



Conteneurisation et Docker

Un cours de Yann Fornier



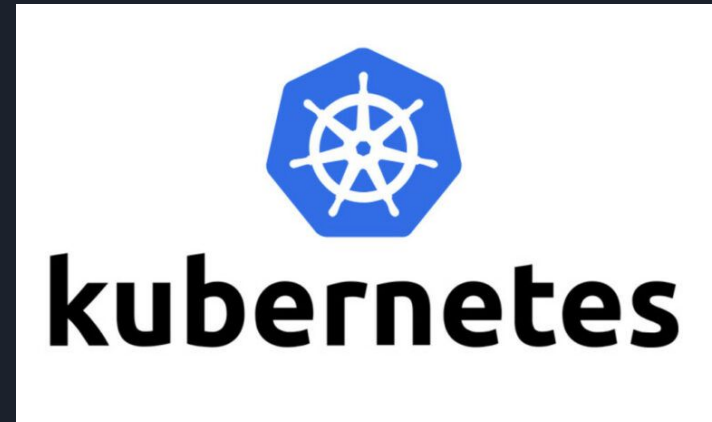
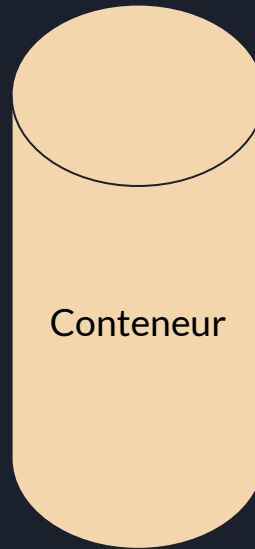
Conteneurisation et Docker

Session 7: Conteneurisation et docker

- Introduction à la conteneurisation
- Présentation & utilisation de Docker

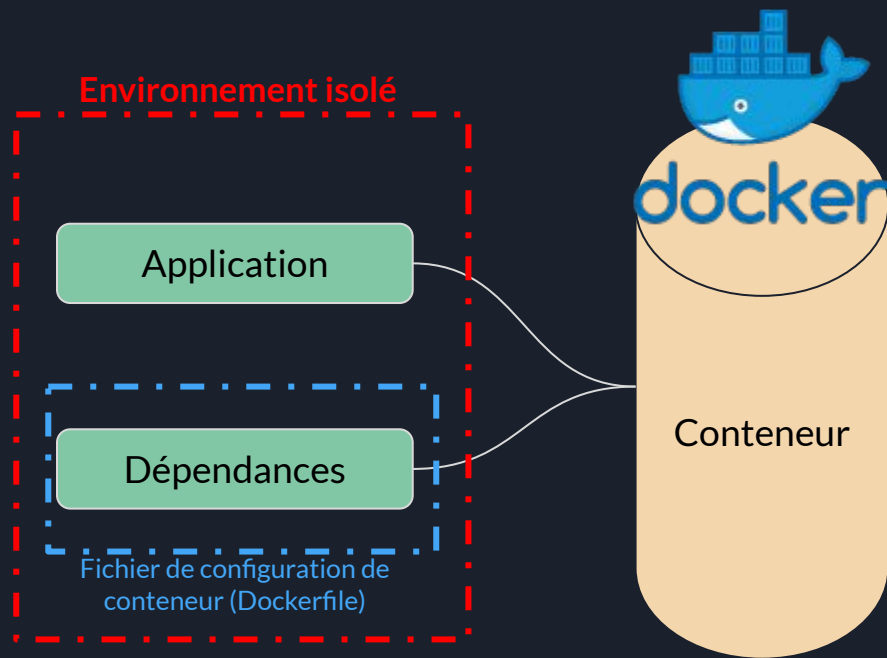


Qu'est ce qu'un conteneur ?



Introduction à la conteneurisation

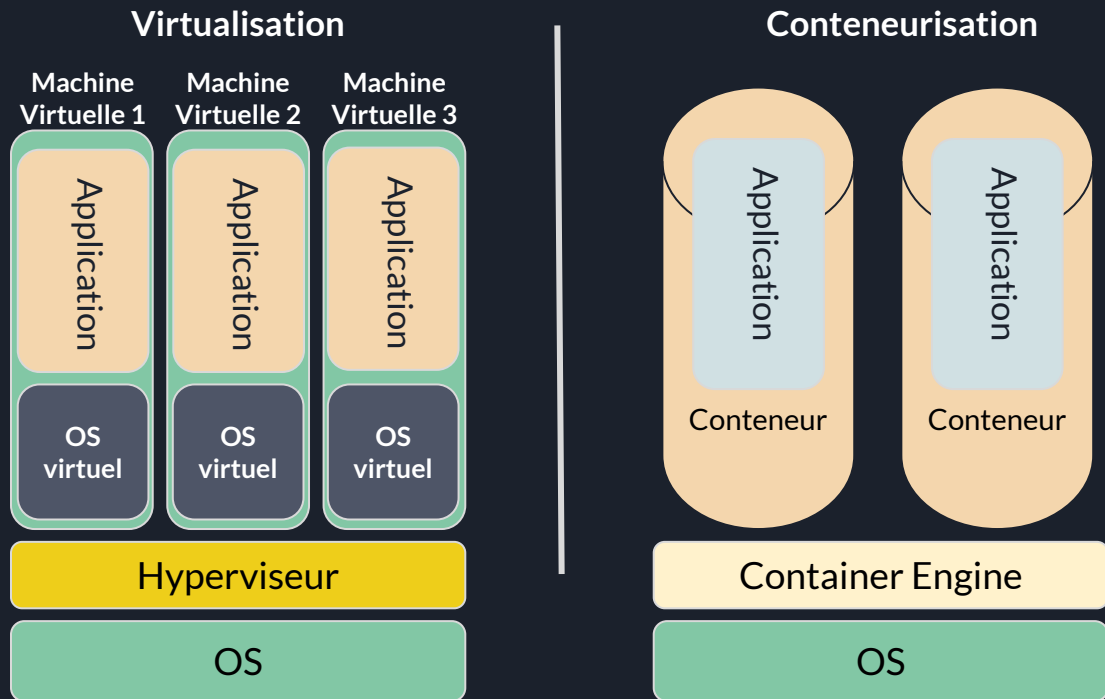
Qu'est ce qu'un conteneur ?



Un conteneur est un environnement logiciel qui permet de lancer et exécuter des applications de manière isolée et autonome. Il encapsule les applications, leurs dépendances et leur configuration dans un environnement qui peut être facilement déplacé d'un système à un autre.

