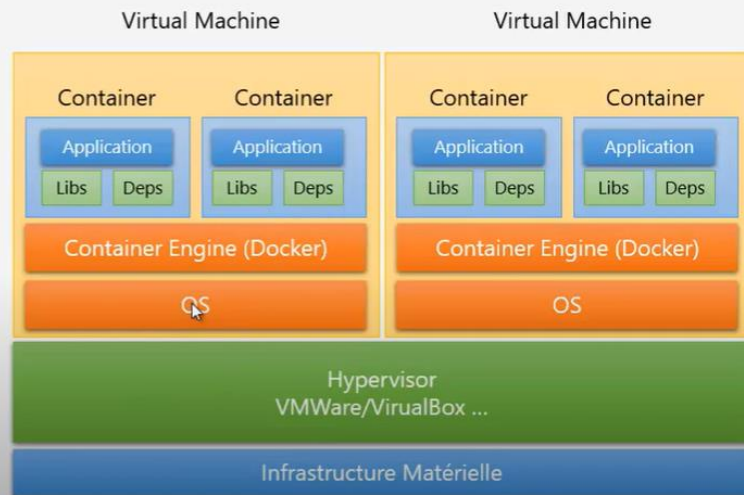
Un conteneur ne nécessite aucune installation pour puisse fonctionner car il admet tout les dépendances , les librairies et tout ce qu'il a besoin pour fonctionner .Facilite les choses pour les opérationnels.

Si les conteneurs n'existaient pas on doit créer un VM pour chaque client et parfois on trouve qu'on crée un VM pour une seule app ce qui est couteux.

Mais le conteneur ne remplace pas la VM pour cela on va utiliser les conteneurs dans des machine virtuelles :

Utiliser des conteneur dans des Machines virtuelles

- Docker ne vient pas pour remplacer les machines virtuelles.
- Dans la pratique on utilise les deux :
 - Les machines virtuelles pour virtualiser les machines
 - Utiliser Docker pour isoler les environnements d'exécution des applications dans des machines virtuelles.
- Ceci pour tirer le bénéfice des technologies



Docker :

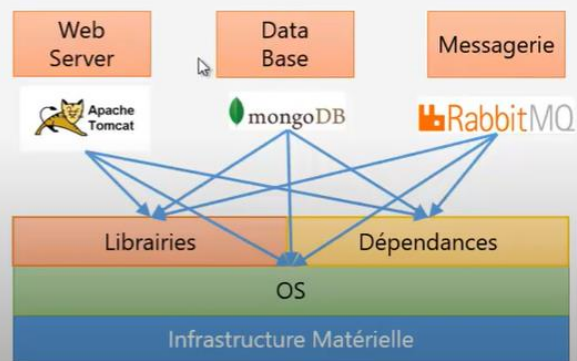
C'est quoi Docker

- Docker permet de créer des environnements (appelées containers) de manière à isoler des applications.
- Il permet d'empaqueter une application ainsi que les dépendances nécessaires dans un conteneur virtuel isolé qui pourra être exécuté sur n'importe quelle machine supportant Docker.
- Docker est un logiciel libre qui permet le déploiement d'applications sous la forme de conteneurs logiciels.

L'importance du Docker :

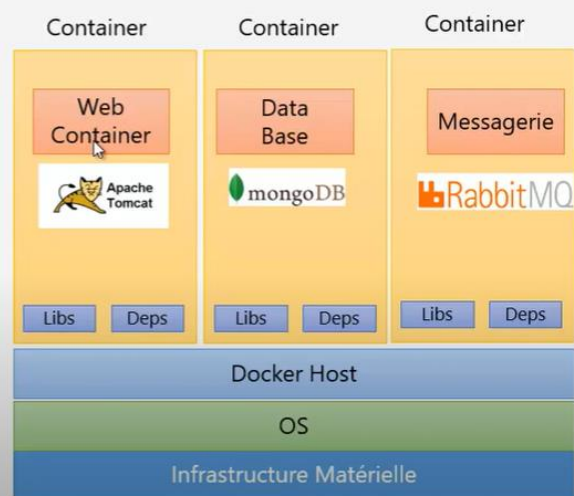
Problèmes de déploiement des applications

- Pour une application utilisant différentes technologies (services), des problèmes sont posés au moment du déploiement en production:
 - Compatibilité des applications avec les OS
 - Installer les dépendances et les librairies requises avec les bonnes versions pour chaque service.
 - Installer les différents environnements :
 - Dev
 - Test
 - Prod
- Ce qui prend beaucoup de temps pour déployer les applications.
- Avec des conflits entre les développeurs et les opérationnels (Administrateurs systèmes)

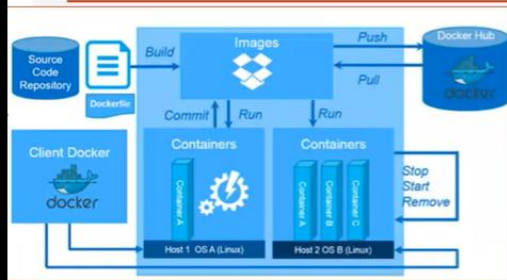


Solution : Conteneurs d'applications

- Embarquer les applications dans des conteneurs
- Exécuter chaque service avec ses propres dépendances dans des conteneurs séparés.



