

The background of the slide is a photograph of a shipping yard. In the center, a large red shipping container is being hoisted by a crane, suspended by several thick cables. To the left, there are stacks of yellow and red containers. In the distance, more stacks of various colored containers (blue, yellow, red) are visible under a bright blue sky with scattered white clouds. A large, semi-transparent white circle is overlaid on the right side of the image, containing the title and author information.

1

Introduction aux conteneurs

Ameur Kais

Introduction

- Les conteneurs, ainsi que les technologies de conteneurisation telles que Docker et Kubernetes , sont devenus des composants de plus en plus courants dans les boîtes à outils de nombreux développeurs.
- L'objectif de la conteneurisation, à la base, est d'offrir un meilleur moyen de créer, de conditionner et de déployer des logiciels dans différents environnements d'une manière prévisible et facile à gérer.

Que sont les conteneurs? i

- Les conteneurs sont une technologie de virtualisation du système d'exploitation utilisée pour emballer les applications et leurs dépendances et les exécuter dans des environnements isolés.
- Ils fournissent une méthode légère d'emballage et de déploiement d'applications de manière standardisée sur de nombreux types d'infrastructure différents.
- Les conteneurs fonctionnent de manière cohérente sur n'importe quel hôte compatible avec les conteneurs, de sorte que les développeurs peuvent tester localement le même logiciel qu'ils déploieront plus tard dans des environnements de production complets.
- Le format de conteneur garantit également que les dépendances d'application sont intégrées à l'image elle-même, ce qui simplifie les processus de transfert et de libération.
- Étant donné que les hôtes et les plates-formes qui exécutent les conteneurs sont génériques, la gestion de l'infrastructure des systèmes basés sur des conteneurs peut être normalisée.

Que sont les conteneurs? ii

- Les conteneurs sont créés à partir **d'images** de **conteneur** : des ensembles qui représentent le système, les applications et l'environnement du conteneur.
- Les images de conteneurs agissent comme des modèles pour créer des conteneurs spécifiques, et la même image peut être utilisée pour générer un nombre illimité de conteneurs en cours d'exécution.
- Les utilisateurs peuvent télécharger des conteneurs prédéfinis à partir de sources externes ou créer leurs propres images adaptées à leurs besoins et les déployer sur un registre d'images de conteneurs local ou distant.

Machines virtuelles vs conteneurs i

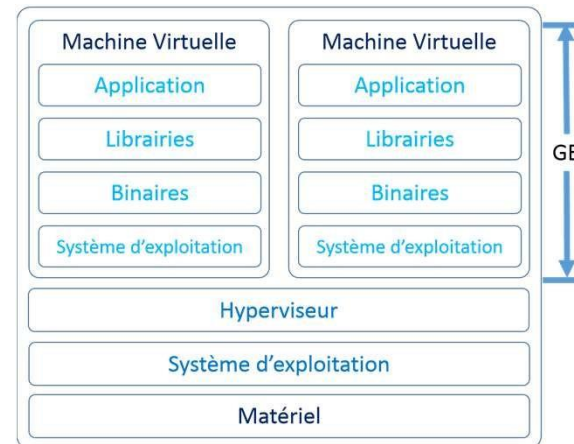
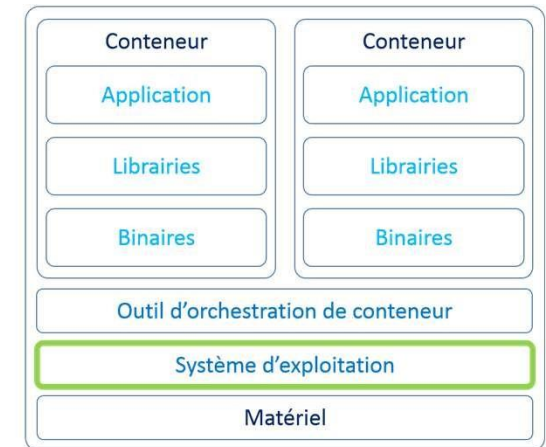
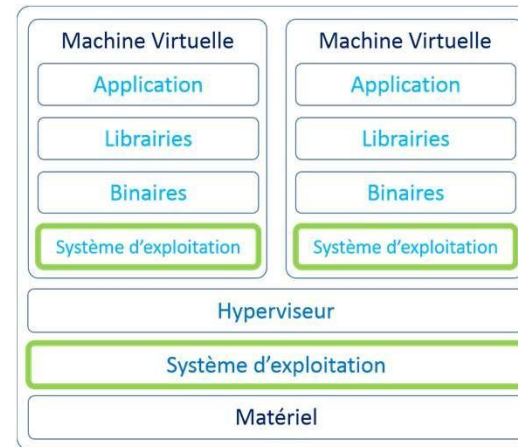
- **Les machines virtuelles** , technologie de virtualisation matérielle qui permet de virtualiser entièrement le matériel et les ressources d'un ordinateur. Un système d'exploitation invité distinct gère la machine virtuelle, complètement distinct du système d'exploitation exécuté sur le système hôte. Sur le système hôte, un logiciel appelé hyperviseur est responsable du démarrage, de l'arrêt et de la gestion des machines virtuelles.
- Les machines virtuelles sont exploitées comme des ordinateurs complètement distincts et ne peuvent pas affecter le système hôte ou d'autres machines virtuelles et offrent de ce fait une isolation et une sécurité optimales.
- Cependant, la virtualisation d'un ordinateur entier nécessite une quantité importante de ressources.
- Le provisionnement et le démarrage de la machine virtuelle peuvent être assez lents. De même, étant donné que la machine virtuelle fonctionne comme une machine indépendante, les administrateurs doivent souvent adopter des outils et des processus de gestion d'infrastructure pour mettre à jour et exécuter les environnements individuels.
- Le résultat final ne diffère pas de manière significative de la gestion d'un parc d'ordinateurs physiques.

Machines virtuelles vs conteneurs ii

- **Les conteneurs** plutôt que de virtualiser tout l'ordinateur, les conteneurs virtualisent directement le système d'exploitation.
- Ils s'exécutent comme des processus spécialisés gérés par le noyau du système d'exploitation hôte.
- Les conteneurs fonctionnent comme s'ils contrôlaient totalement l'ordinateur.
- Les conteneurs sont gérés de manière similaire aux applications.
- Les conteneurs occupent un espace qui se situe quelque part entre la forte isolation des machines virtuelles et la gestion native des processus conventionnels.
- Les conteneurs offrent une compartimentation et une virtualisation axée sur les processus, qui offrent un bon équilibre entre confinement, flexibilité et vitesse.
- Les conteneurs sont relativement légers: ils ne contiennent que les bibliothèques et les outils nécessaires à l'exécution de l'application conteneurisée. Ils sont donc plus compacts que les machines virtuelles et démarrent plus rapidement.

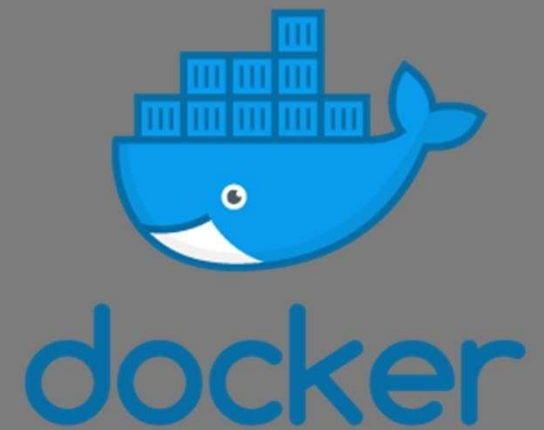


Machines virtuelles vs conteneurs iii



Docker

Container Engine



Qu'est-ce que Docker?

