Docker For Developpers

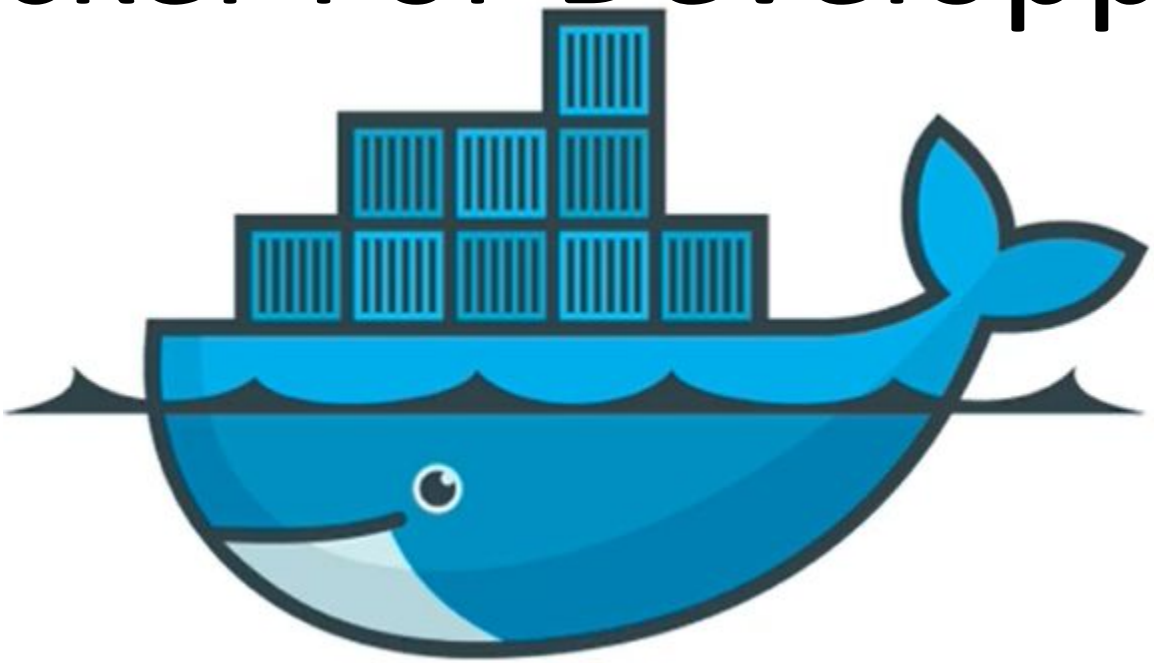


Table des matières

Docker For Developpers

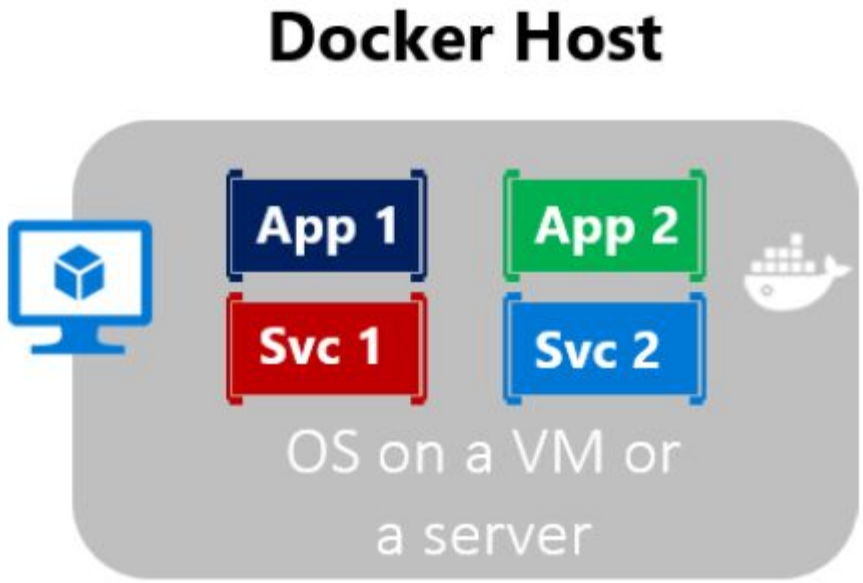
Introduction

Docker For Developpers

Introduction

Containerization

- Approche pour le développement de logiciel ou de services
- Les dépendances et la configuration sont « empaquetées » ensemble dans une Image
- Les « containers » agissent comme un standard de déploiement d'unité logicielle pouvant contenir différents codes et dépendances
- Les logiciels « containerisés » permettent aux développeur de déployer sur différents environnements avec peu voir aucunes modifications.
- Un « container » isole les applications contenues de l'OS et des autres applications
- L'application « containerisée » s'exécute dans un Host s'exécutant par-dessus l'OS (Windows/Linux) avec un coût en terme de ressources moindre qu'une machine virtuelle.
- Facilite la « scalabilité » par l'ajout de nouveaux containers



Introduction

Les avantages

1. Isolation
2. Portabilité
3. Agilité
4. Scalabilité
5. Contrôle du cycle de vie d'une application

En résumé :

Isolation parfaite entre l'environnement de développement et l'environnement opérationnel

