Orchestration et conteneurs

Mickael AZEMA

podman

docker

aws

kubernetes

Tour de table

- Présentez-vous
- Entreprise/Fonction
- Que connaissez-vous de l’orchestration et conteneurs

Planning

❖Cours 1

- De la virtualisation à la conteneurisation
- TP: Docker en pratique

❖Cours 2

- Orchestration des conteneurs
- TP: Kubernetes sur son PC et EKS

✤Cours 3

- TP: Déployer un serveur apache sur EKS avec helm

✤Cours 4

- TP: Déployer l’application Nextcloud sur EKS + Mysql

✤Cours 5/6

- Project: Gitops pour déployer nextcloud + mysql

❖Cours 6

- DE QCM + projet

De la virtualisation à la conteneurisation

Objectifs

Présentation des concepts de virtualisation

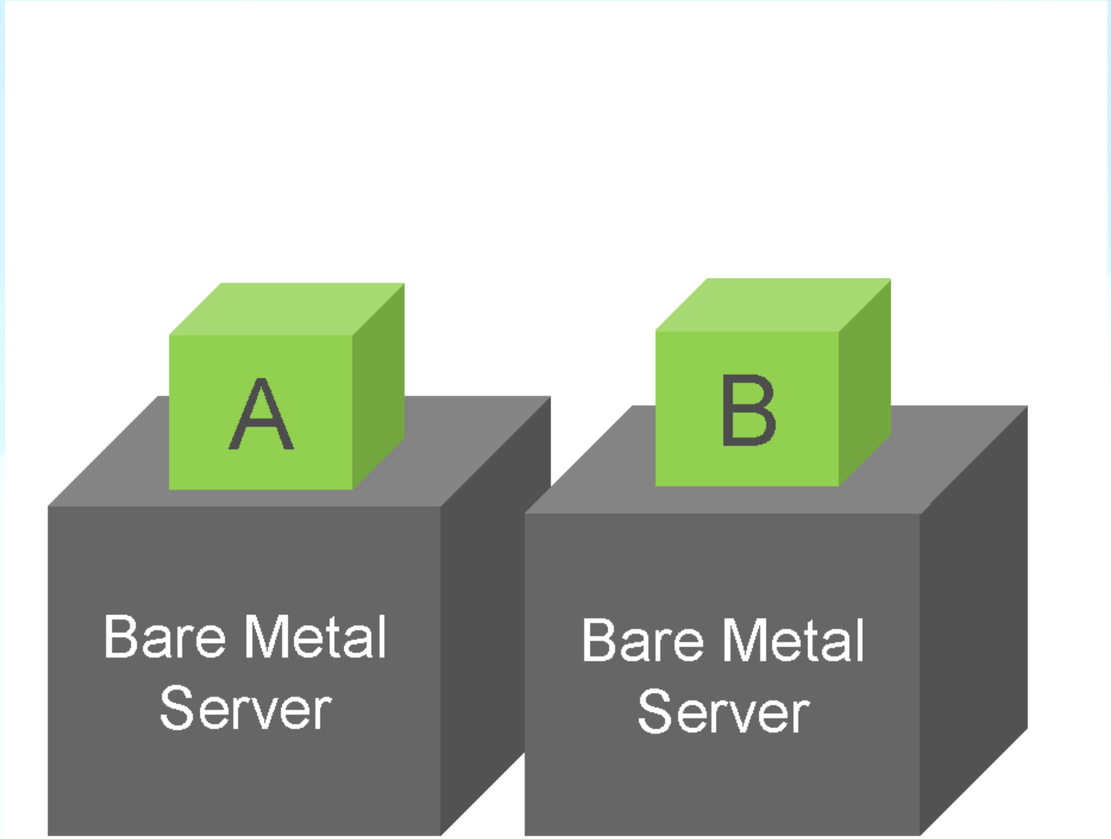
Comprendre l'intérêt de la virtualisation

Présentation de conteneurisation et son fonctionnement

Comprendre les avantages de conteneurisation et ses faiblesses

Virtualisation

Avant la virtualisation



Qu’est-ce que la virtualisation

Définition de la virtualisation

La virtualisation est **une abstraction des ressources informatiques physiques**.

Les composants matériels et logiciels peuvent être abstraits.

Un composant informatique créé dans le cadre de la virtualisation est appelé *composant virtuel* ou *logique* et peut être utilisé de la même manière que son équivalent physique.