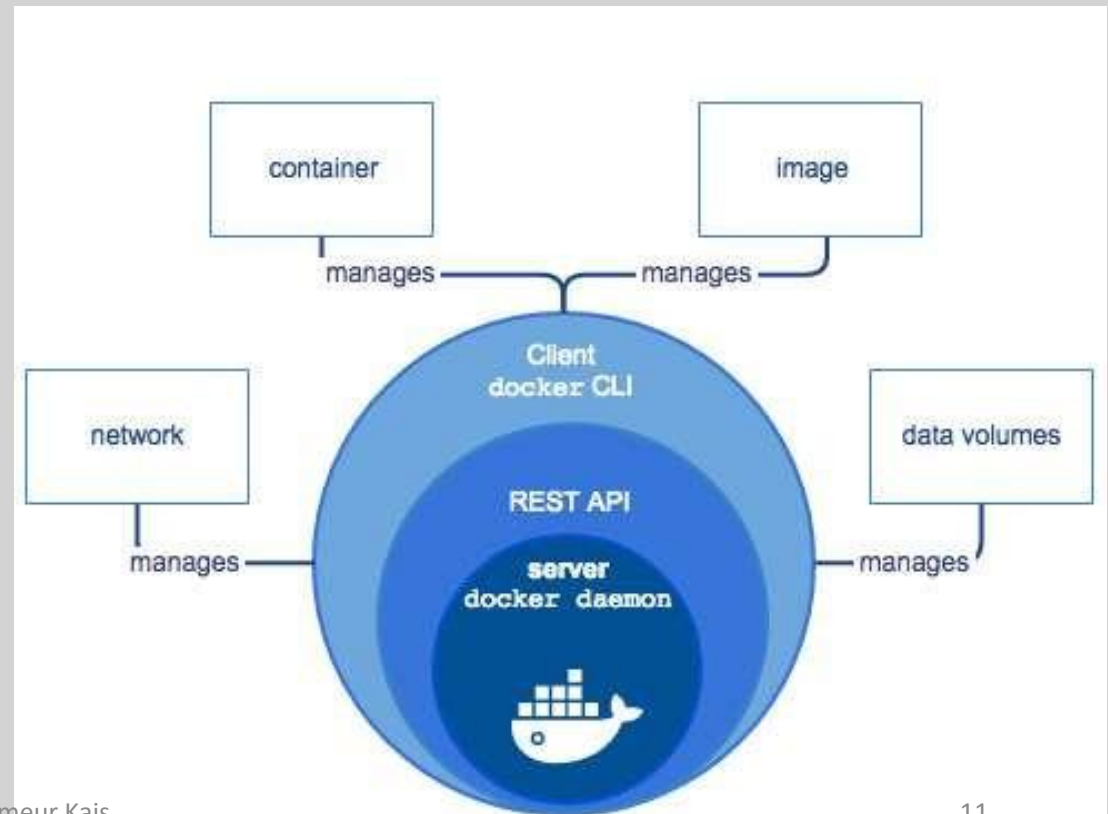
- Docker n'est qu'une des nombreuses implémentations de conteneurs sur Linux, il a la particularité d'être le point d'entrée le plus courant dans le monde des conteneurs et la solution la plus couramment déployée.
- Alors que des **normes ouvertes (Open Container Initiative – OCI)** ont été développées pour les conteneurs afin d'assurer l'interopérabilité, la plupart des plates-formes et outils liés aux conteneurs considèrent Docker comme leur principale cible lors du test et de la publication de logiciels.
- Docker n'est peut-être pas toujours la solution la plus performante pour un environnement donné, mais c'est probablement l'une des options les plus testées.

Concepts Docker

- **Flexible** : même les applications les plus complexes peuvent être conteneurisées.
- **Léger** : les conteneurs exploitent et partagent le noyau hôte, ce qui les rend beaucoup plus efficaces en termes de ressources système que les machines virtuelles.
- **Portable** : vous pouvez créer localement, déployer sur le cloud et exécuter n'importe où.
- **Faiblement couplé** : les conteneurs sont hautement autonomes et encapsulés, ce qui vous permet de remplacer ou de mettre à niveau l'un sans en perturber les autres.
- **Évolutif** : vous pouvez augmenter et distribuer automatiquement les répliques de conteneurs dans un centre de données.
- **Sécurisé** : les conteneurs appliquent des contraintes et des isollements agressifs aux processus sans aucune configuration requise de la part de l'utilisateur.

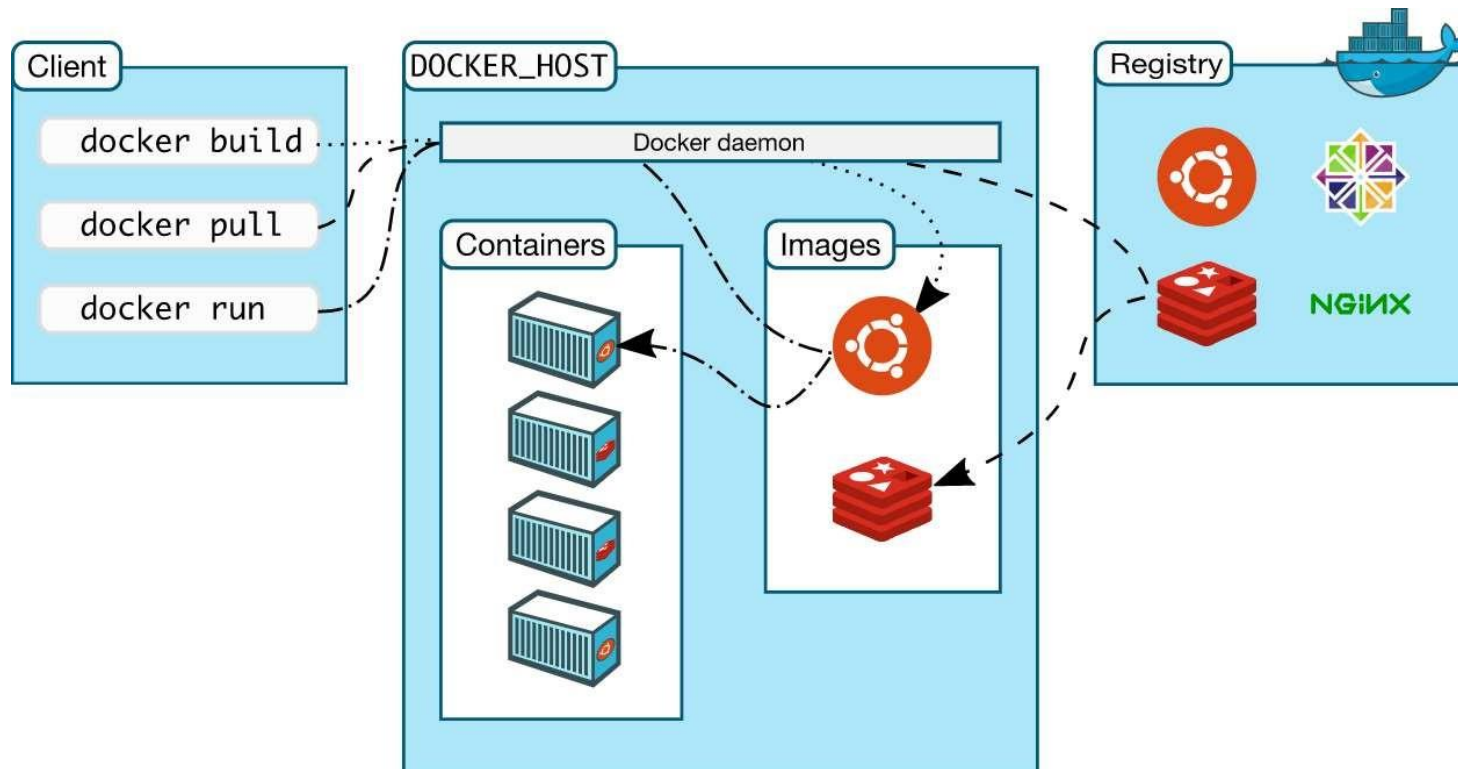
Docker Engine

- Docker Engine est une application client-serveur avec ces composants majeurs:
- Un serveur de type longue durée d'exécution appelé processus démon (dockerd).
- Une API REST qui spécifie les interfaces que les programmes peuvent utiliser pour parler au démon et lui indiquer quoi faire.
- Un client d'interface de ligne de commande (CLI) (docker).