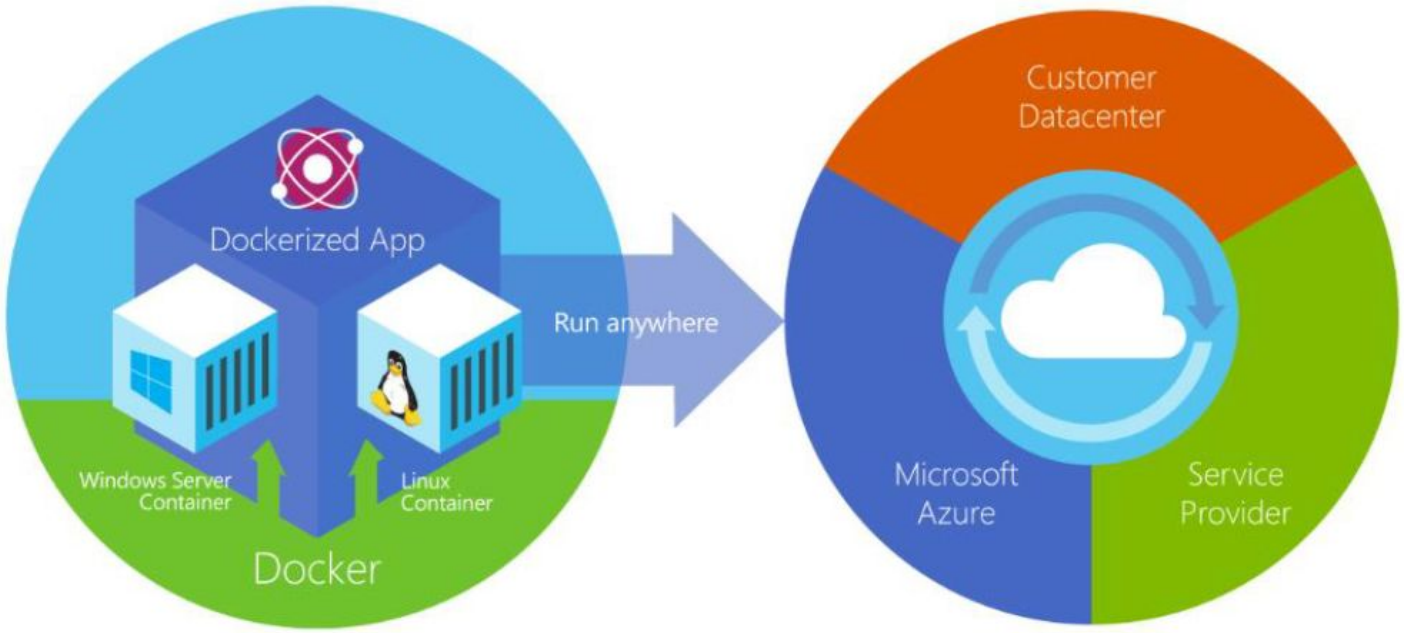
Docker ?

Docker For Developpers

Docker ?

Docker est un projet Open-Source pour automatiser le déploiement d'applications portables dans un container auto-suffisant qui peut s'exécuter aussi bien dans le cloud que dans un environnement local.

Mais Docker c'est aussi une entreprise qui promotionne, maintient, fait évoluer…cette technologie en travaillant en collaboration avec les fournisseurs de Cloud (AWS, Azure,…) et d'OS (Linux, Windows,…)



Docker?

Les « Docker Image Containers » peuvent s'exécuter nativement sur Linux ou Windows.

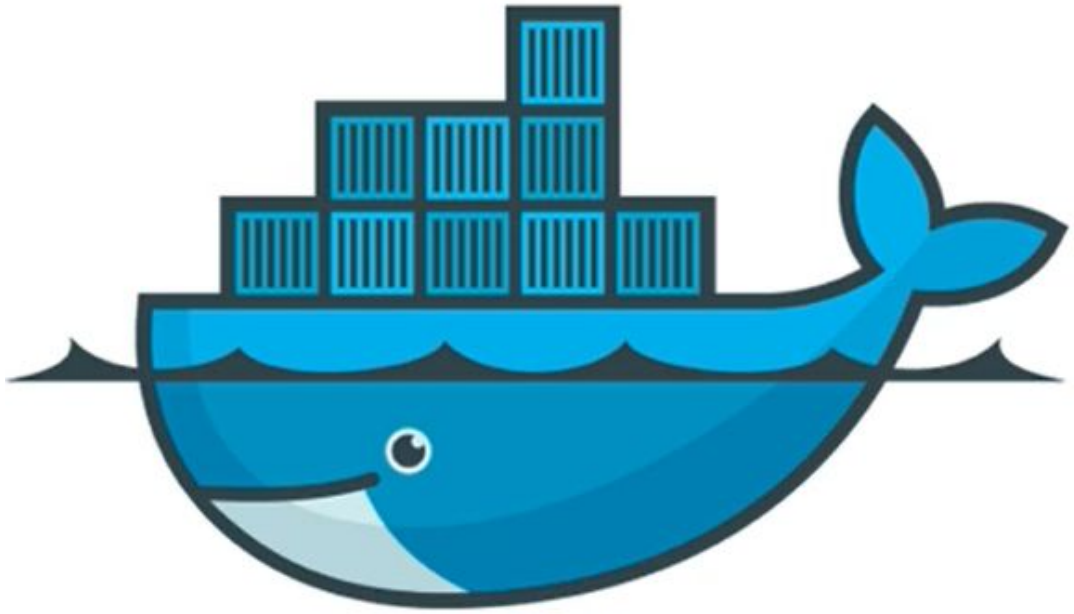
Mais les images Windows ne peuvent fonctionner que sur des hosts Windows et les images Linux sur des hosts Linux.