Les différents types de virtualisation

Des serveurs

De bureau

De stockage

De réseaux

D’applications

De données

Qu’est-ce que la virtualisation

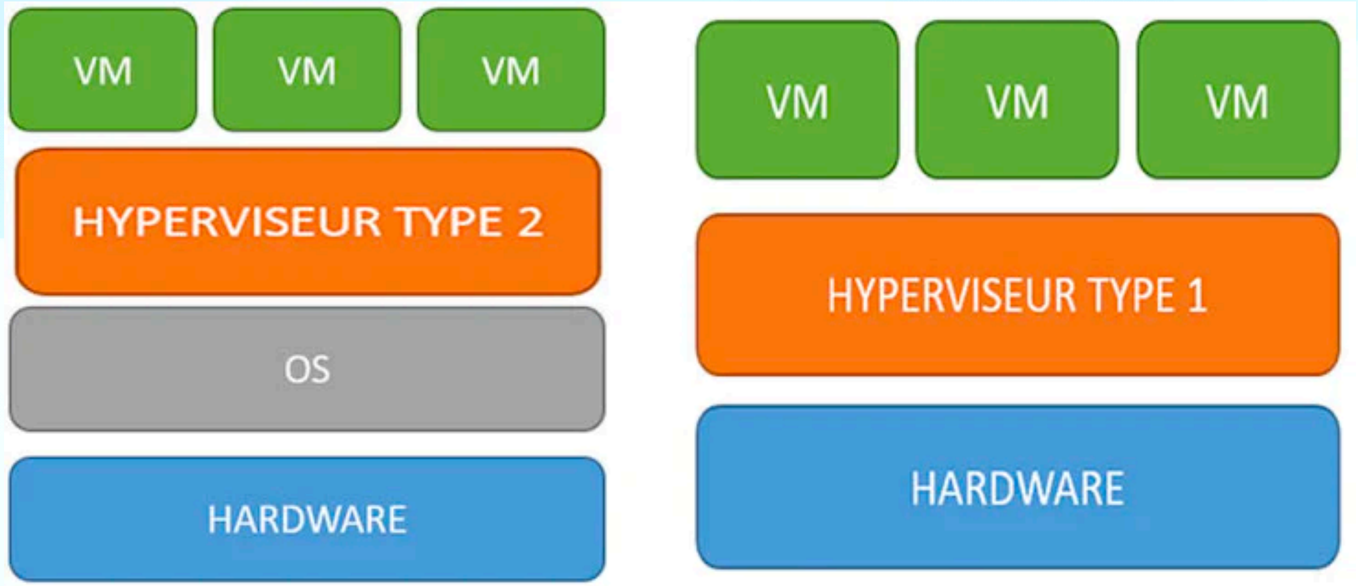
Comment fonctionne la virtualisation ?

Des logiciels, appelés hyperviseurs, isolent les ressources physiques des environnements virtuels, qui nécessitent ces ressources.

Les hyperviseurs répartissent vos ressources physiques pour permettre aux environnements virtuels de les utiliser.