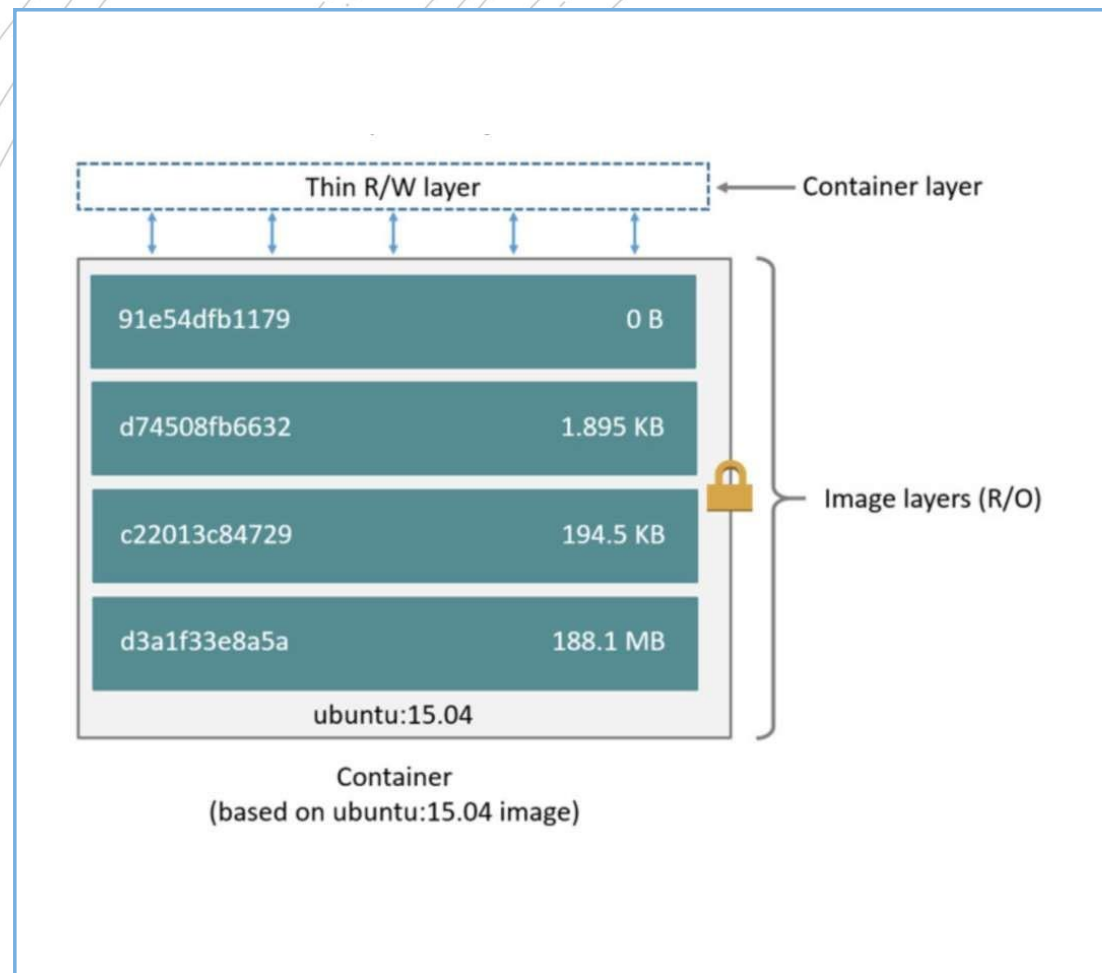


Images et conteneurs

- Un conteneur est un processus en cours d'exécution, avec quelques fonctionnalités d'encapsulation supplémentaires afin de le maintenir isolé de l'hôte et des autres conteneurs.
- Un aspect important de l'isolation est que chaque conteneur interagit avec son propre système de fichiers privé
- Ce système de fichiers est fourni par une image Docker .
- Une image comprend tout ce qui est nécessaire pour exécuter une application - le code ou le binaire, les environnements d'exécution, les dépendances et tout autre objet du système de fichiers requis.



Architecture Docker



Les images Docker

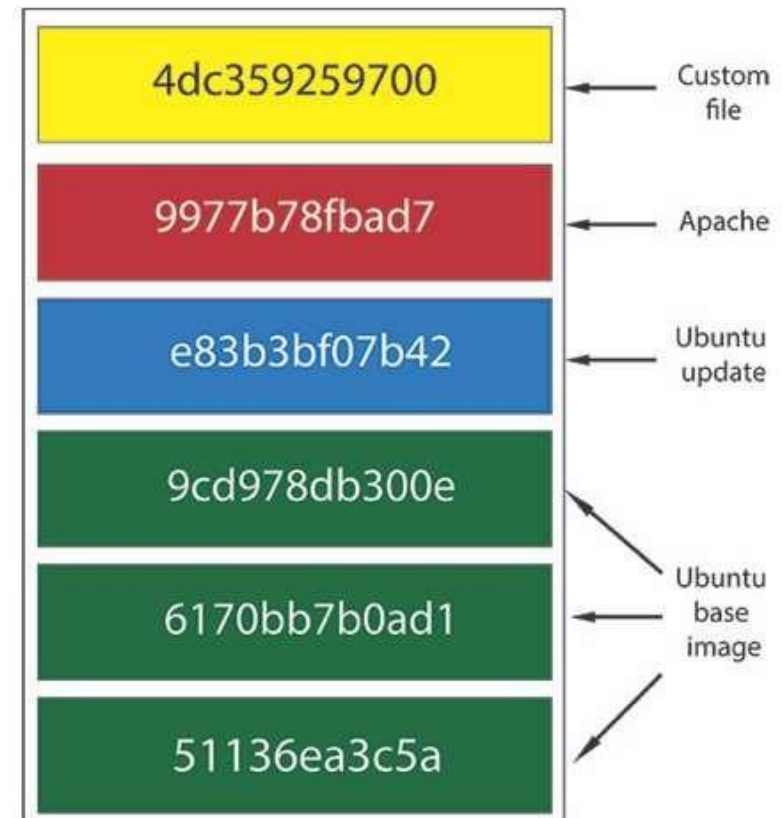
- Une image se décompose en layers
- Une image Docker fait référence à une liste de couches en lecture seule qui représentent les différences dans le système de fichiers.
- Une conteneur = une image + un layer r/w
- Une image, peut servir de base pour plusieurs conteneurs

Différence entre image et conteneur

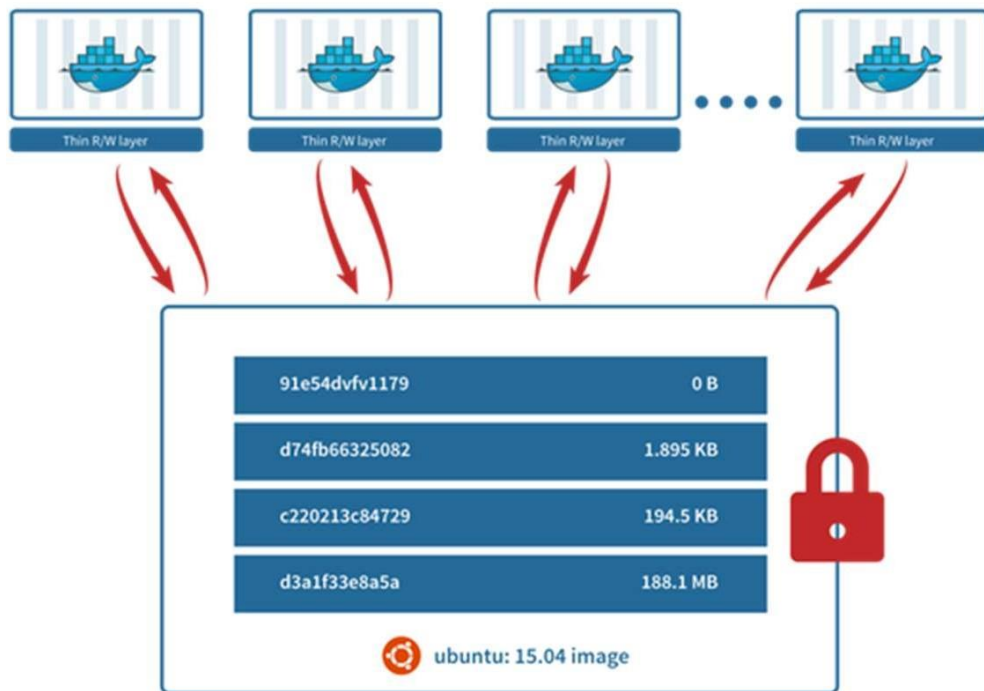
- La principale différence entre un conteneur et une image est la couche inscriptible supérieure.
- Toutes les écritures dans le conteneur qui ajoutent de nouvelles ou modifient des données existantes sont stockées dans cette couche inscriptible.
- Lorsque le conteneur est supprimé, la couche inscriptible est également supprimée.
- L'image sous-jacente reste inchangée.
- Étant donné que chaque conteneur a sa propre couche de conteneur inscriptible et que toutes les modifications sont stockées dans cette couche de conteneur, plusieurs conteneurs peuvent partager l'accès à la même image sous-jacente tout en ayant leur propre état de données

Layers

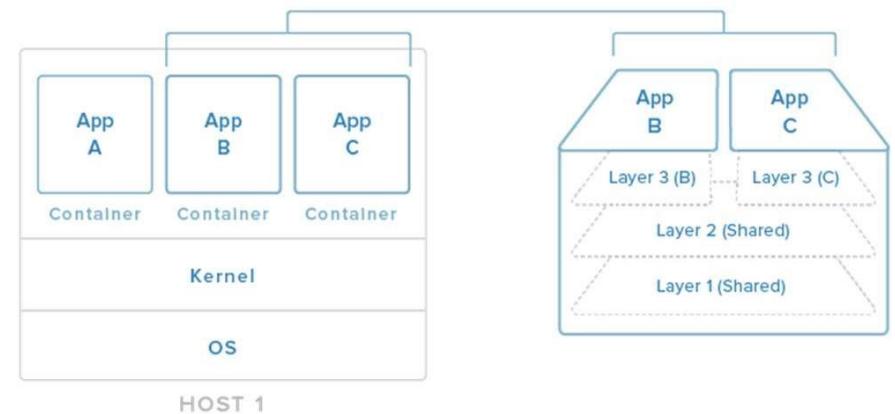
- La création de conteneurs d'applications à utiliser implique la création d'une image de base Docker sur laquelle certains ou tous les conteneurs d'applications sont basés.
- Les layers peuvent être réutilisés entre différents conteneurs
- Gestion optimisée de l'espace disque.
- L'utilisation d'une image de base permet de réutiliser les configurations d'image car de nombreuses applications partageront des dépendances, des bibliothèques et une configuration.