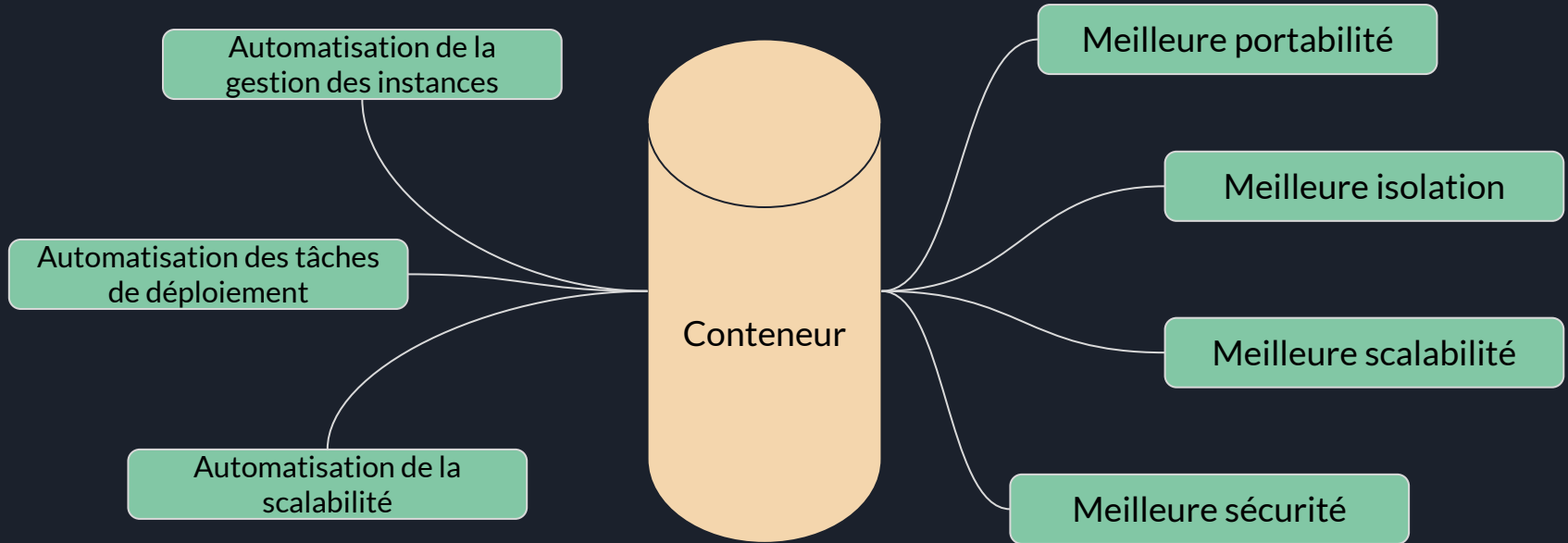
Introduction à la conteneurisation

Qu'est ce qu'un conteneur ?

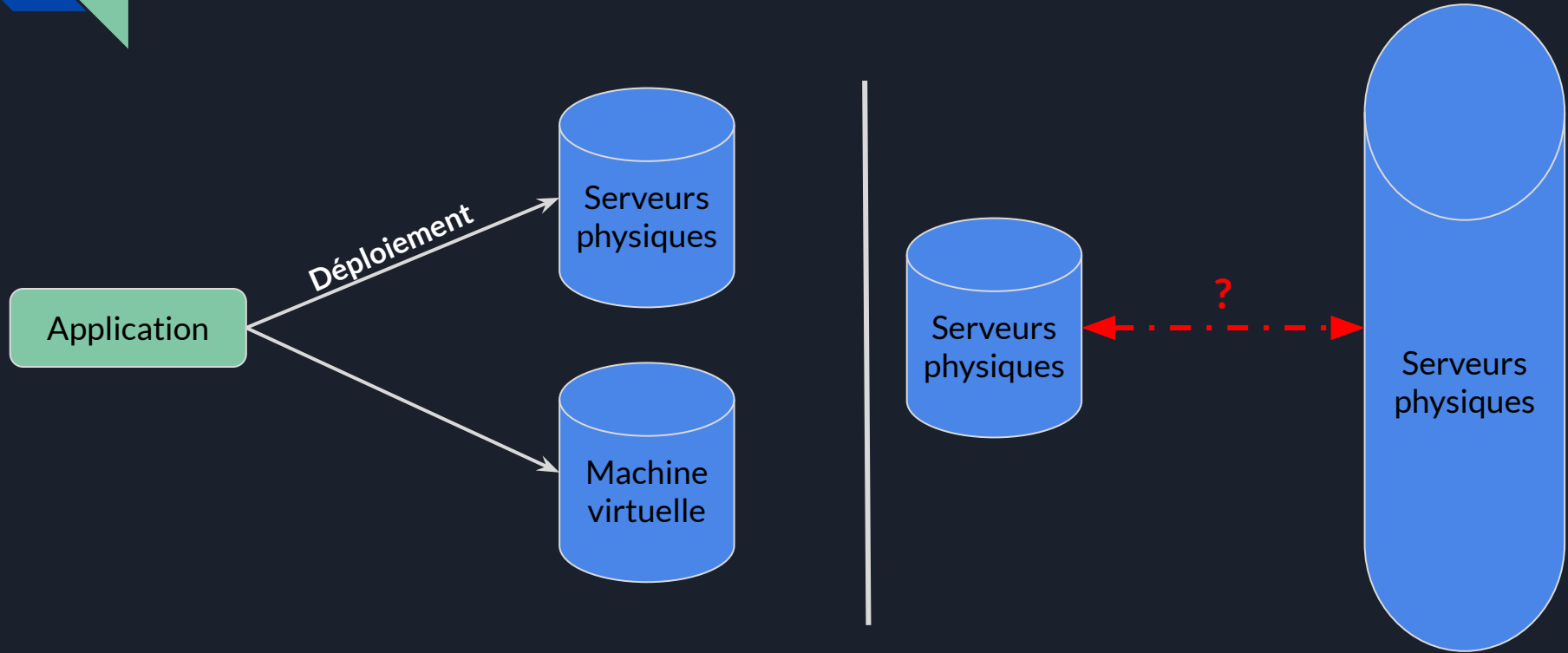


Les conteneurs sont similaires aux machines virtuelles, mais contrairement à ces dernières, ils ne nécessitent pas de système d'exploitation virtuel, ils partagent donc le système d'exploitation hôte, ce qui les rend plus légers et plus rapides à démarrer.