Il faut donc des VM ou un server pour héberger les hosts.

Pour exécuter des *Windows Containers*, nous avons deux possibilités :

- Windows Server Containers
  Fournit une isolation des applications « native » en partageant un Kernel avec le container host et tous les containers s'y exécutant.
- Hyper-V Container
  Fournit une meilleure isolation en exécutant chaque container sur une machine virtuelle optimisée sans partager le Kernet du serveur.

Docker?



En quelques mots :

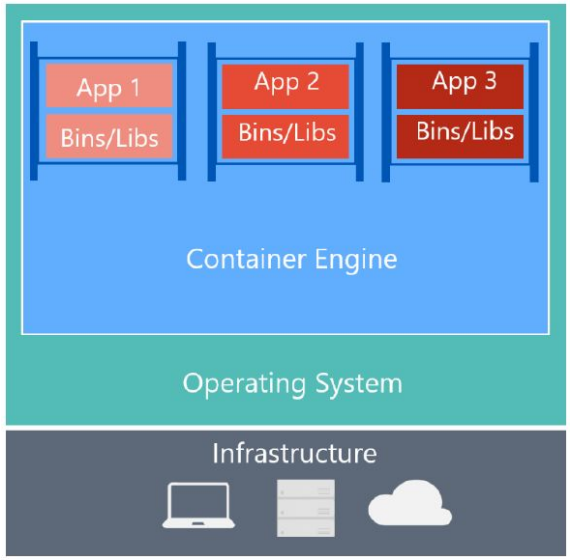
- Plateforme ouverte, sécurisée et légère
- Simplifie la construction, l'exécution et le déploiement des applications
- S'exécute nativement sur Linux ou Windows Server
- S'exécute sur les machines pour le développement sous Windows/Mac par l'intermédiaire de machines virtuelles

Docker VS Virtual Machines

Docker For Developpers

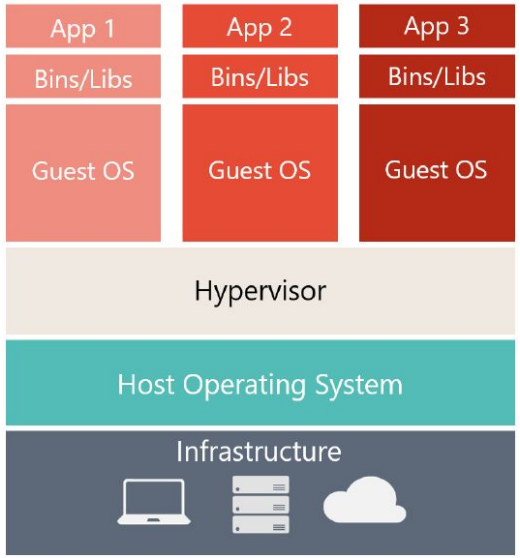
Docker VS Virtual Machines

Docker



Chaque conteneur contient l'application et toutes les dépendances nécessaires tout en partageant le kernel de l'OS avec les autres conteneurs.

Virtual Machines



Chaque machine virtuelle contient l'application, les librairies et dépendances et un OS complet.
Une virtualisation complète requiert plus de ressources que la dockerisation

Docker VS Virtual Machines

Même si la finalité est la même (Isoler les applications), le fonctionnement est quand à lui différent.