Plusieurs conteneurs



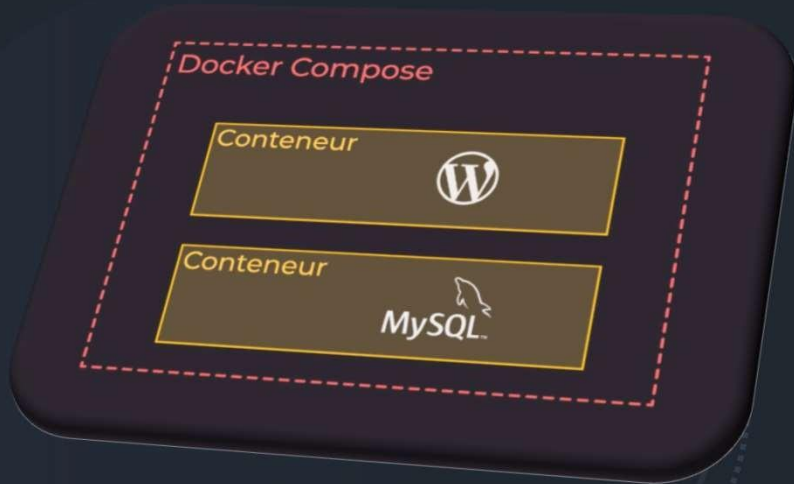
CONTAINER OVERVIEW



Composition de conteneurs

En général, le workflow de développement ressemble à ceci:

- Créez et testez des conteneurs individuels pour chaque composant de votre application en créant d'abord des images Docker.
- Assemblez vos conteneurs et votre infrastructure de support dans une application complète.
- Testez, partagez et déployez votre application conteneurisée complète.

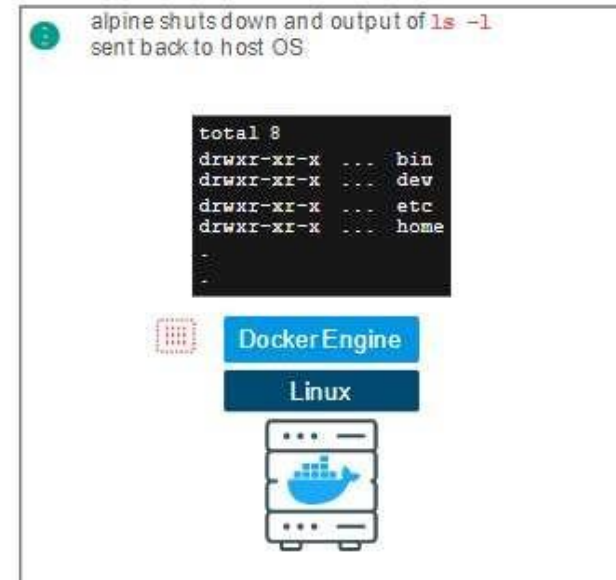
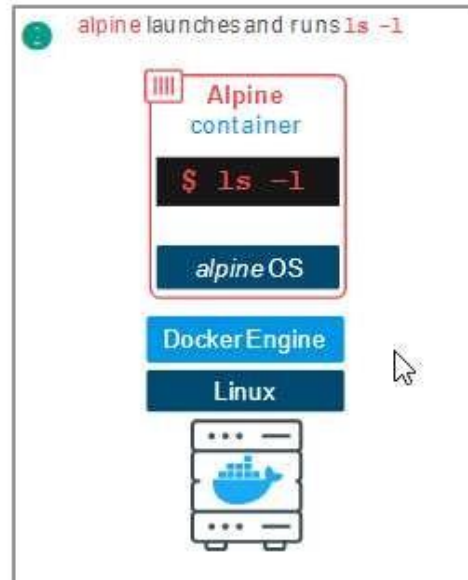
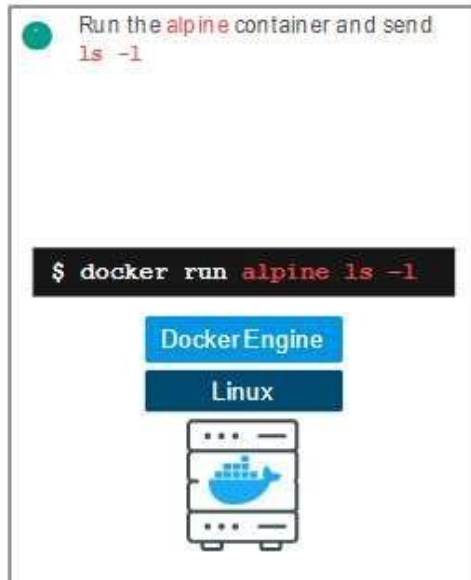


Créez et exécutez un conteneur

Hello World: What Happened?



docker run Details

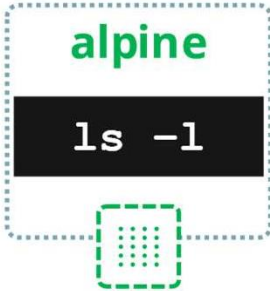


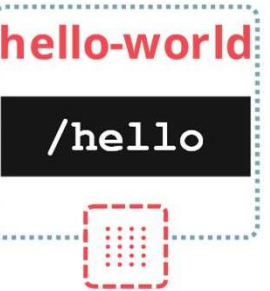


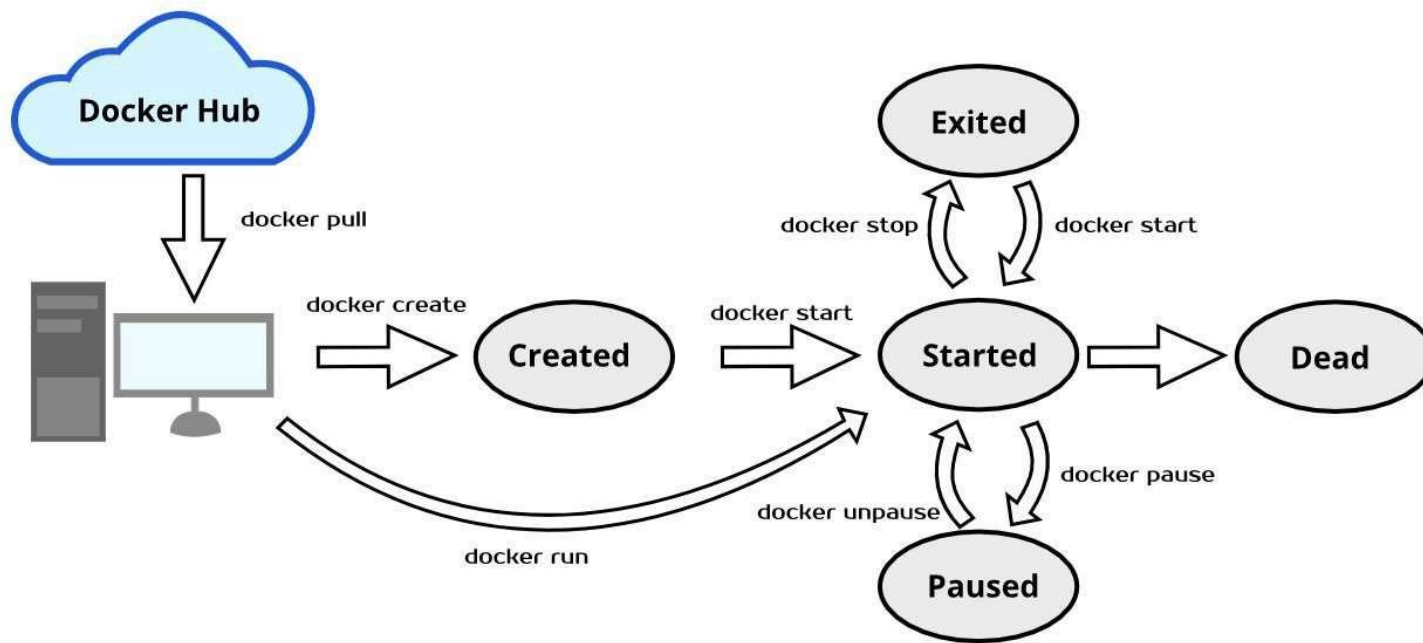
docker run

Docker Container Instances

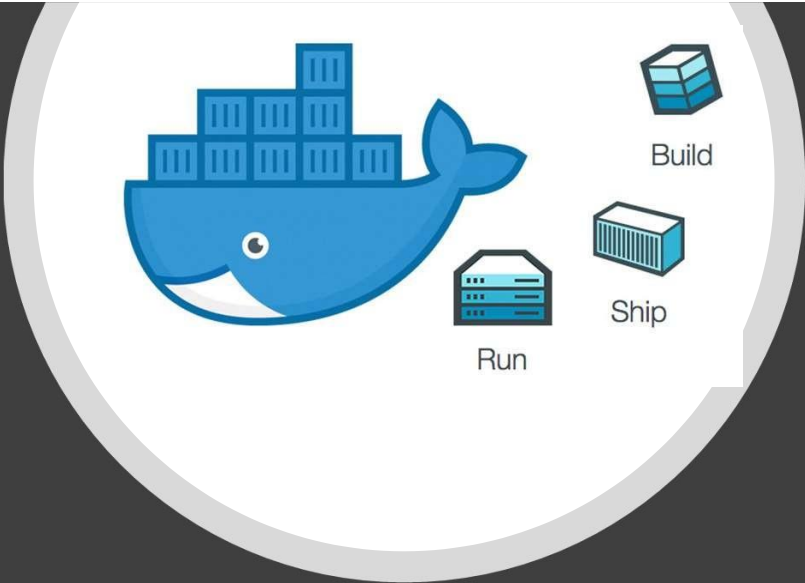
Output of `docker container ls -a`

Container Instances

	Exited (0)	Exited (0)	Exited (0)	Exited (0)
				