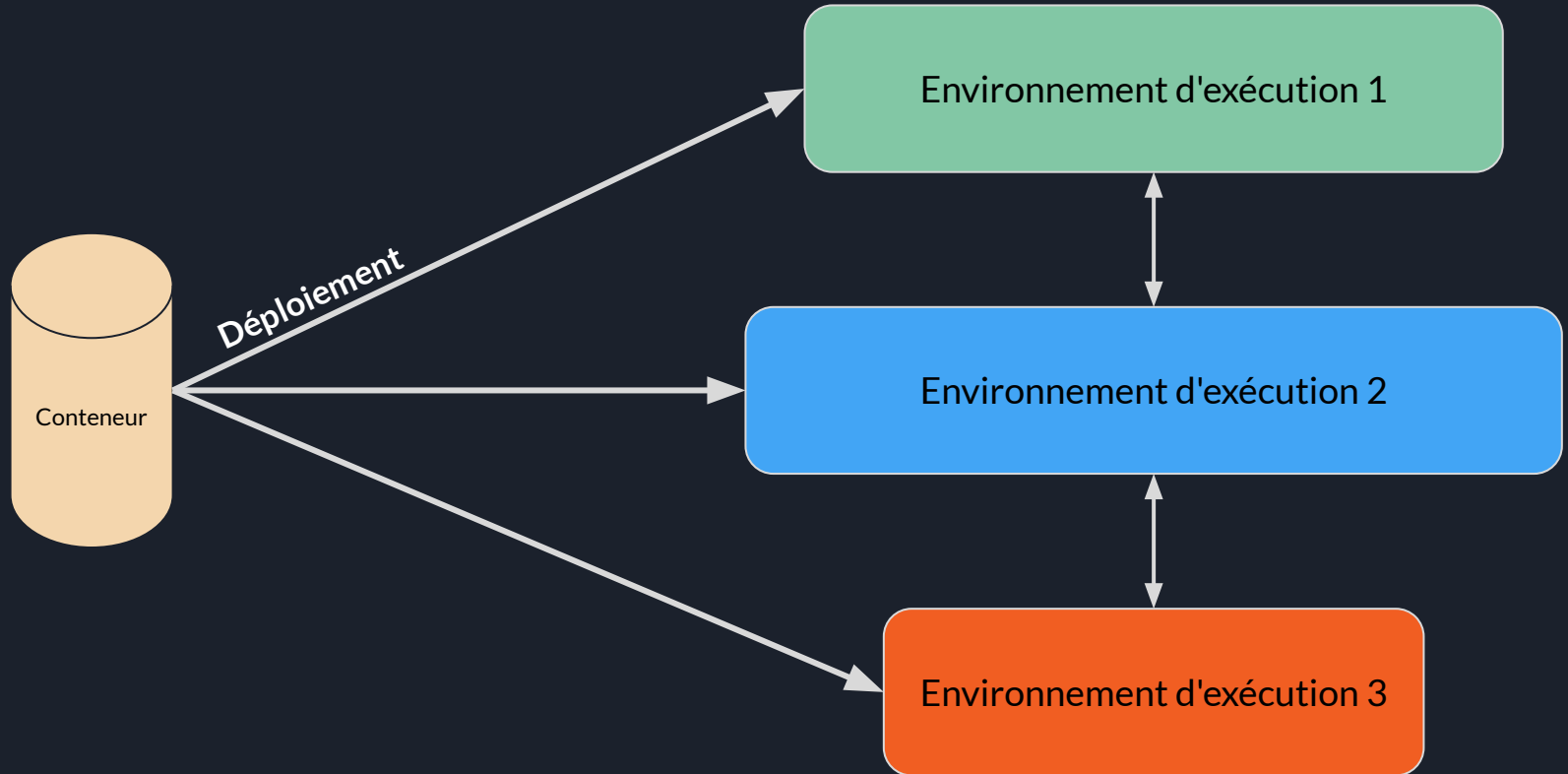
Qu'est ce qu'un conteneur ?



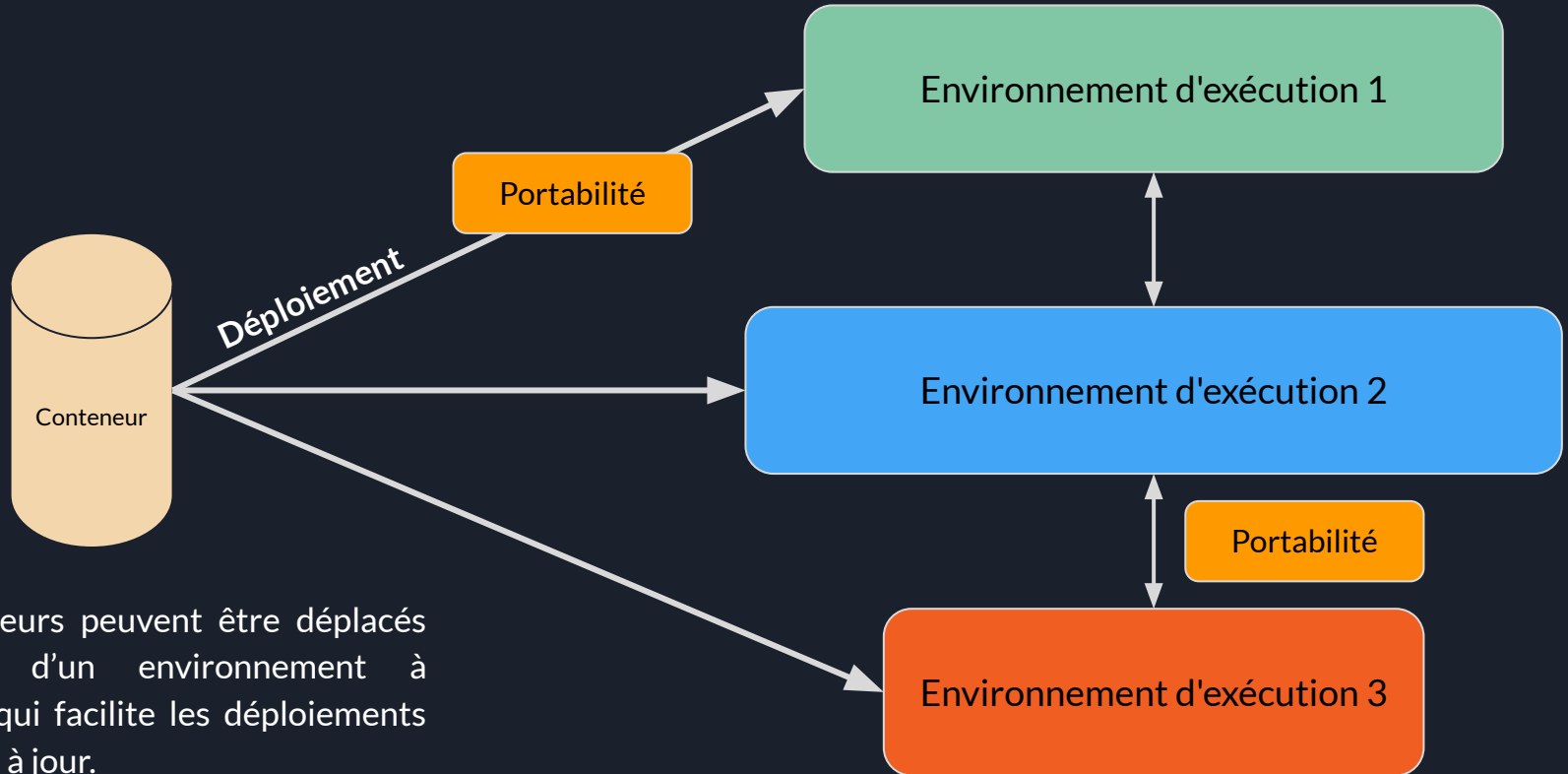
Les défis de l'environnement d'exécution des applications