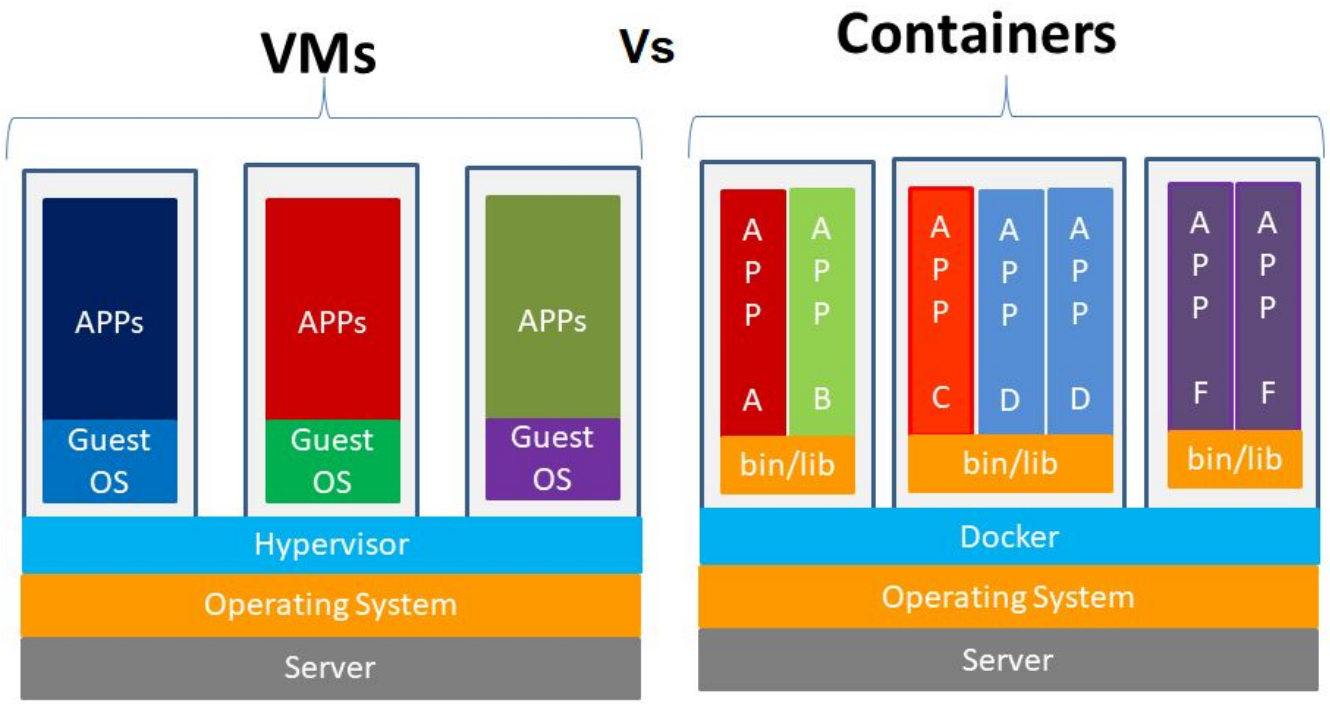
1. <u>OS et Hardware</u>

   Une VM possède son propre OS et utilise une partie des ressources de l'hôte. Un containeur possède le même OS que la machine hôte et le hardware est partagé

2. <u>La taille</u>

   Une VM possédant son propre OS, sa taille peut être très grande.

   A contrario, un container n'utilise que quelques méga pour ses librairies (bin/lib)



Docker VS Virtual Machines

3. Performance de démarrage

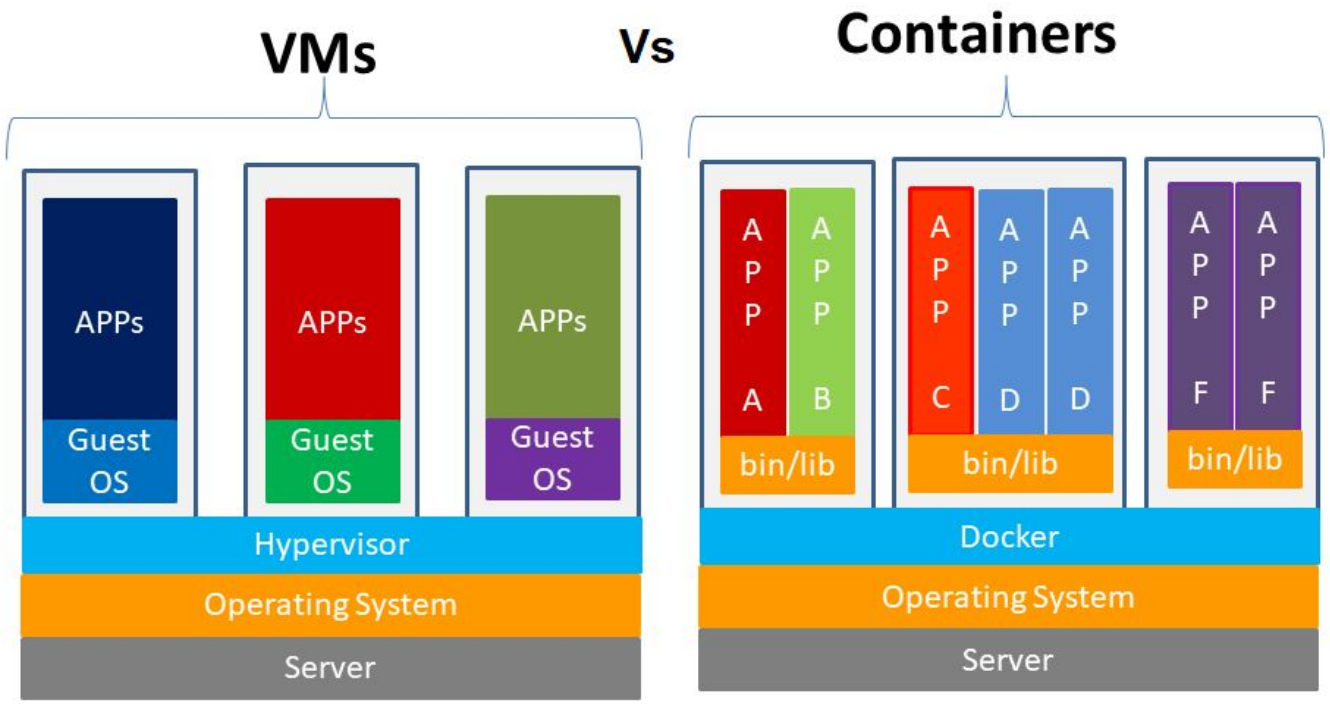
   Une VM doit exécuter et initialiser un nouvel OS tandis que le container est « ready to use »

4. Isolation

   Une VM s'exécute de manière complètement isolée de l'hôte et des autres vm's. Un container partage le même OS et ne fournit pas d'isolation complète

5. Maintenance

   Chaque VM doit être mise à jour, patchée séparément. Un container partage les mise à jours et patch de l'OS hôte.