Container IDs	ff0a5c3750b9	36171a5da744	a6a9d46d0b2f	c317d0a9e3d2
Container Names	elated_ramanujan	fervent_newton	lonely_kilby	stupefied_mcclintock



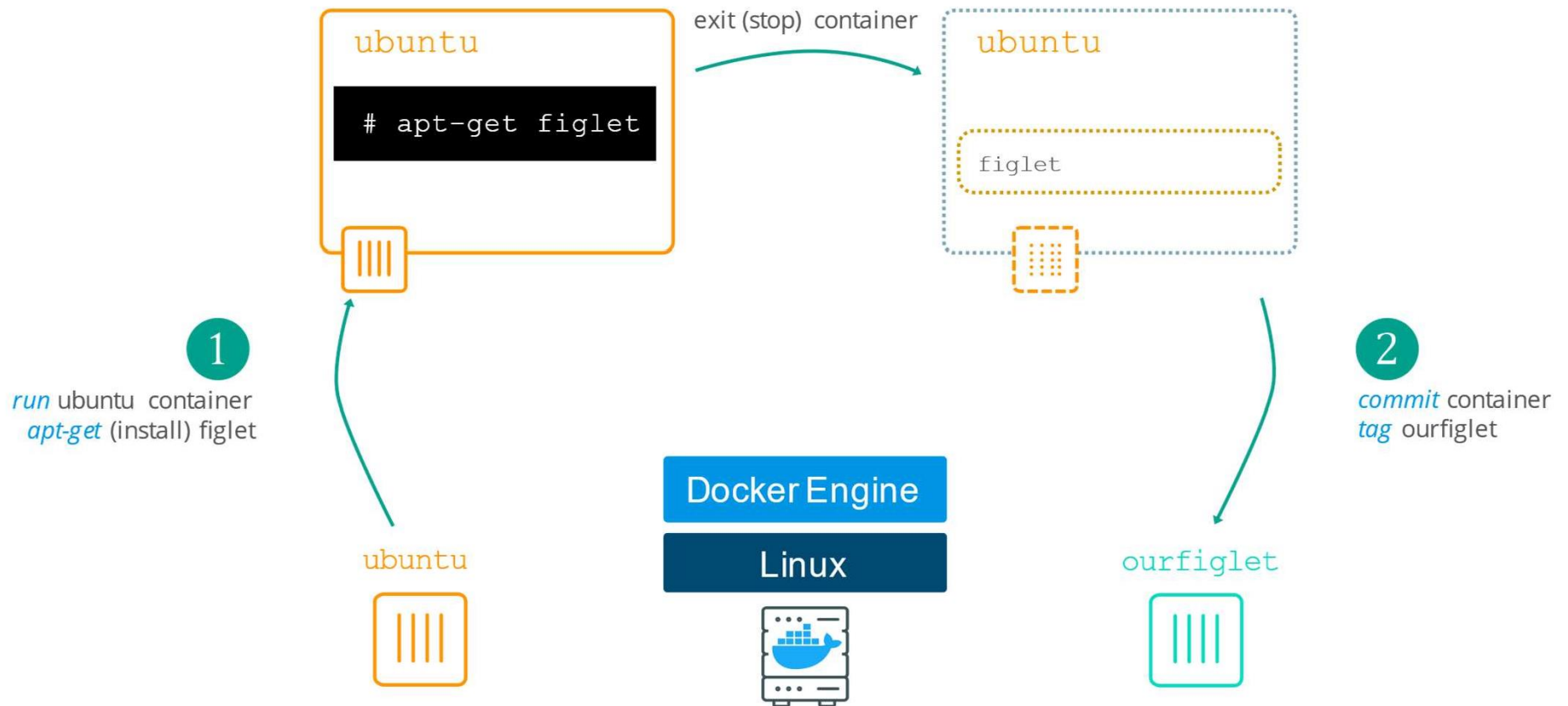
Docker Lifecycle



Créez une image Docker



Image Creation: Instance Promotion



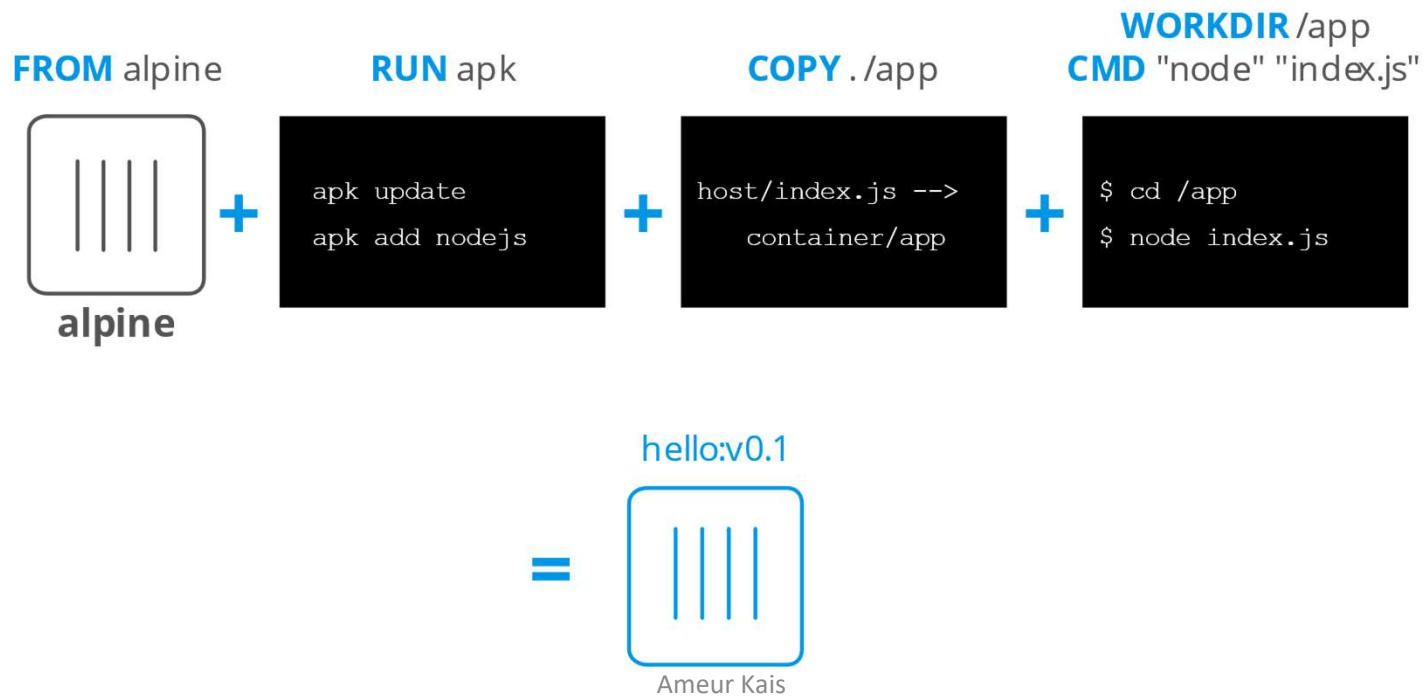
Dockerfile

- Au lieu de créer une image binaire statique, nous pouvons utiliser un fichier appelé Dockerfile pour créer une image.
- Le résultat final est essentiellement le même, mais avec un Dockerfile, nous fournissons les instructions pour construire l'image, plutôt que juste les fichiers binaires bruts.
- Ceci est utile car il devient beaucoup plus facile de gérer les changements, d'autant plus que vos images deviennent plus grandes et plus complexes.

Dockerfiles

Dockerfile:

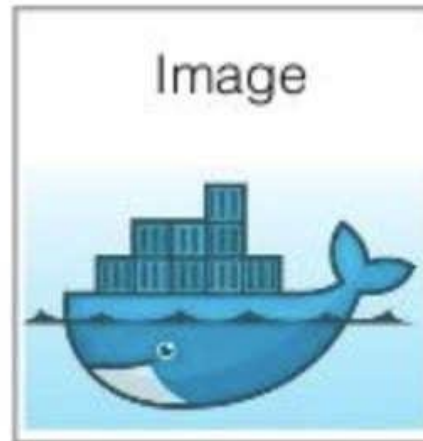
```
FROM alpine
RUN apk update && apk add nodejs
COPY . /app
WORKDIR /app
CMD ["node", "index.js"]
```



```
FROM ubuntu:16.04
MAINTAINER John Doe <john.doe@example.com>
RUN apt-get update && apt-get install -y python3
RUN python3 --help
```