Les avantages de la conteneurisation

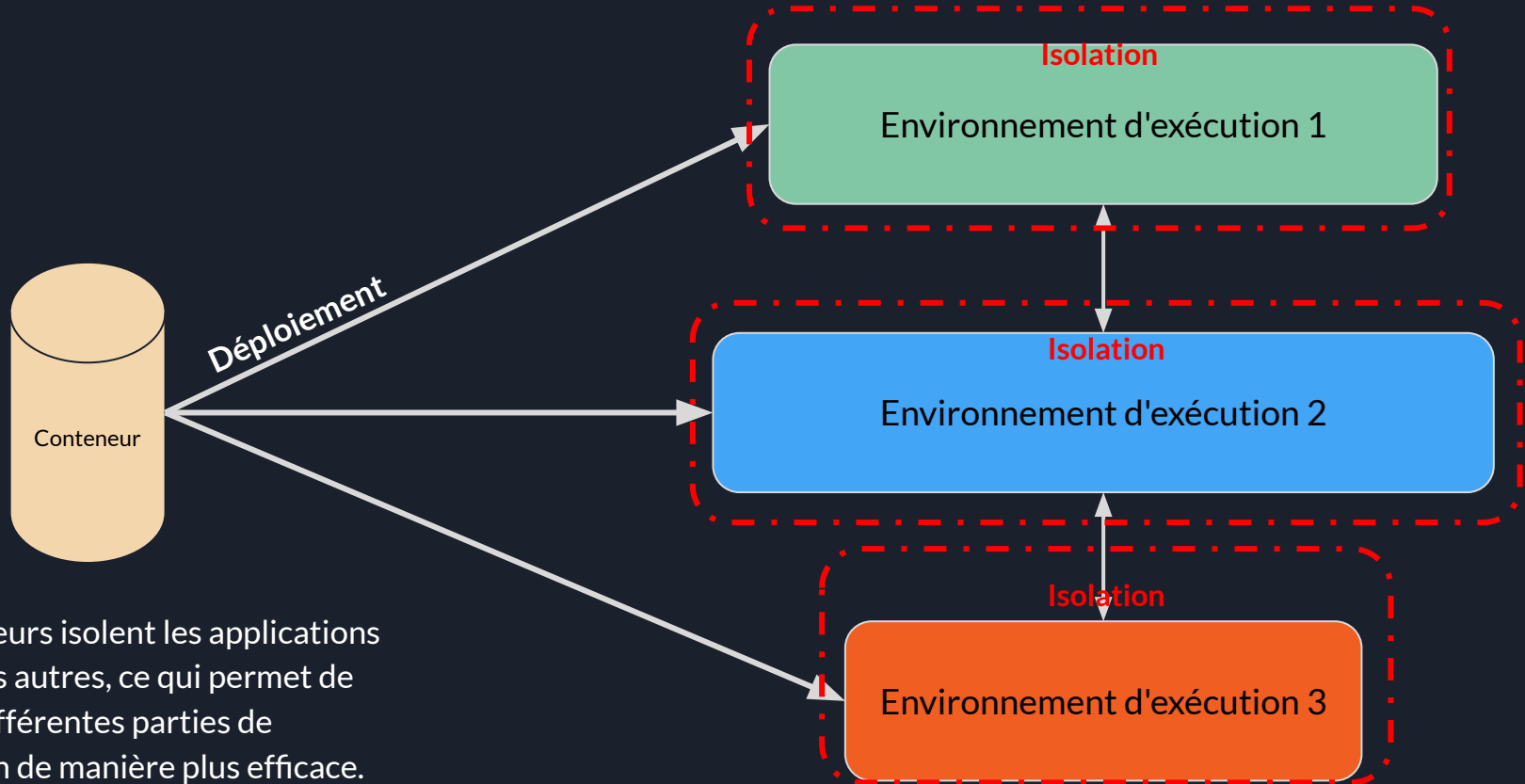


Les avantages de la conteneurisation



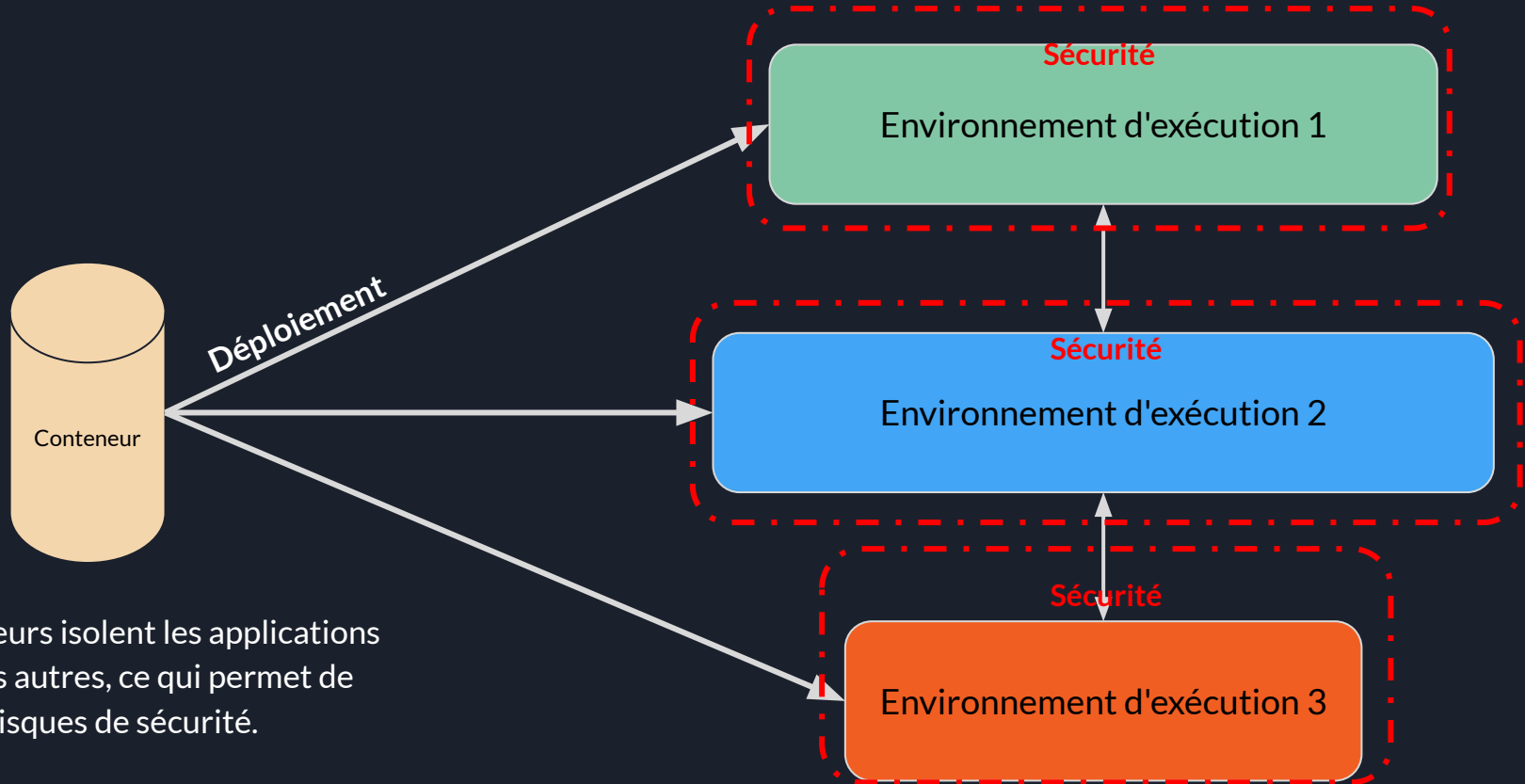
Les conteneurs peuvent être déplacés facilement d'un environnement à l'autre, ce qui facilite les déploiements et les mises à jour.