Qu’est-ce que la virtualisation

Type d’hyperviseur



VirtualBox

Parallels™

KVM

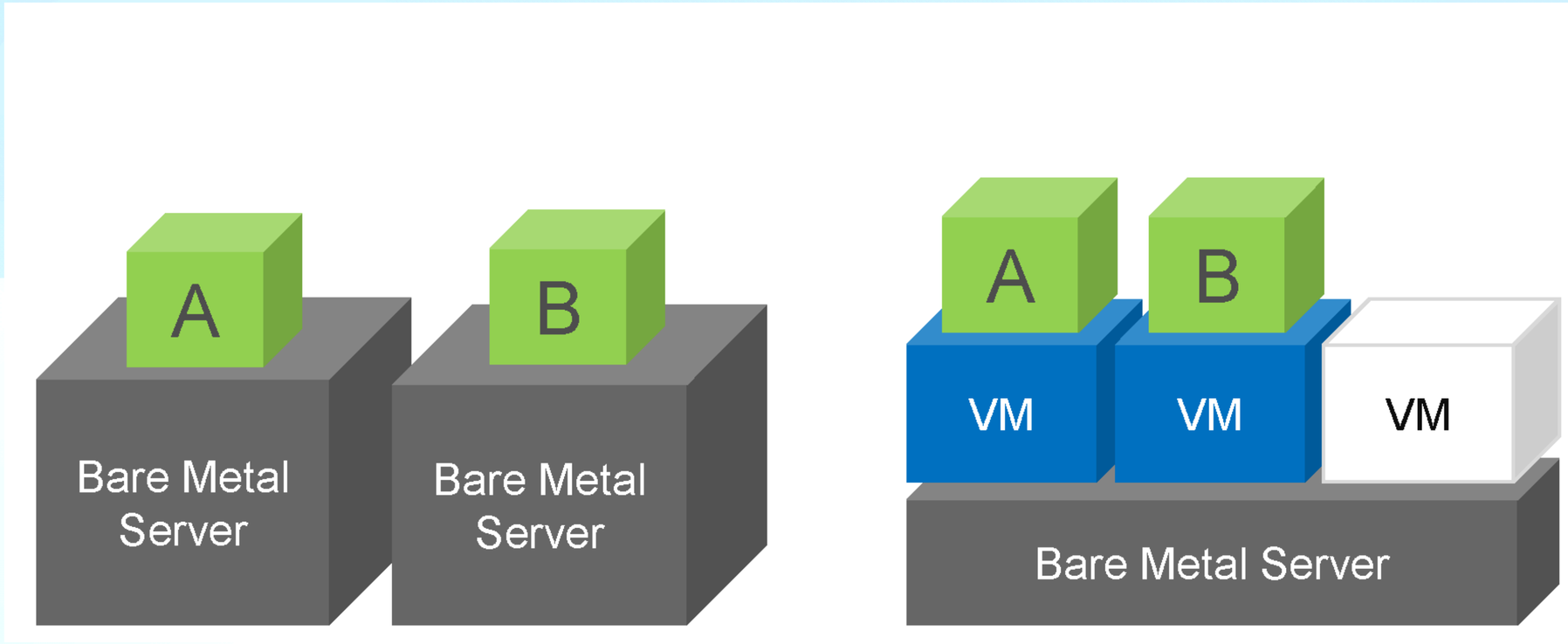
Xen™

Microsoft Hyper-V

vmware vSphere

vmware Workstation

Des serveurs physique à la virtualisation



Les avantages de la virtualisation

Utilisation plus efficace des ressources matérielles

Isolation des applications

Simplification de la gestion des infrastructures

Réduction des coûts opérationnels

Conteneurisation

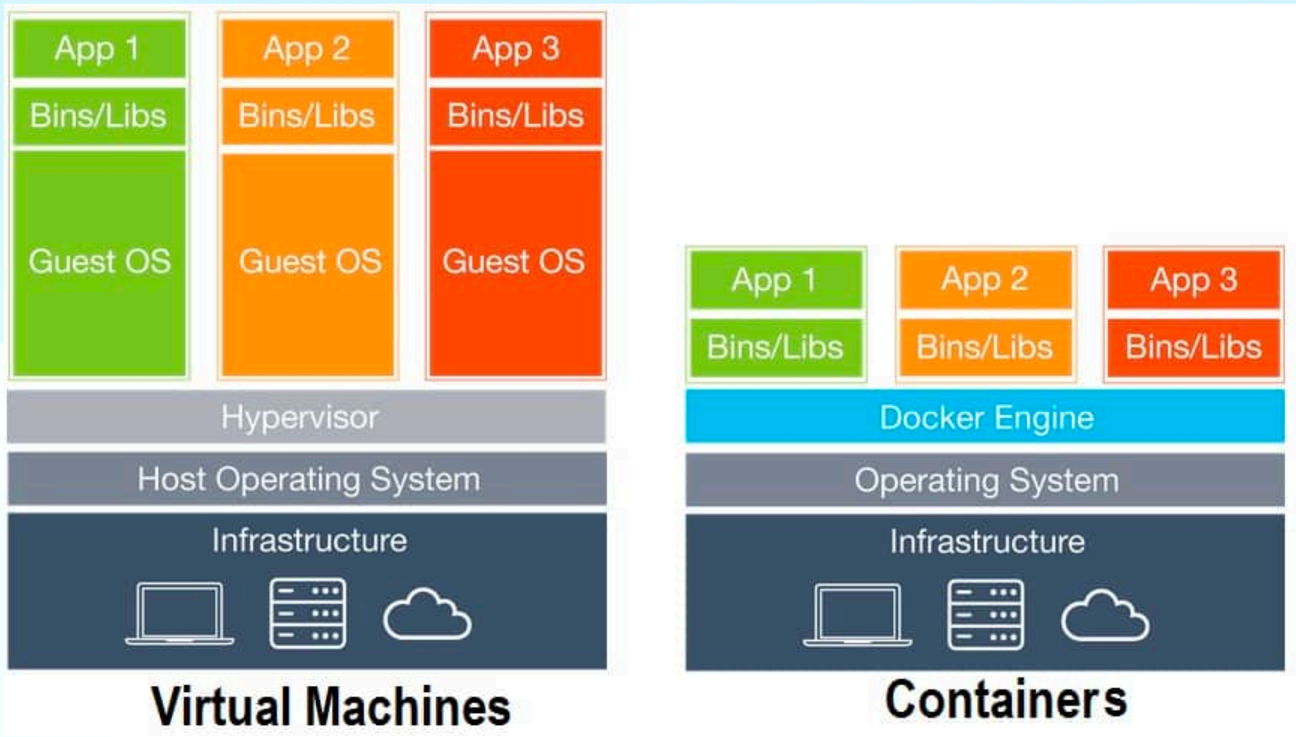
Qu'est-ce que conteneurisation ?

La conteneurisation consiste à rassembler le code du logiciel et tous ses composants (bibliothèques, frameworks et autres dépendances) de manière à les isoler dans leur propre « conteneur ».

Le logiciel ou l'application dans le conteneur peut ainsi être déplacé et exécuté de façon cohérente dans tous les environnements et sur toutes les infrastructures, indépendamment de leur système d'exploitation.

Le conteneur fonctionne comme une sorte de bulle, ou comme un environnement de calcul qui enveloppe l'application et l'isole de son entourage. C'est en fait un environnement de calcul portable complet.