Images et Containers

Docker For Developpers

Images et Containers

Docker Image :

Template en lecture-seule permettant de construire un container. Contient les librairies et les exécutables nécessaire pour exécuter une application.

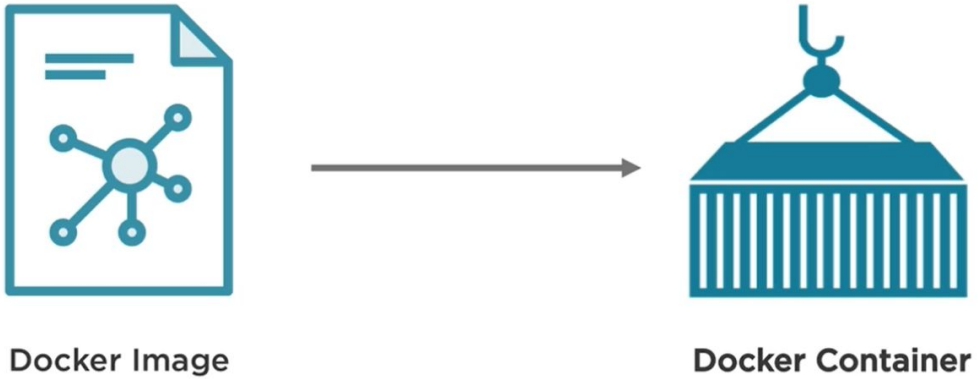
Exemple : WindowsCore avec IIS + .net Core + application

Une image seule n'a pas de sens car finalement elle ne contient que des définitions d'environnement et de la configuration.

Docker Container

Un container est un environnement isolé, sécurisé créé grâce à une image. C'est le container qui est l'unité d'exécution des environnements (OS, Framework,…) et du code des applications.

Il peut être démarré, stoppé, supprimé, déplacé,…facilement et rapidement.



Bénéfices pour les développeurs

Docker For Developper

Bénéfices

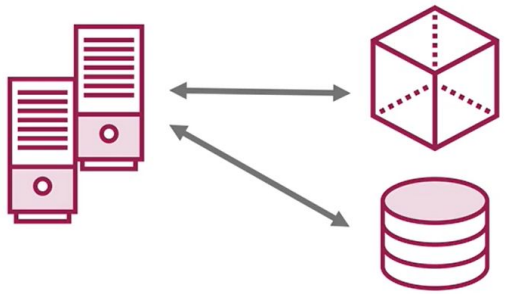
1) <u>Plus vite opérationnel</u>

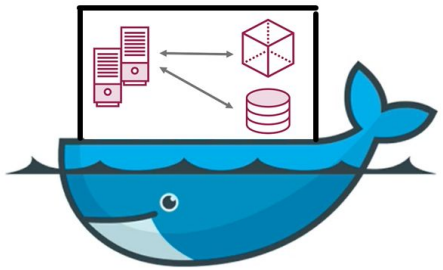
Pour un projet, nous pouvons avoir des designer, web developper, …

Chacun a besoin d'avoir accès à un même environnent de développement mais avec des droits différents par exemple.

Il est donc possible de créer des environnements sous docker pouvant être exécuté sur les machines des intervenants avec les caractéristiques et configurations communes.

Si un nouveau membre rejoint l'équipe, il est très facile de repartir de l'image pour mettre à disposition un docker.