Les avantages de la conteneurisation



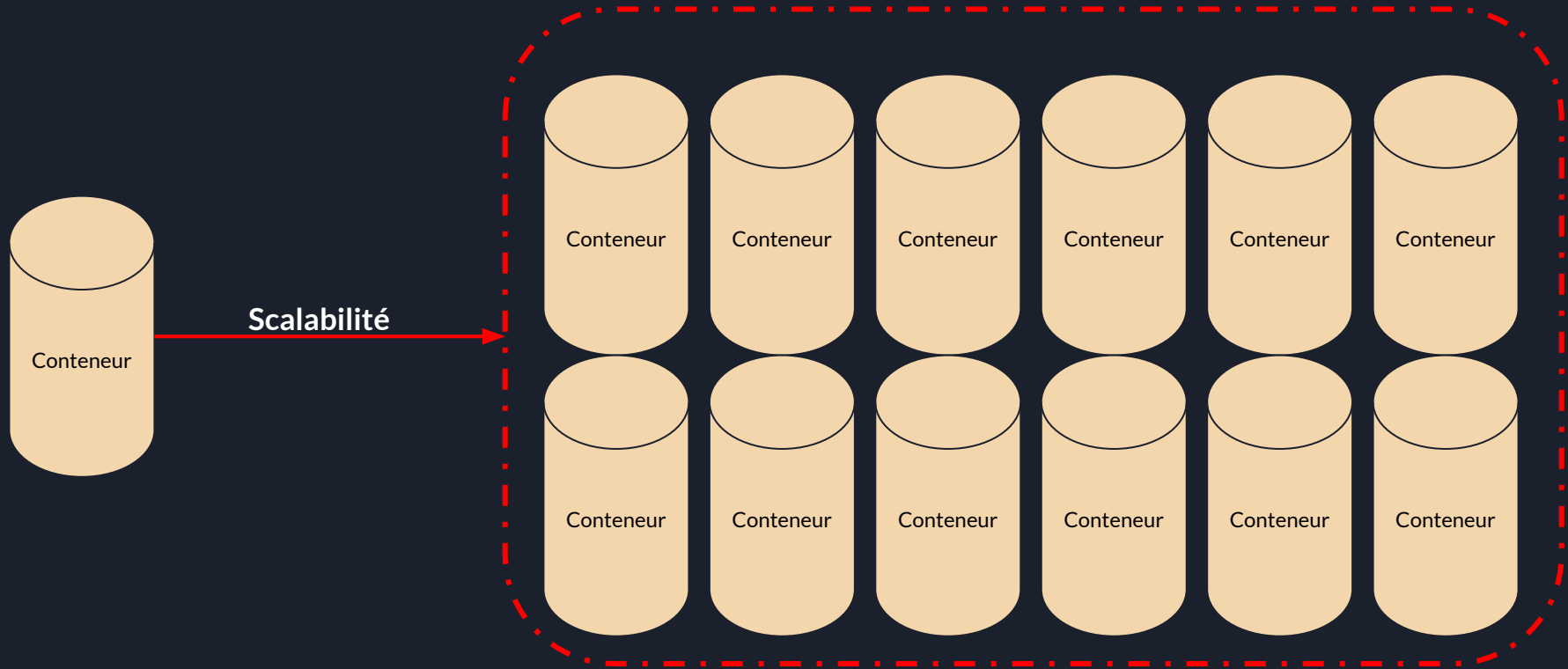
Les conteneurs isolent les applications les uns des autres, ce qui permet de gérer les différentes parties de l'application de manière plus efficace.