Les différences entre les machines virtuelles et les conteneurs



docker

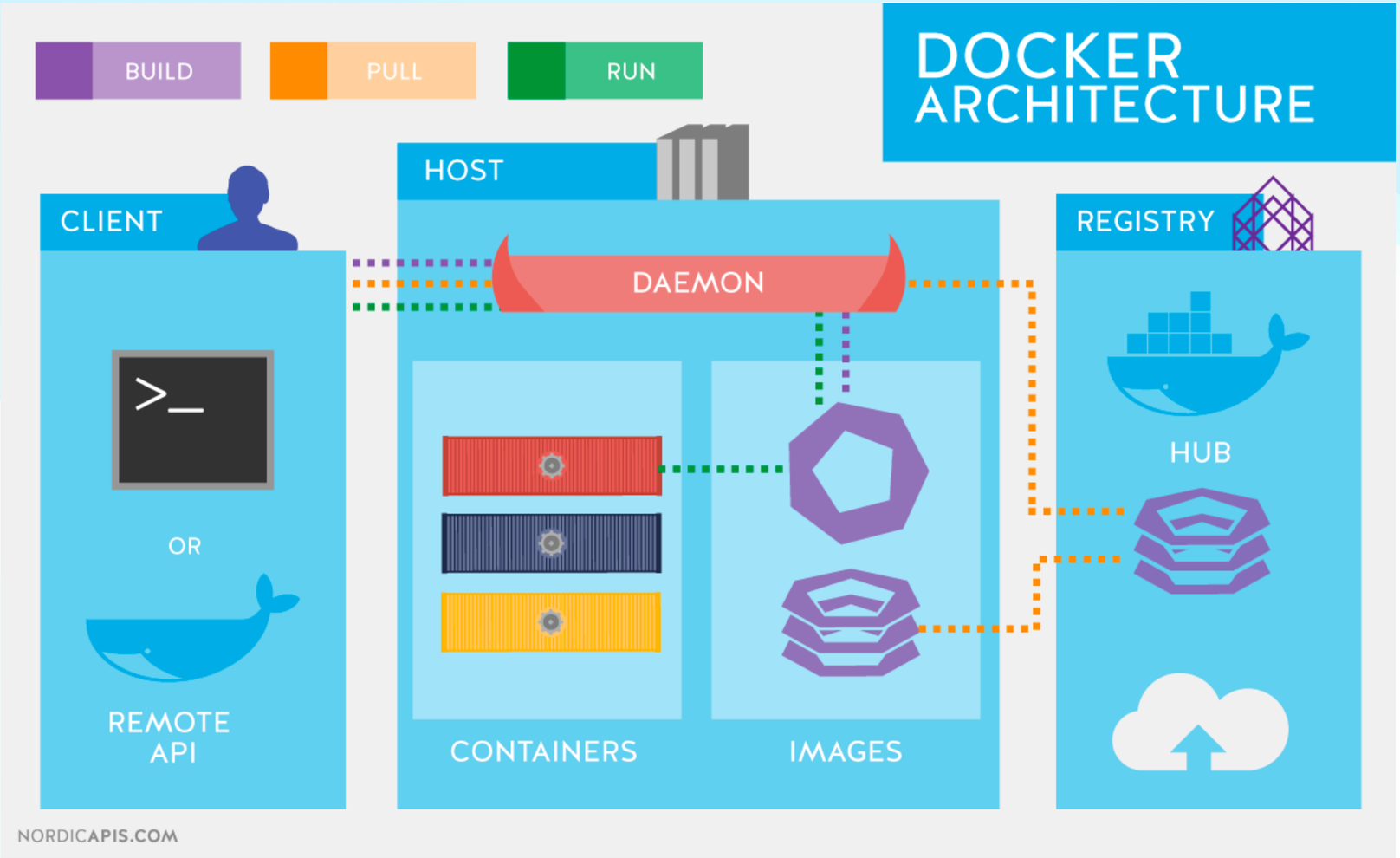
Quelles sont les solutions de conteneurisation sur le marché ?

LXC

docker

podman

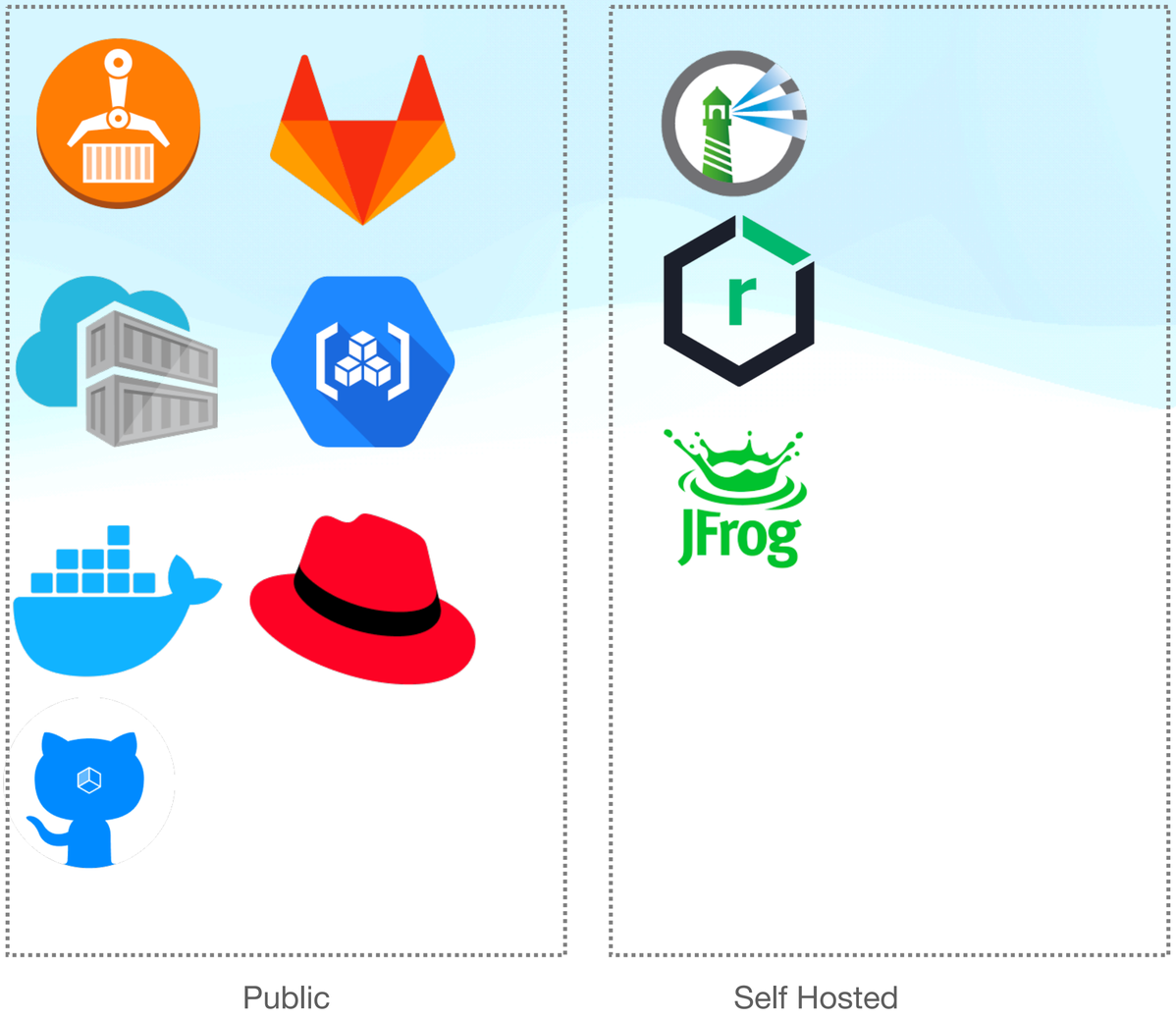
Docker Architecture



Conteneur registry et caractéristiques

Une conteneur registry est un référentiel centralisé qui stocke et distribue des images de conteneurs