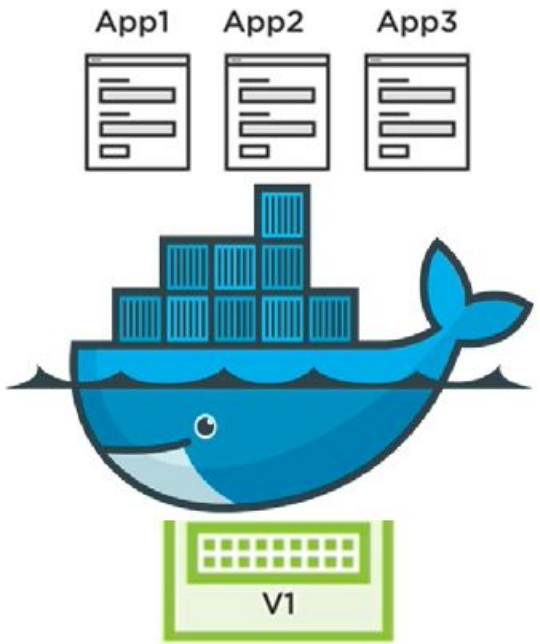
Éliminer les conflits

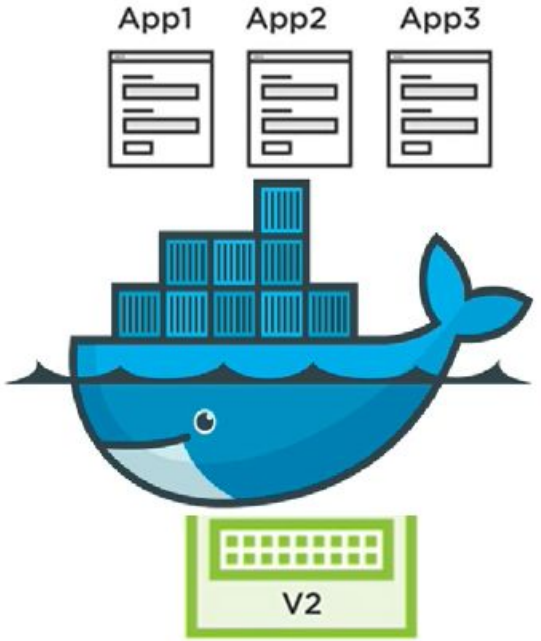
Que cela soit des conflits de version ou de Framework, ils peuvent être résolus plus facilement avec des environnements docker.

Imaginons qu'il faille passer à la version suivante d'un Framework. Quel serait l'impact sur l'application actuelle?

Peut-on prendre le risque de migrer l'application actuelle ?

Comme docker isole chaque image, nous avons donc plus de facilité à migrer un container vers la nouvelle version, tout en gardant un version fonctionnelle (l'ancienne)

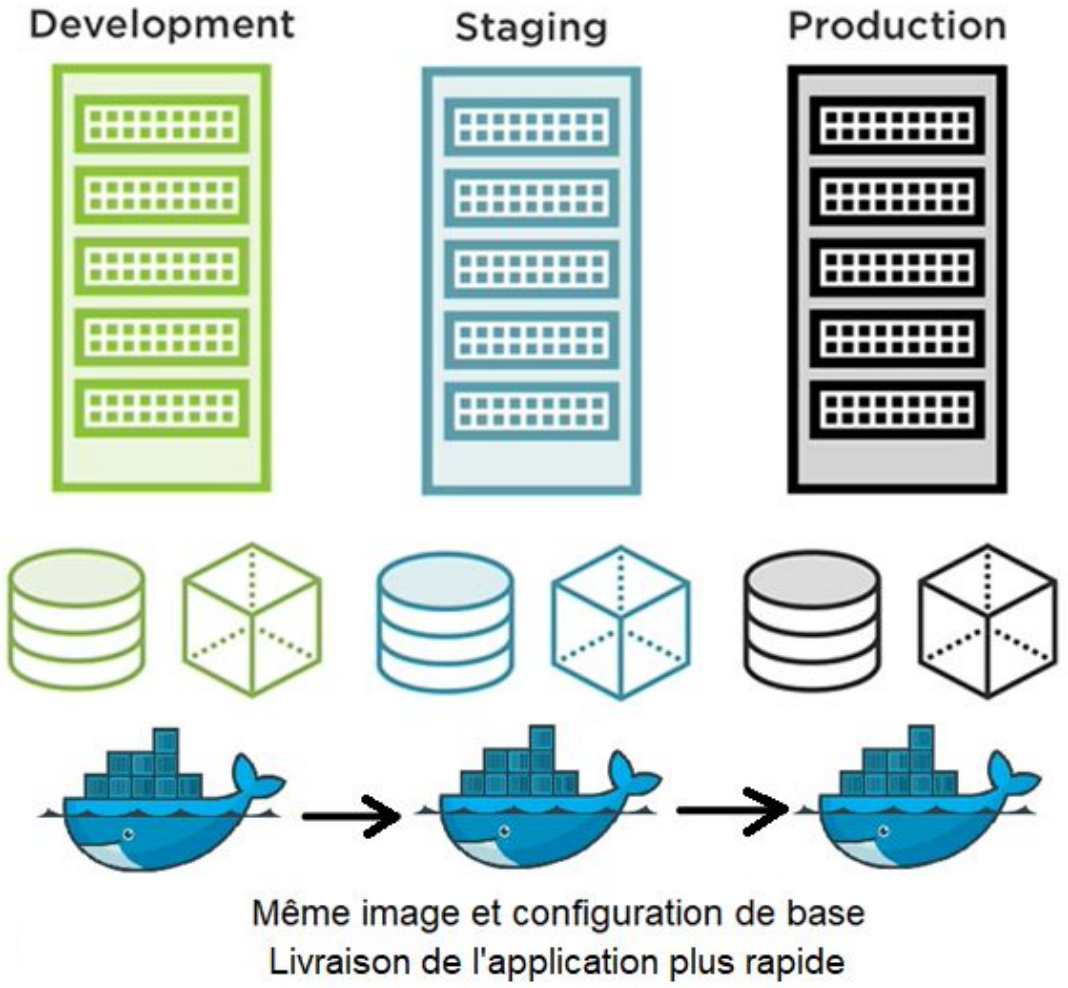




Consistance des environnements

Lors du développement d'application, il est important que les environnements soient identiques afin d'accélérer le déploiement et le testing et d'éviter les conflits entre les versions dev, staging et prod…

La solution docker permet de définir l'environnement basé sur la prod et de créer par la suite des environnements staging et dev en tout point identique.