Les avantages de la conteneurisation



Les conteneurs isolent les applications les unes des autres, ce qui permet de limiter les risques de sécurité.

Les avantages de la conteneurisation





Conclusion

En résumé, la conteneurisation est une technique de “virtualisation légère” qui permet d’isoler les applications et leurs dépendances dans des environnements autonomes appelés conteneurs.

Cela permet de garantir que l’application fonctionne de manière identique sur tous les environnements, qu’il s’agisse d’un environnement de développement, de test ou de production. Il offre des avantages tels que la portabilité, l’isolation, la scalabilité et la sécurité.

Il est souvent comparé aux machines virtuelles qui ont besoin de plus de ressources et peuvent être plus lents à démarrer, tandis que les conteneurs sont plus légers et plus rapides, ils sont donc plus adaptés pour les applications qui nécessitent une scalabilité et une flexibilité élevées.



Les concepts clés de la conteneurisation : images, conteneurs, registres

La conteneurisation repose sur trois concepts clés : les images, les conteneurs et les registres.



Images



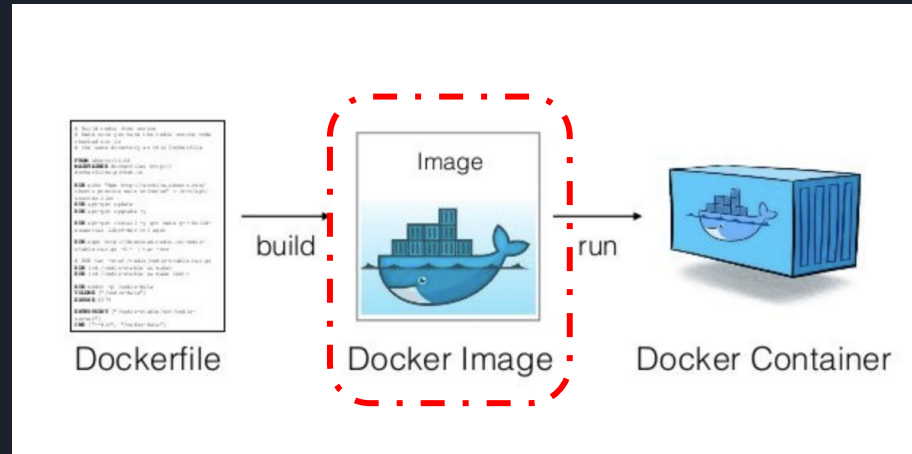
Conteneurs



Registres

Les concepts clés de la conteneurisation : images, conteneurs, registres

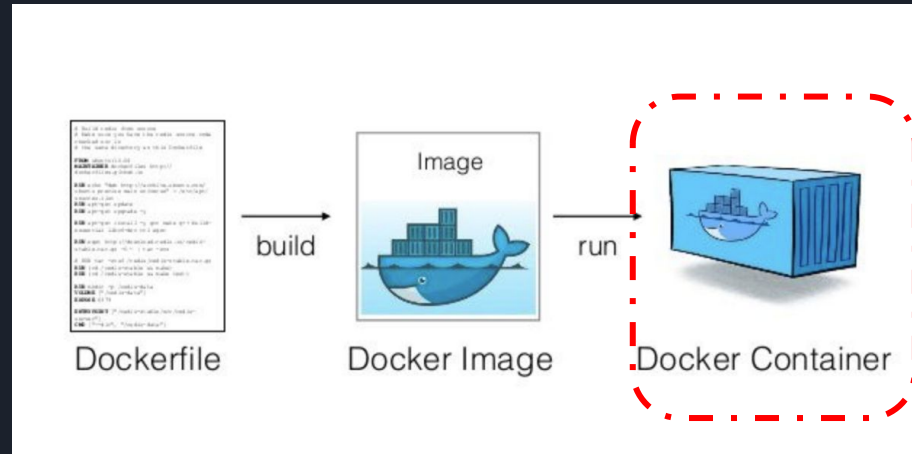
Une image est un package qui contient une application et toutes ses dépendances, ainsi que les instructions nécessaires pour créer un conteneur à partir de l'image.



Les concepts clés de la conteneurisation : images, conteneurs, registres

Un conteneur est une instance en cours d'exécution d'une image. Les conteneurs sont légers et portables, et peuvent être déployés rapidement et facilement.

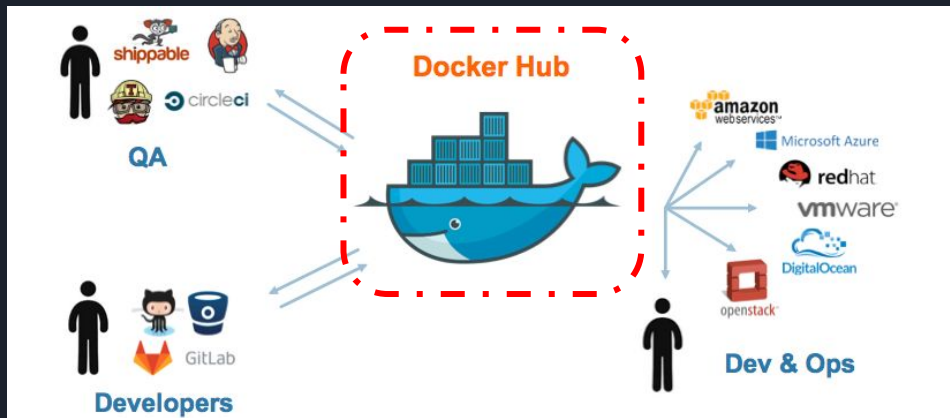
Conteneurs



Les concepts clés de la conteneurisation : images, conteneurs, registres

Un registre est un dépôt centralisé pour stocker et distribuer des images Docker. Le registre public de Docker, Docker Hub, contient des milliers d'images pré-construites que les développeurs peuvent utiliser pour créer des conteneurs.

Registres



Access the world's largest library of container images



busybox
Official
↓ 1B+



ubuntu
Official
↓ 1B+



python
Official
↓ 1B+



postgres
Official
↓ 1B+



memcached
Official
↓ 1B+



httpd
Official
↓ 1B+



mysql
Official
↓ 1B+



traefik
Official
↓ 1B+



rabbitmq
Official
↓ 1B+



docker
Official
↓ 1B+

[See all Docker Official Images](https://hub.docker.com/)

<https://hub.docker.com/>

Introduction à Docker

Docker est la technologie de conteneurisation la plus populaire et la plus largement utilisée.