- Stockage et distribution centralisés
- Sécurité
- Gestion des versions
- Catalogue de versions
- Haute disponibilité



Public

Self Hosted

Demo Registry

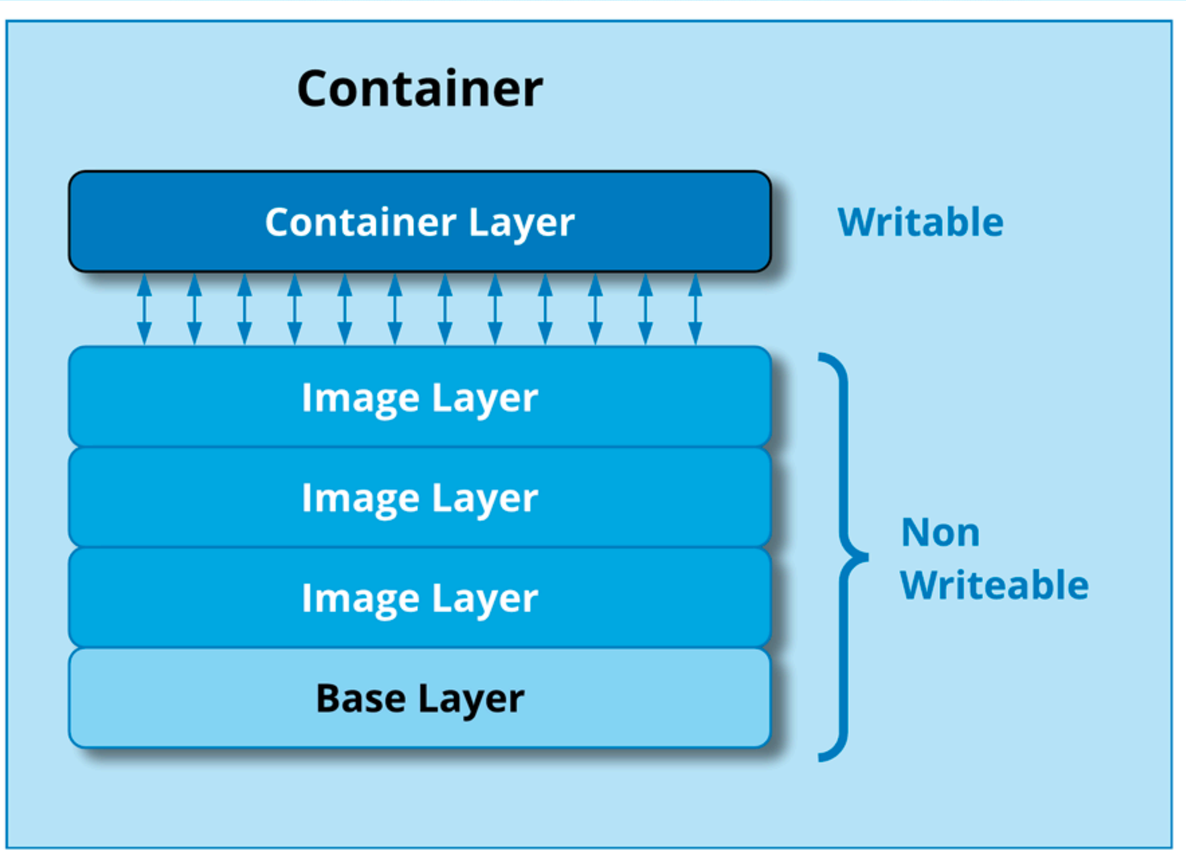
Qu’est ce qu’un image de conteneur

Une **collection ordonnée de changements d’un système de fichier** et des **paramètres d’exécution** correspondant à son utilisation à l’exécution

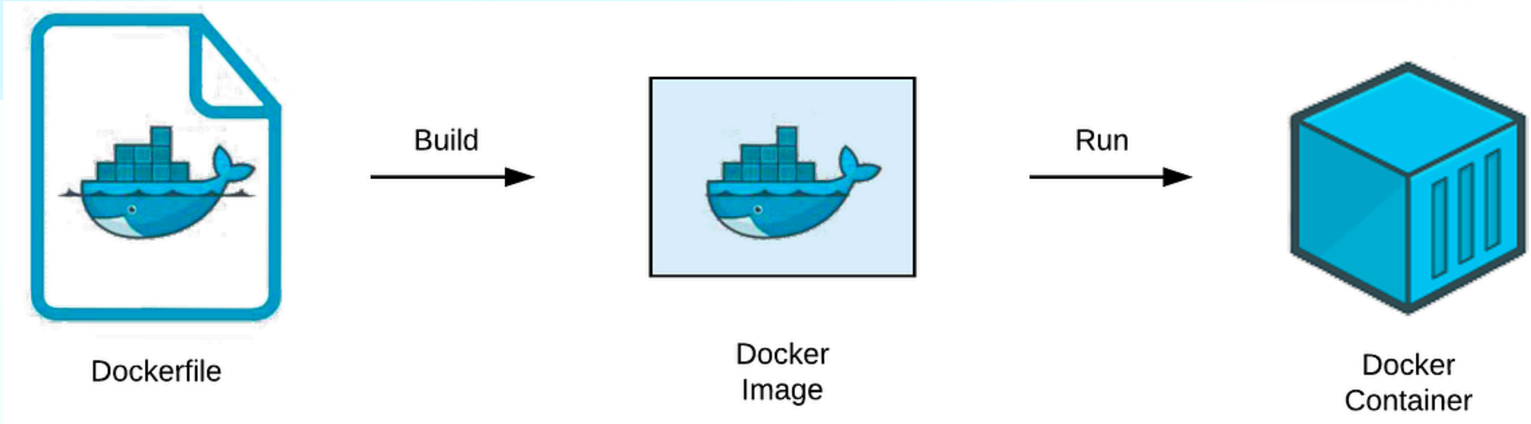
C’est **une pile de couches** dont chaque couche dépend de tous les précédents.
Chaque couche représente donc un ensemble de **changement que l’on fait au système de ficher de base**.