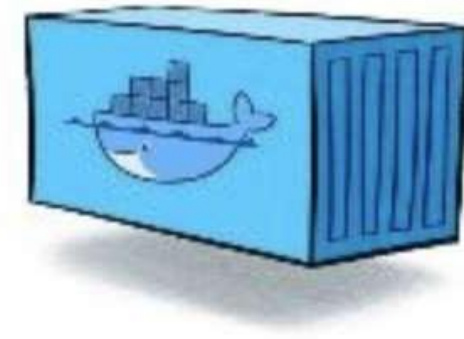
Dockerfile

build



Docker Image

run

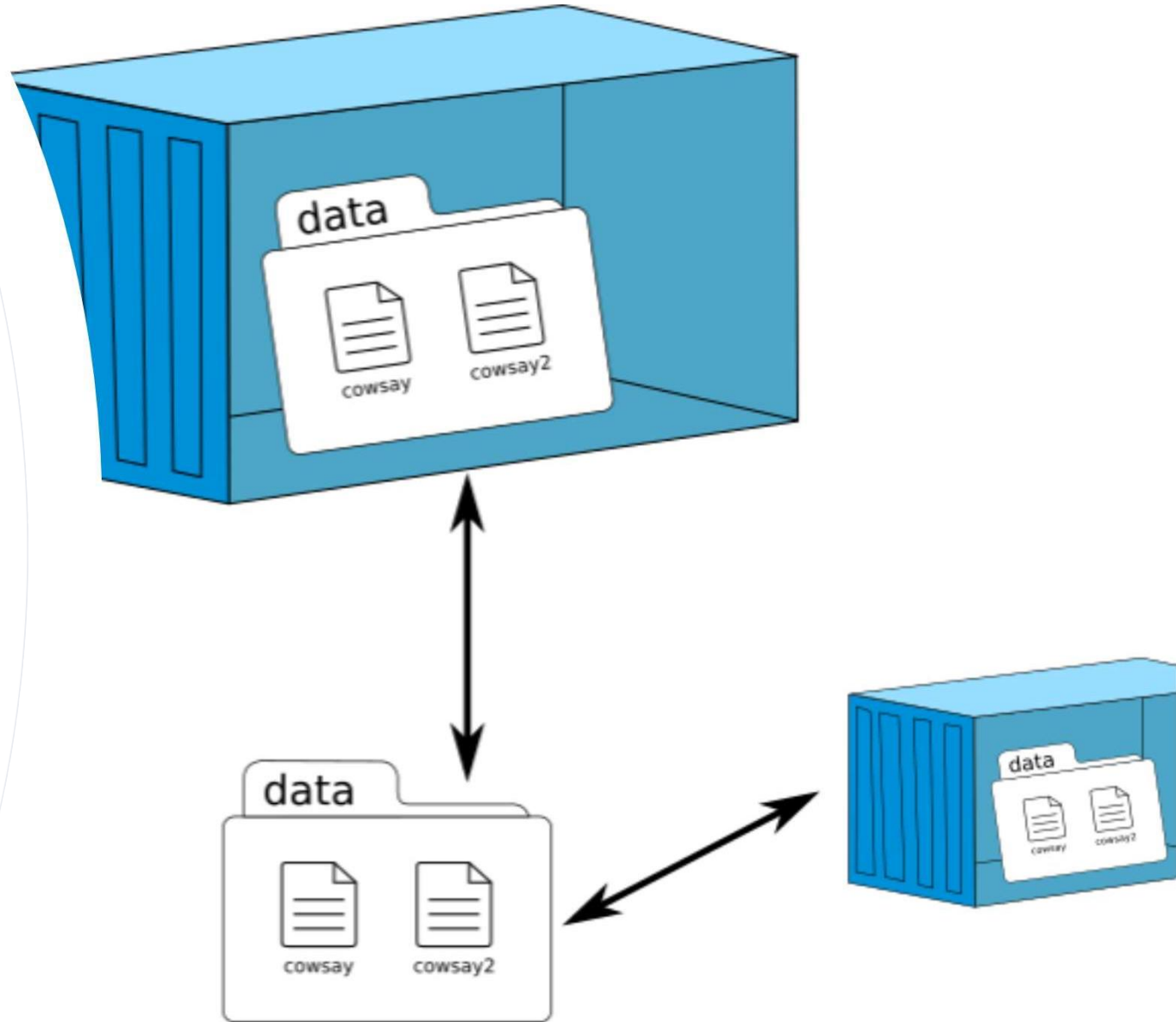


Docker Container

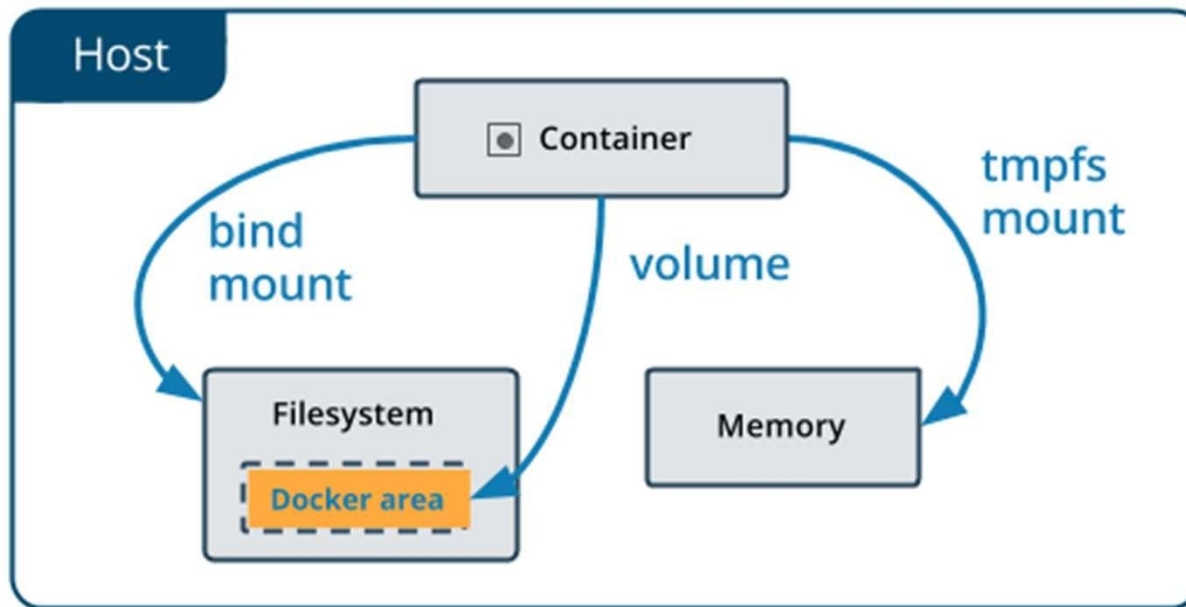
Docker build



Volumes



volume



- Les volumes sont le mécanisme préféré pour la persistance des données générées par et utilisées par les conteneurs Docker.

- Alors que les bind mount dépendent de la structure de répertoires et du système d'exploitation de la machine hôte, les volumes sont entièrement gérés par Docker.



Docker networking

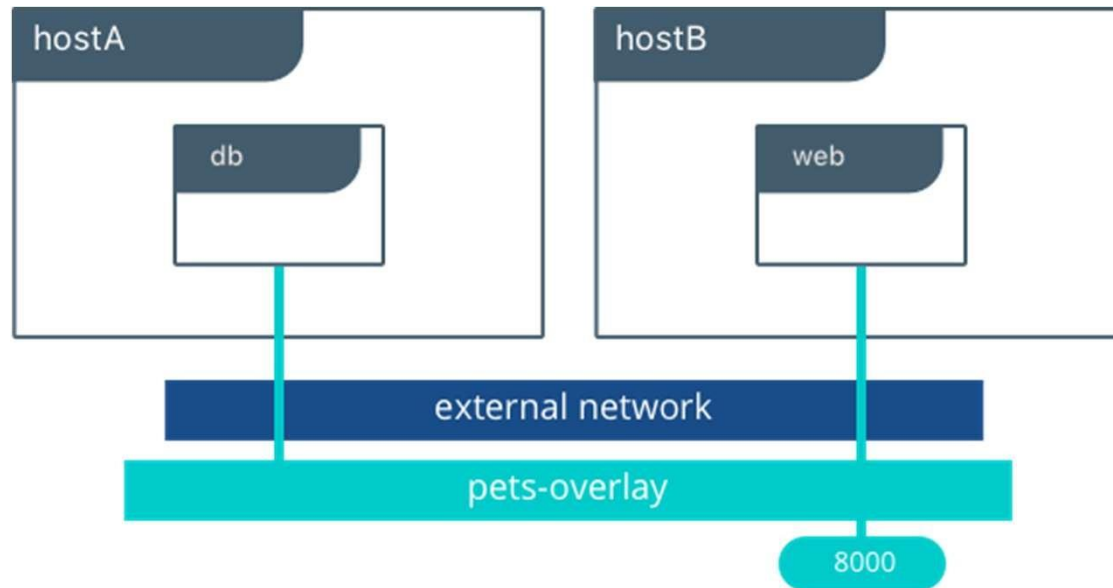
Pilotes réseau

Le sous-système réseau de Docker est enfichable, à l'aide de pilotes.

- **bridge**: Pilote réseau par défaut. Les réseaux bridge sont généralement utilisés lorsque vos applications s'exécutent dans des conteneurs autonomes qui doivent communiquer.
- **host**: Pour les conteneurs autonomes, supprimez l'isolation réseau entre le conteneur et l'hôte Docker et utilisez directement la mise en réseau de l'hôte.
- **overlay**: Les réseaux overlay connectent plusieurs nœuds Docker ensemble et permettent aux services Swarm de communiquer entre eux ou pour faciliter la communication entre un service Swarm et un conteneur autonome, ou entre deux conteneurs autonomes sur différents nœuds Docker. Cette stratégie supprime la nécessité d'effectuer un routage au niveau du système d'exploitation entre ces conteneurs.