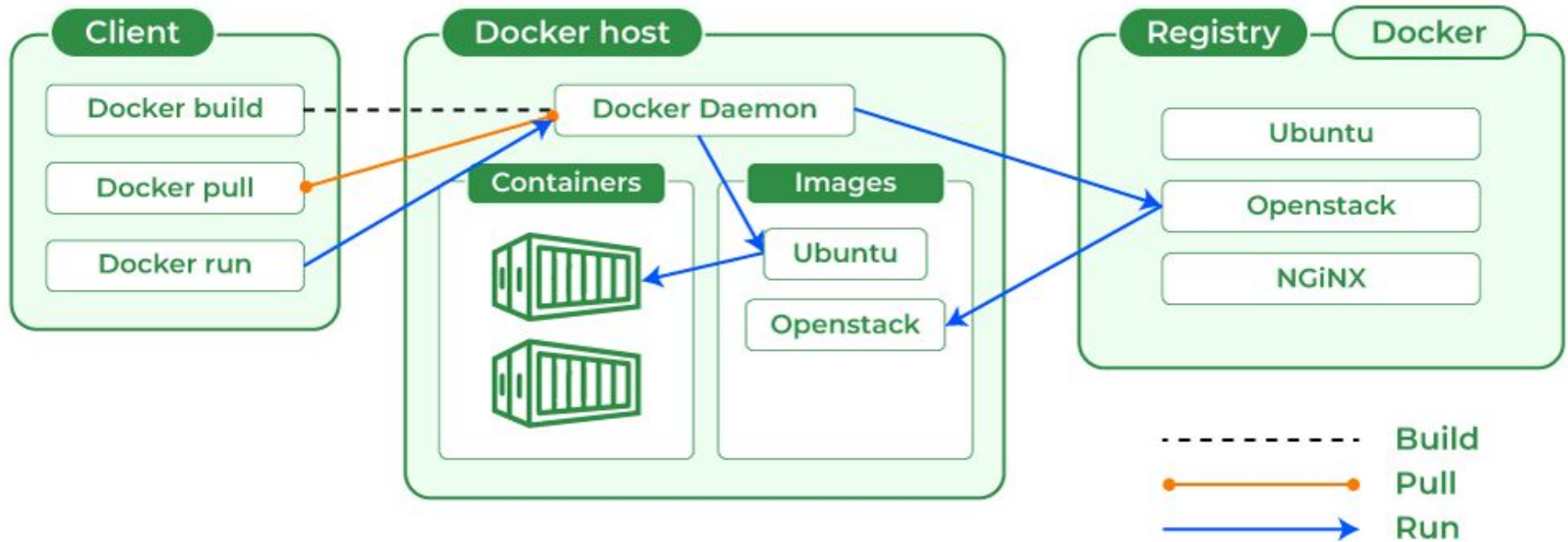
Histoire de Docker

Docker a été créé en 2013 par Solomon Hykes et est rapidement devenu un outil incontournable dans l'écosystème des applications web. La popularité de Docker est due en partie à sa facilité d'utilisation et à sa portabilité.