Ces changements sont commis par les instructions présentes dans le **Dockerfile de l’image**

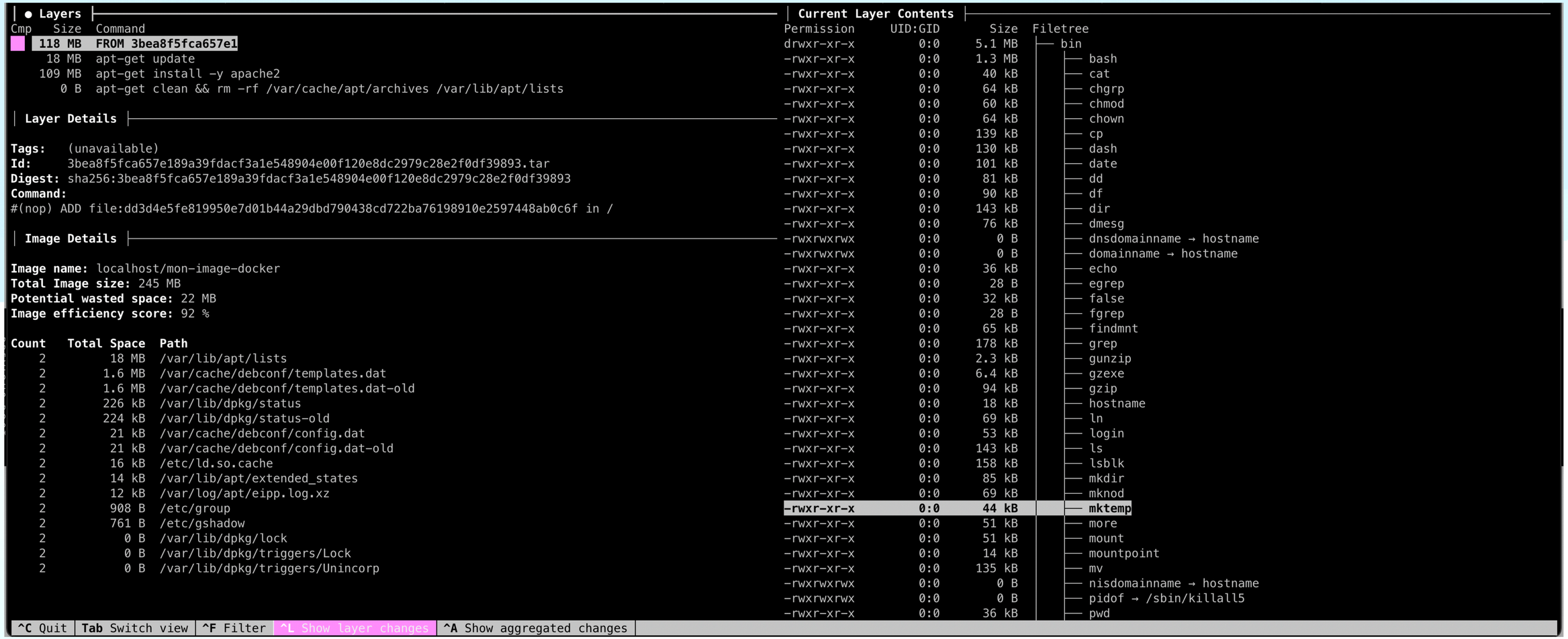
Cette décomposition en couche des images Docker permet de **partager les différents couches entre celles-ci** et **d’économiser de l’espace de stockage et en volume de téléchargement** : en effet pour des images qui partagent les mêmes couches de base, ces couches ne sont pas dupliquées ni au niveau stockage ni au niveau du téléchargement



Création de conteneurs à partir d'images



Couche d’une image docker



Création d'images avec un Dockerfile

Un **Dockerfile** est un fichier texte avec lequel vous allez pouvoir donner à Docker les instructions nécessaires pour pouvoir créer une image.