Le système de fichier en couche

Docker For Developper

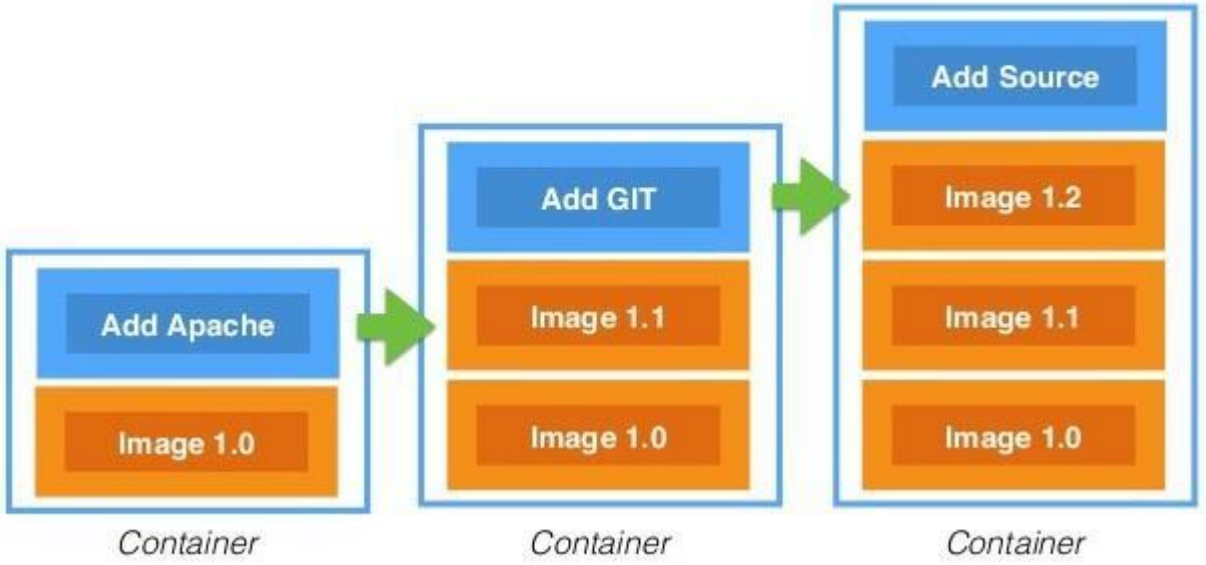
Le système de fichier en couche

Le principe est de créer des images (read-only) pour chacune des étapes de la mise en place du containeur final (read-write)

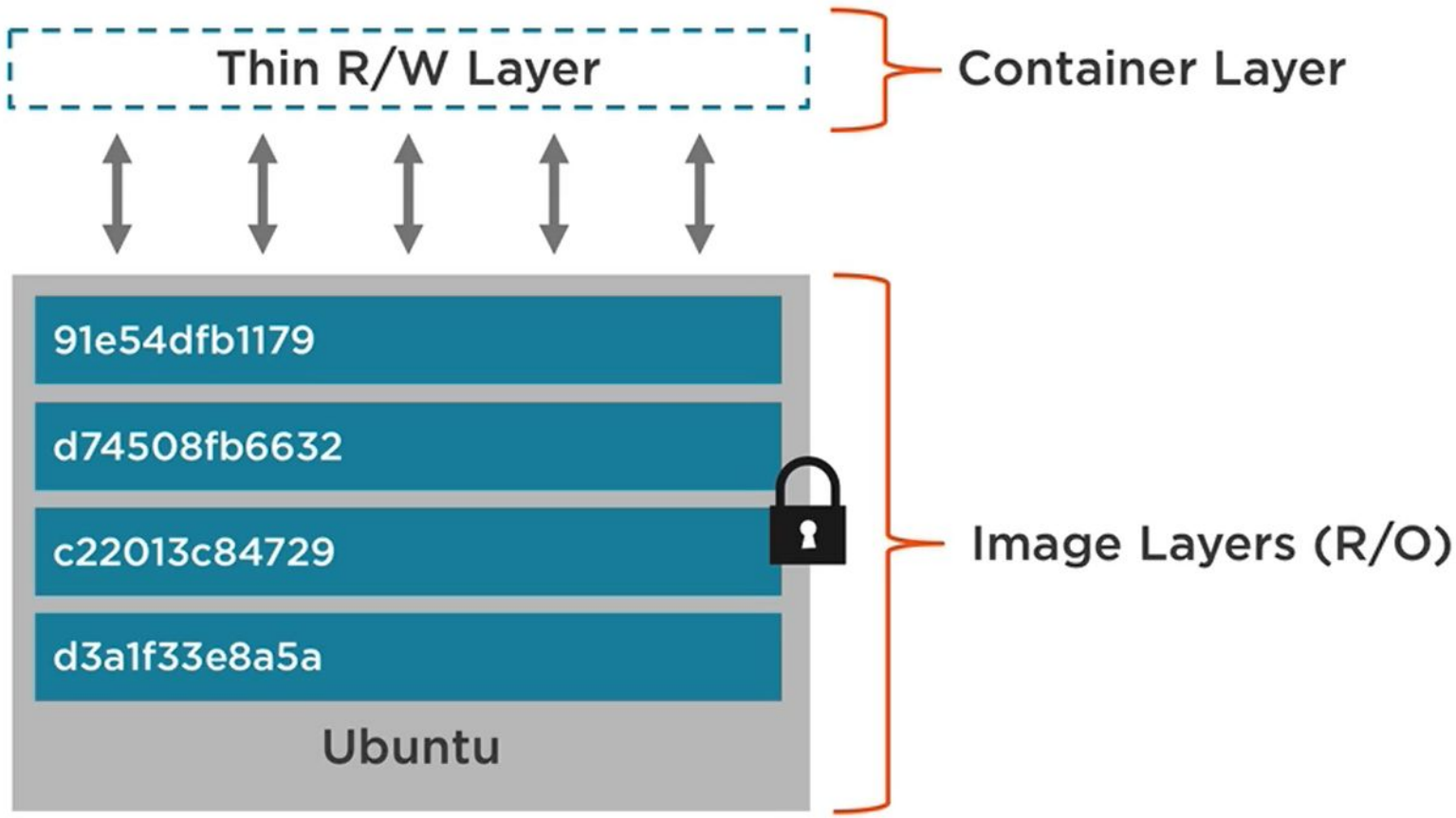
Un peu comme les commit pour un gestionnaire de code source.