Pilotes réseau - suite

- **macvlan**: permet d'attribuer une adresse MAC à un conteneur, le faisant apparaître comme un périphérique physique sur votre réseau. Le démon Docker achemine le trafic vers les conteneurs par leurs adresses MAC.
- **none**: Pour ce conteneur, désactivez tous les réseaux. Habituellement utilisé avec un pilote réseau personnalisé.
- **Plugins réseau** : vous pouvez installer et utiliser des plugins réseau tiers avec Docker. Ces plugins sont disponibles auprès de Docker Hub ou auprès de fournisseurs tiers.

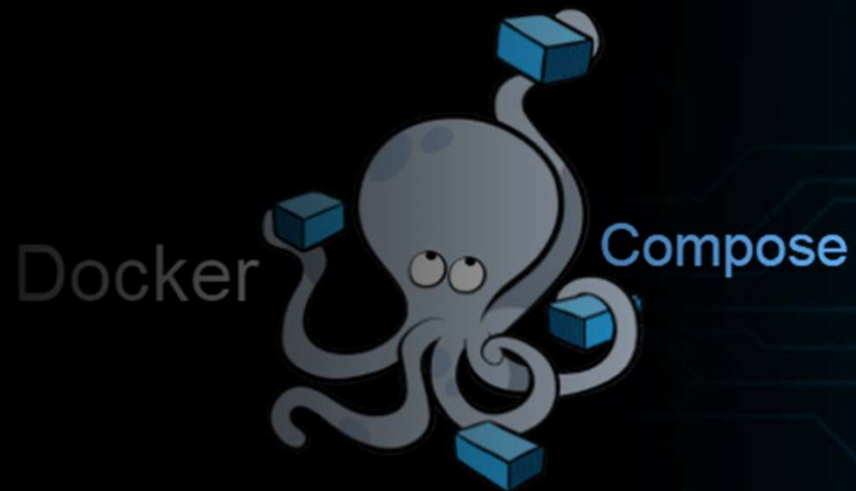


- Si vous souhaitez une mise en réseau multi-hôte native, vous devez utiliser un driver overlay.
- Il crée un réseau distribué entre plusieurs hôtes possédant le moteur Docker.
- Docker gère de manière transparente le routage de chaque paquet vers et depuis le bon hôte et le bon conteneur.

Le driver overlay



Docker compose

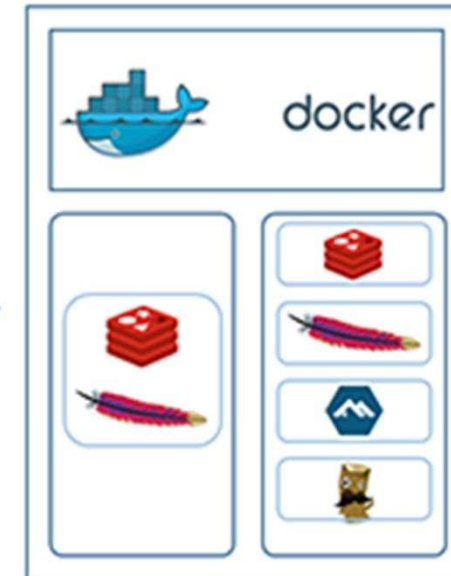


`$ docker-compose up`

docker-compose.yml

```
version: "3"
services:
  web:
    build: .
    volumes:
      - web-data:/var/www/data
  redis:
    image: redis:alpine
    ports:
      - "6379"
    networks:
      - default
```

Docker Host



Introduction

- Docker Compose est un outil permettant de définir le comportement de vos conteneurs et d'exécuter des applications Docker à conteneurs multiples.
- La config se fait à partir d'un fichier YAML, et ensuite, avec une seule commande, vous créez et démarrez tous vos conteneurs de votre configuration.

Ameur Kais

Docker compose

- Avec les applications à plusieurs niveaux (tiers), Dockerfile et les commandes d'exécution deviennent de plus en plus complexes.
- Docker Compose est un outil pour rationaliser la définition et l'instanciation des applications Docker multi-niveaux et multi-conteneurs.
- Compose nécessite un seul fichier de configuration et une seule commande pour organiser et augmenter le niveau d'application.
- Docker Compose simplifie la conteneurisation d'une application à plusieurs niveaux et à plusieurs conteneurs, qui peuvent être assemblées à l'aide du fichier de configuration **docker-compose.yml** et de la commande **docker-compose** pour fournir un service d'application unique.
- Le fichier Compose permet de documenter et de configurer toutes les **dépendances** de service de l'application (bases de données, files d'attente, caches, API de service Web, etc.)

Docker compose - suite

Docker Compose définit et exécute des services complexes:

- définit des conteneurs uniques via Dockerfile
- décrit une application multi-conteneurs via un fichier de configuration unique (`docker-compose.yml`)
- gère la pile d'applications via un seul binaire (`docker-compose up`)
- lie des services via Service Discovery

Docker-compose.yml

- Le fichier de configuration Docker Compose spécifie les services, les réseaux et les volumes à exécuter:
 - **services** - l'équivalent de passer des paramètres de ligne de commande à l'exécution de docker
 - **networks** - analogues aux définitions du réseau docker créer
 - **volumes** - analogues aux définitions du volume docker create

YAML

- Le fichier de configuration Compose est au format déclaratif YAML :
- **Y** **A** **M** **L** **A** **i**n't **M** **a**rku**p** **L** **a**ngu**a**ge (YAML)
- La philosophie de YAML est que «lorsque les données sont faciles à visualiser et à comprendre, la programmation devient une tâche plus simple»
- Convivial et compatible avec les langages de programmation modernes pour les tâches courantes
- Structure minimale pour un maximum de données:
 - L'indentation peut être utilisée pour la structure
 - Les deux-points (:) séparent les paires de clé : valeurs
 - Les tirets sont utilisés pour créer des listes à puces

Exemple

```
version: "3"
services:
  web:
    build: .
    volumes:
      - web-data:/var/www/data
  redis:
    image: redis:alpine
    ports:
      - "6379"
    networks:
      - default
```



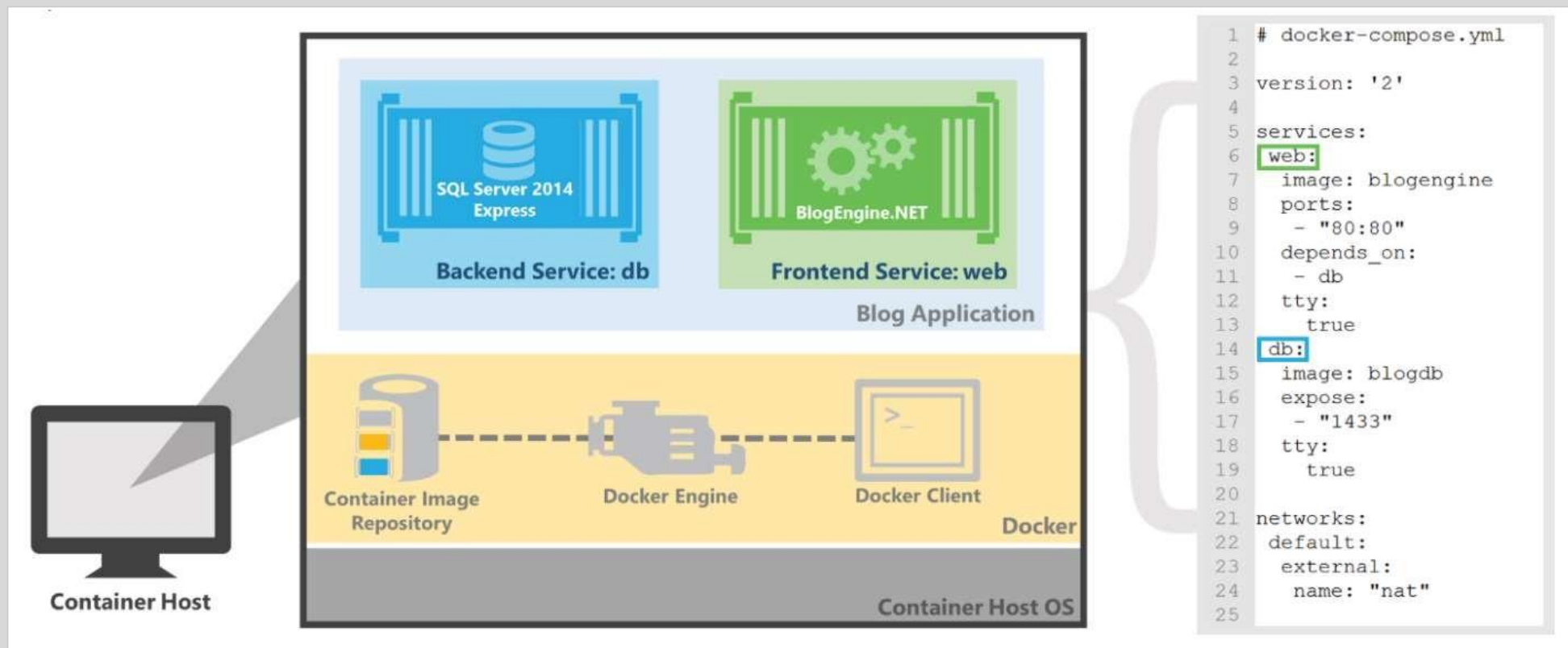
```
web:
  build: ./web
  ports:
    - "3000:3000"
  volumes:
    - ./code/web:/srv/www/
  command: node /srv/www/server.js
  links:
    - api
    - redis
    - mongodb
    - elasticsearch
```