Les points importants :

- Choix de l’image de base
- Le moins de paquets installés
- Supprimer les caches
- Utiliser un USER pour le runtime du conteneur
- Minimiser le nombre de tâches

Les directives

- **FROM**
- **EXPOSE**
- **USER**
- **WORDIR**
- **ADD**
- **COPY**
- **RUN**
- **CMD/ENTRYPOINY**

Aller plus loin

docker

Exemple de Dockerfile

```
FROM debian:latest

RUN apt-get update

RUN apt-get install -y apache2

RUN apt-get clean && rm -rf /var/cache/apt/archives /var/lib/apt/lists

CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

Scanning des images de conteneur

Pourquoi ?

Sécurité: Les images Docker peuvent contenir des logiciels obsolètes ou des configurations mal sécurisées, ce qui peut entraîner des failles de sécurité.C

Conformité: Dans certains cas, les entreprises sont tenues de respecter des normes de sécurité et de conformité, telles que PCI DSS, HIPAA ou GDPR.

Fiabilité: Les images Docker provenant de sources tierces peuvent être compromises ou contenir des logiciels malveillants.

Gestion des risques: En scannant les images Docker, les équipes de sécurité peuvent identifier les risques potentiels et prendre des mesures pour atténuer ces risques avant qu'ils ne se transforment en problèmes de sécurité plus importants.

clair
A Container Image Security Analyzer by CoreOS

aqua trivy

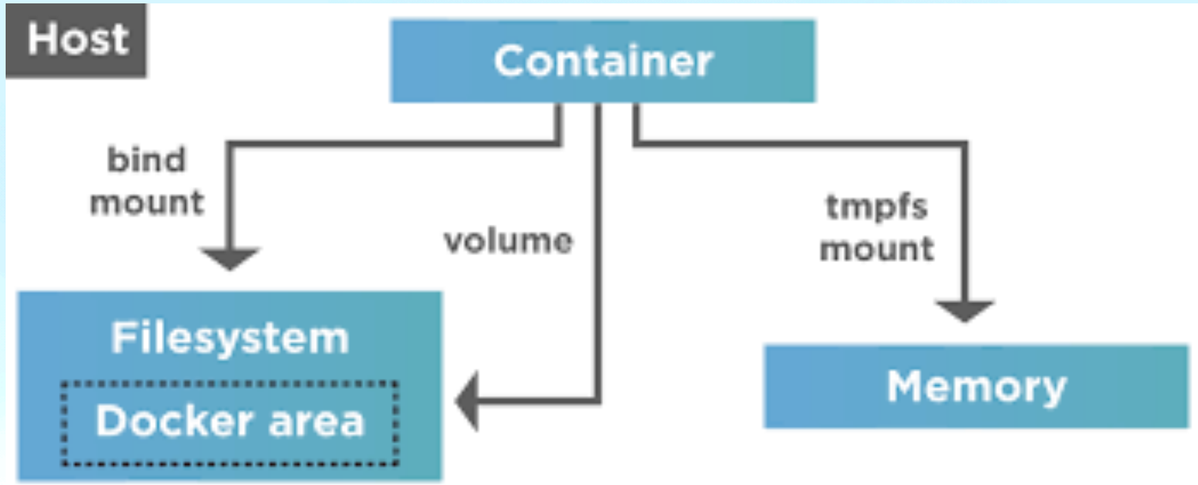
Demo build d’image et scanning

Conteneur persistance de la donnée

Les conteneurs sont conçus pour être, la plupart du
temps, **immuables** et **éphémères.**

Si des modifications sont nécessaires, on redéploie de nouveaux conteneurs basés sur des images différentes.

Cela permet d'obtenir de la **fiabilité** et de la **consistance**, tout en permettant la reproduction de tous les changements.



Volumes: un emplacement de stockage de données qui peuvent être monté à l'intérieur de plusieurs conteneurs

Bind Mounts: monter un répertoire ou un fichier de l'hôte à l'intérieur d'un conteneur seulement

Demo persistance de la donnée

Communication entre les conteneurs

Network: Les réseaux Docker permettent aux conteneurs Docker de communiquer entre eux, ainsi qu'avec les hôtes et les réseaux externes. Les réseaux Docker sont utilisés pour isoler les applications et pour assurer la sécurité.

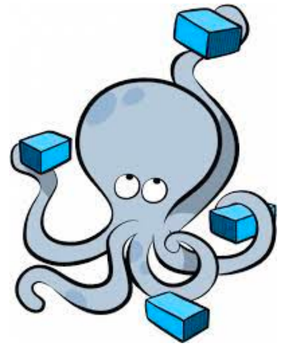
Exposition de ports: Docker permet d'exposer des ports de conteneurs Docker pour permettre la communication avec les applications externes.

Composer plusieurs conteneurs

Docker/Podman Compose sont des outils open-source permettant de décrire et de lancer des applications multi-conteneurs dans Docker.

Ils utilisent un fichier YAML pour décrire des services, des variables d'environnement, des réseaux et des volumes requis pour chaque conteneur, ainsi que les configurations associées.

Ensuite, ils utilisent ce fichier pour lancer et gérer l'ensemble des conteneurs de manière cohérente.



Demo compose et exposition

Limites et faiblesses