Le containeur est donc la couche du haut permettant d'écrire les changements nécessaires (ajout outils, configuration,…)

Grâce à ce système, plusieurs containeur peuvent partager la même image tout en gardant leurs propres données et leur états.



Le système de fichier en couche



Installation

Docker For Developper

Installation

Sous Windows

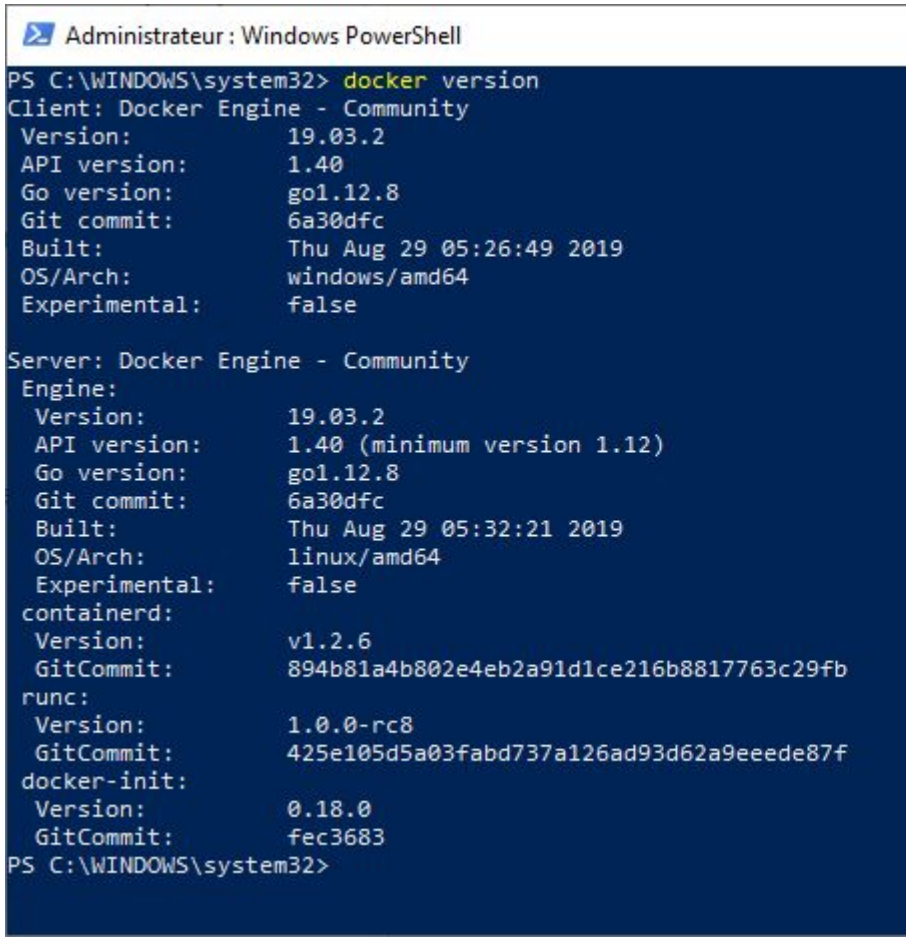
https://docs.docker.com/docker-for-windows/install/

Sous Linux

https://runnable.com/docker/install-docker-on-linux

Lorsque l'installation est effective, vous pouvez rentrer la commande suivante dans votre powershell/command line environnement

docker version



Quelques commandes

Docker For Developper

Quelques commandes

Containers		
docker container	create	Créer un containeur à partir d'une image
	start	Lancer un containeur existant
	run	Créer un nouveau containeur et le lancer
	ls	Lister les containeurs en exécution
	inspect	Obtenir des informations sur un containeur
	logs	Afficher les logs
	stop	Stopper proprement un containeur
	kill	Stopper le containeur de manière brusque
	rm	Supprimer un containeur

Quelques commandes

Images		
docker image	build	Compiler une image
	push	Envoyer une image vers une repository distant
	ls	Lister les images
	history	Récupérer les informations sur une image
	inspect	Obtenir un maximum d'informations sur l'image
	rm	Supprimer une image

Mise en pratique

Après l'installation, ouvrez une fenêtre powershell

et lancez la commande suivante :

```
docker run -dp 80:80 docker/getting-started
```

Ensuite dans votre navigateur :

http://localhost:8080/tutorial/our-application/

et lets go !