**these links
point to these
services**

```
api:
  build: ./api
  ports:
    - "3030:3030"
  volumes:
    - ./code/api:/srv/www/
  command: node /srv/www/server.js
```

```
redis:
  image: redis
```

Exemple fichier Docker-compose – Blog App



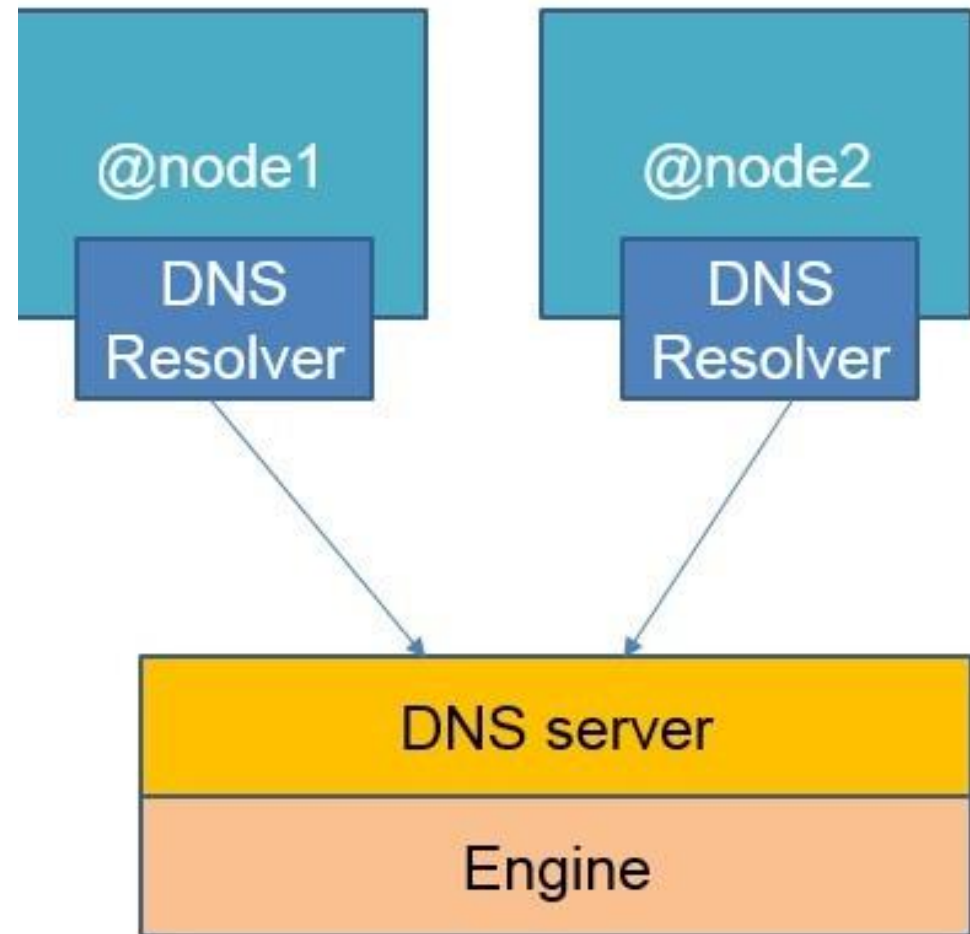


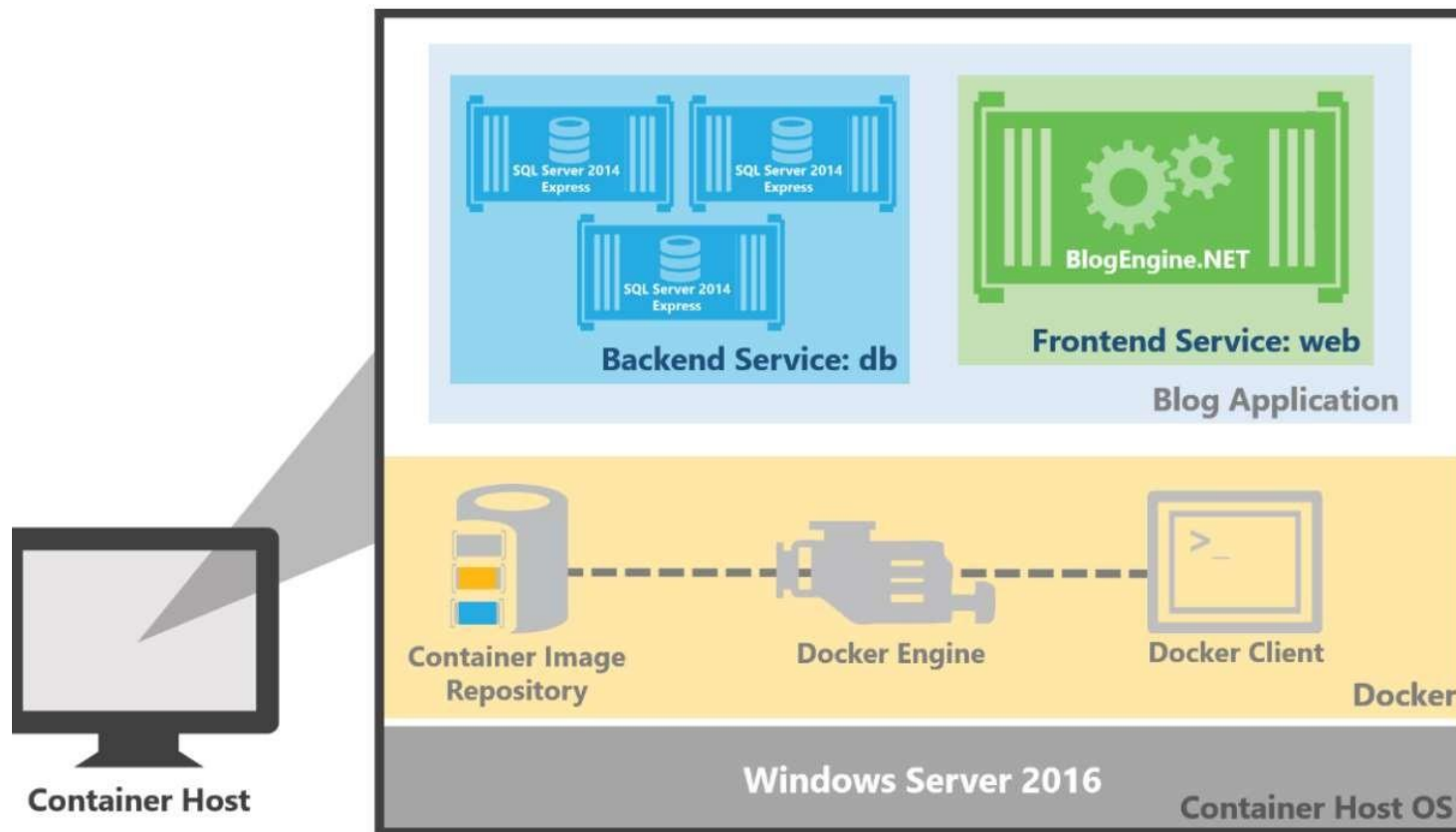
Service Discovery

- Service Discovery en lui-même est une exigence clé pour faire évoluer les applications multiservices à l'aide de l'équilibrage de charge (scale-out) basé sur DNS.
- La découverte de service (Service Discovery) est intégrée à Docker, et offre deux avantages clés:
- l'enregistrement du service et le mappage du nom du service vers l'IP (DNS).
- La découverte de services est particulièrement utile dans le contexte d'applications évolutives, car elle permet de découvrir et de référencer les services multi-conteneurs de la même manière que les services à conteneur unique;
- avec Service Discovery, la communication intra-application est simple et concise: n'importe quel service peut être référencé par son nom, quel que soit le nombre d'instances de conteneur utilisées pour exécuter ce service.

Service Discovery

- Service fourni par du DNS embarqué
- Hautement disponible (HA)
- Utilise le Network Control Plane pour identifier les états
- Peut être utilisé pour découvrir les services et les conteneurs





```
:> docker-compose scale web=1 db=3
```

Scale-out
avec Docker-
compose

Docker Swarm

- Solution native (officielle) de Docker pour faire du clustering
- Transformer un ensemble d'hôtes Docker en un unique hôte virtuel Docker.
- Utilise la même API
- Tous les outils qui communiquent avec un daemon Docker peuvent utiliser Swarm (ex: Dokku, Compose, Machine, Jenkins et bien sur le client Docker).
- Alternatives : Shipyard, Mesos, Kubernetes, Rancher,
...