Limitations de ressources: Les conteneurs Docker partagent les ressources du système d'exploitation de l'hôte, ce qui peut entraîner des limitations en termes de capacité de stockage, de puissance de traitement et de mémoire. Si plusieurs conteneurs sont exécutés sur un même hôte, il peut y avoir une compétition pour les ressources, ce qui peut affecter les performances des conteneurs.

Dépendances système: Les conteneurs Docker dépendent des bibliothèques et des composants système du système d'exploitation de l'hôte. Cela peut poser des problèmes de compatibilité lors de la migration vers d'autres environnements.

Sécurité: Bien que les conteneurs Docker soient isolés les uns des autres, il existe toujours un risque de failles de sécurité si les images ou les conteneurs ne sont pas correctement configurés. Les conteneurs doivent être sécurisés et les images doivent être vérifiées pour les vulnérabilités.

Complexité: La conteneurisation peut ajouter de la complexité à l'infrastructure de déploiement. La gestion de plusieurs conteneurs et de plusieurs images peut être complexe et nécessite une planification minutieuse.

Surcharge réseau: Les conteneurs peuvent avoir besoin de communiquer avec d'autres conteneurs ou avec des serveurs externes, ce qui peut entraîner une surcharge du réseau.

Durée de vie: Les conteneurs Docker sont conçus pour être temporaires et éphémères. Bien que cela soit avantageux pour la flexibilité et la gestion des ressources, cela peut rendre difficile la persistance de données à long terme.

Les avantages de conteneurisation

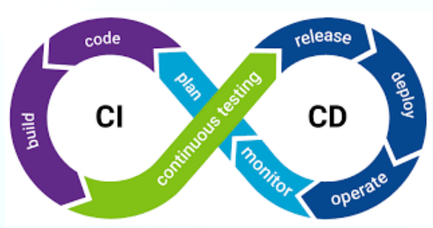
Création et déploiement rapides d’applications

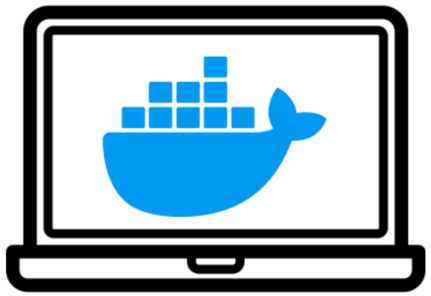
Portabilité des applications

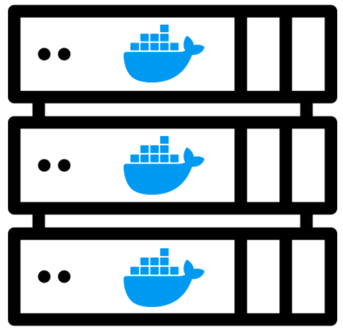
Isolation des applications

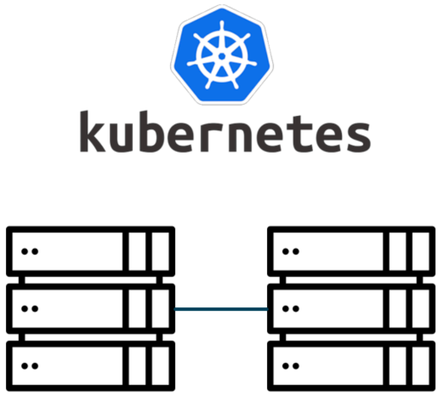
Utilisation efficace des ressources matérielles

Cas d’utilisation de la conteneurisation en entreprise

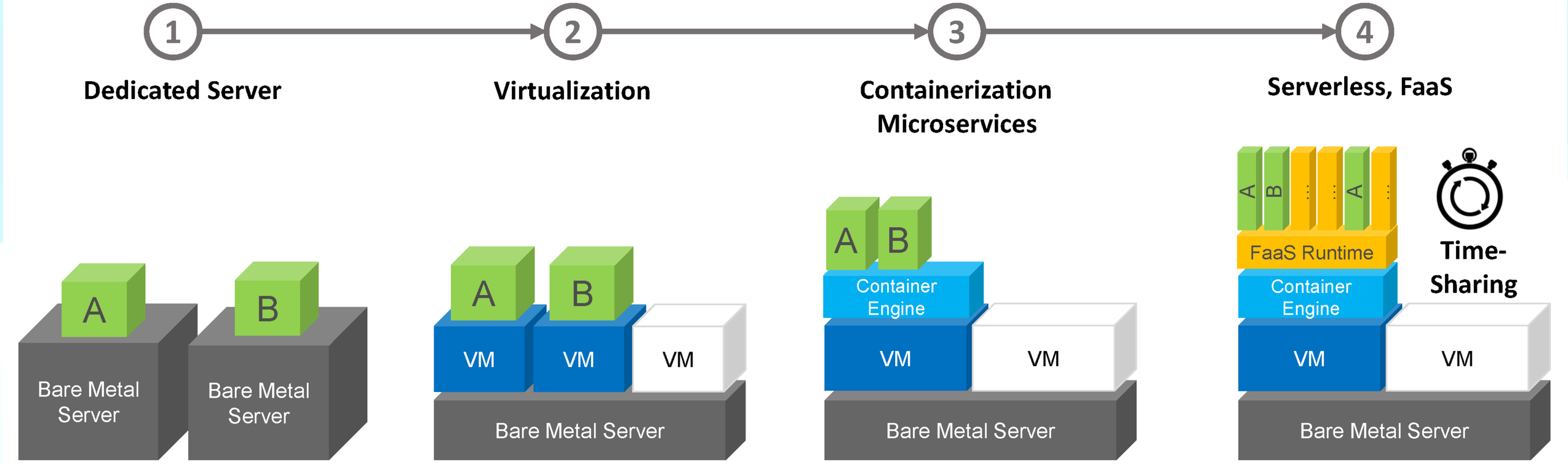








Evolution depuis les serveurs physiques



Conclusion

La conteneurisation est une technologie qui a révolutionné le déploiement d'applications modernes en offrant une portabilité, une flexibilité et une scalabilité accrues.

Les conteneurs Docker ont permis de créer des environnements d'exécution isolés et reproductibles pour les applications, ce qui facilite leur déploiement et leur gestion.

Les avantages de la conteneurisation incluent une meilleure utilisation des ressources matérielles, une réduction des coûts d'infrastructure, une amélioration de la vitesse de déploiement et une isolation efficace des applications.

Cependant, il est important de prendre en compte les limites et les faiblesses de la conteneurisation, notamment en termes de dépendances système, de sécurité et de complexité de gestion.

Annexes

- https://www.redhat.com/fr/topics/containers/whats-a-linux-container
- https://www.padok.fr/blog/image-docker-dive
- https://docs.docker.com/engine/reference/commandline/docker/
- https://podman.io/getting-started/