Architecture globale de Docker



- Le développeur crée un fichier Dockerfile contenant les commandes que docker va exécuter pour construire une image docker de cette application.
 - \$ **docker build**
- L'image docker contient tout ce dont l'application a besoin pour s'exécuter correctement.
- Les images Docker peuvent être publiées dans un registre publique (Docker hub) ou privé.
 - \$ **docker push image_name**
- Pour télécharger une image docker d'une application dans un Host Docker, il suffit d'utiliser:
 - \$ **docker pull image_name**
- La création et l'exécution d'un conteneur d'une application se fait par instantiation et exécution de l'image en utilisant :
 - \$ **docker run image_name**
- Avec **docker run**, si l'image n'existe pas dont le host, elle va procéder au téléchargement celle-ci avant d'en créer et exécuter un conteneur docker.
- Docker se compose de :
 - Docker Engine** qui permet de créer le Host Docker sur une machine Linux (Docker daemon)
 - Un client Docker** qui peut se trouver dans n'importe quelle autre machine et qui est connecté à Docker Engine via différents connecteurs exposés par **dockerd** (socket, REST API, etc..)

Image vs Container


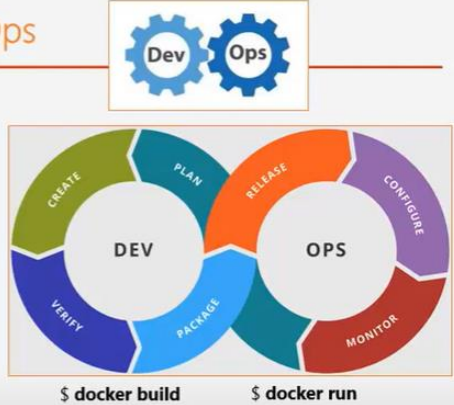
- Une image docker est juste un fichier package représentant la template des conteneurs. Elle définit la structure du conteneur en englobe l'application containerisée et l'ensemble de ses dépendances.
- Un Conteneur représente une instance d'une image. Un conteneur est exécutée par le Docker Host. Ce qui implique l'exécution de l'application qu'il transporte dans un environnement isolé fourni par le conteneur.

Docker host est la machine dont il y'a docker engine.

Docker engine doit être installé sur une machine Linux. (on peut installer l'app desktop du Docker pour ne pas travailler avec une machine virtuelle. nb: l'app desktop crée sa machine virtuelle pour pouvoir travailler avec Docker)

DevOps:

Docker contribue à instaurer la culture DevOps



- Sans Docker :**
 - Le développeur
 - développe l'application.
 - Génère le package de l'application à déployer (App.war)
 - Envoie à l'opérationnel (Administrateur système)
 - App.war
 - Un descriptif des dépendances qu'il faut installer et configurer pour que l'application s'exécute normalement.
 - L'opérationnel
 - doit se débrouiller pour satisfaire les exigences de l'application.
 - Pour chaque mise à jour, c'est toujours les mêmes histoires qui se répètent.
 - Ce qui rend la vie dur au administrateur systèmes (Opérationnels)
 - Ce qui crée beaucoup de conflits entre les développeurs qui tentent d'améliorer constamment les applications et les opérationnels qui doivent redéployer les mises à jour.
- Avec Docker:**
 - Le développeur
 - développe l'application
 - Construit une image Docker de son application contenant toutes les dépendances dont l'appli a besoin.
 - Publie l'image Docker de le registre Docker
 - L'opérationnel déploie l'application en instanciant des conteneurs à partir de l'image Docker récupérée à partir du repository Docker.

Docker Compose :

Docker Compose

- Compose est un outil permettant de définir et d'exécuter des applications Docker à conteneurs multiples.
- Avec Compose, vous utilisez un fichier au format YAML (**docker-compose.yml**) pour configurer les services de votre application.
- Ensuite, avec une seule commande, vous créez et démarrez tous les services de votre configuration.
 - \$ docker-compose up**

```
$ docker run --name=web simple-web-app
$ docker run --name=database mongodb
$ docker run --name=messaging redis:alpine
$ docker run --name=orchestration ansible
```

docker-compose.yml

```
services:
  web:
    image: "simple-web-app"
  database:
    image: "mongodb"
  messaging:
    image: "redis"
  orchestration:
    image: "ansible"
```

```
$ docker-compose up
```