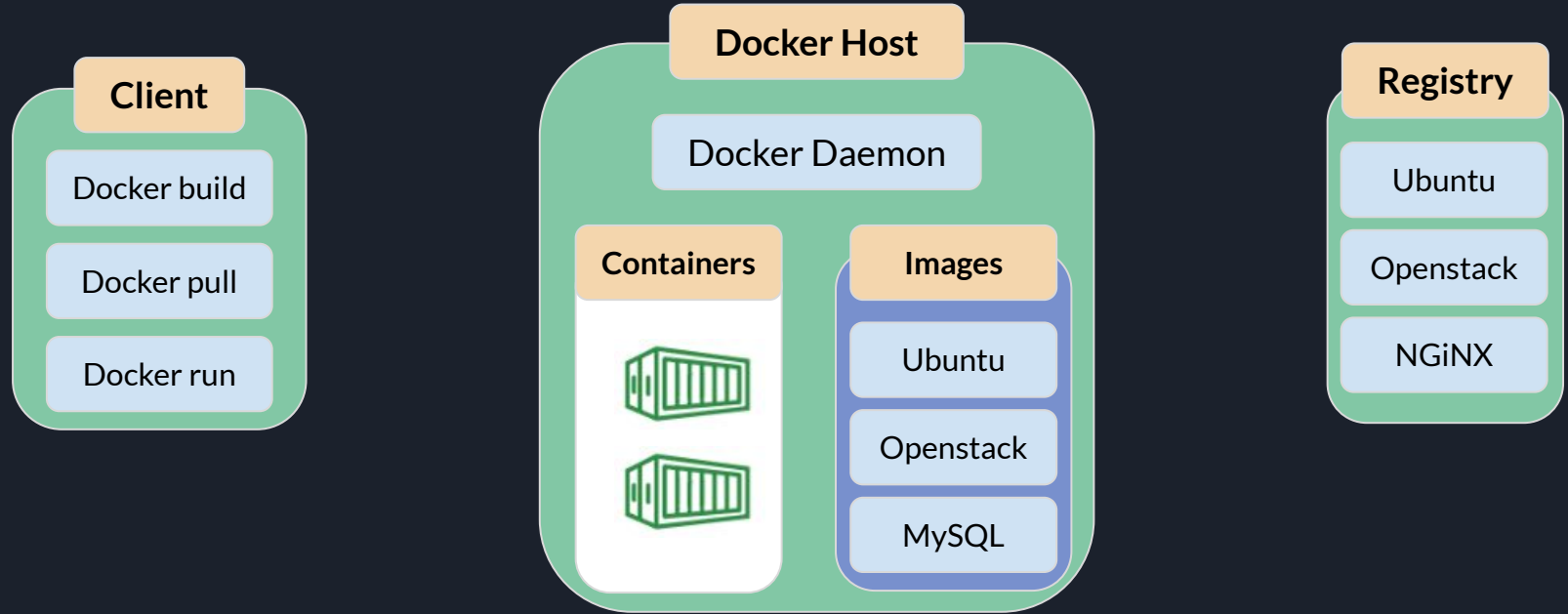


Solomon Hykes

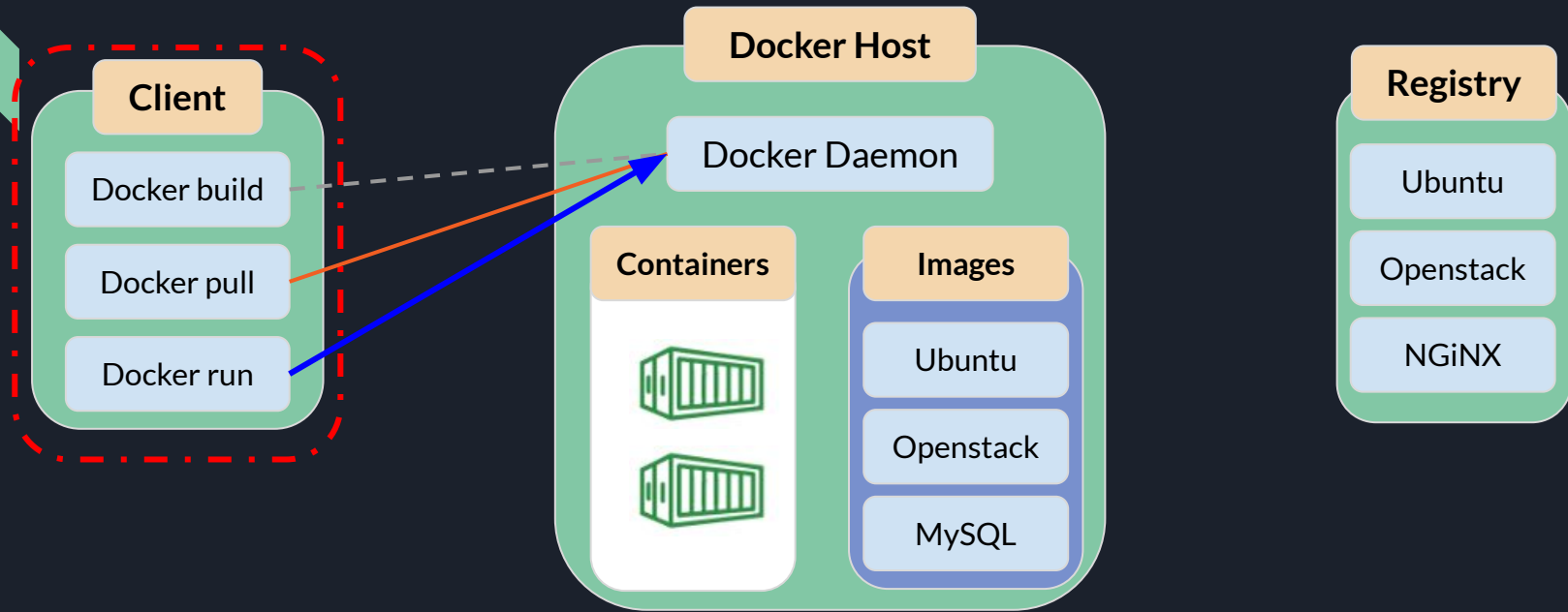
Les composants de Docker



Les composants de Docker

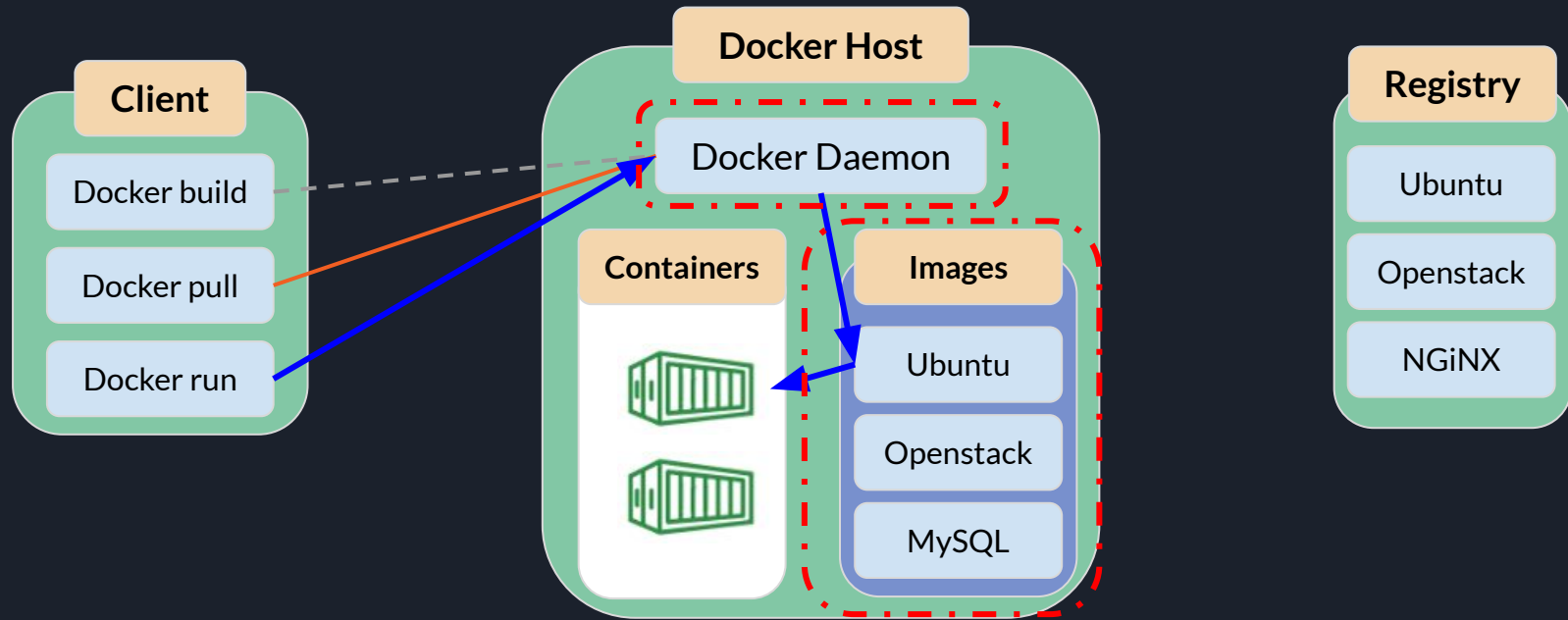


Les composants de Docker



Le client Docker est la principale interface permettant de communiquer avec le système Docker. il reçoit les commandes par la CLI et les transmet au Docker Daemon.

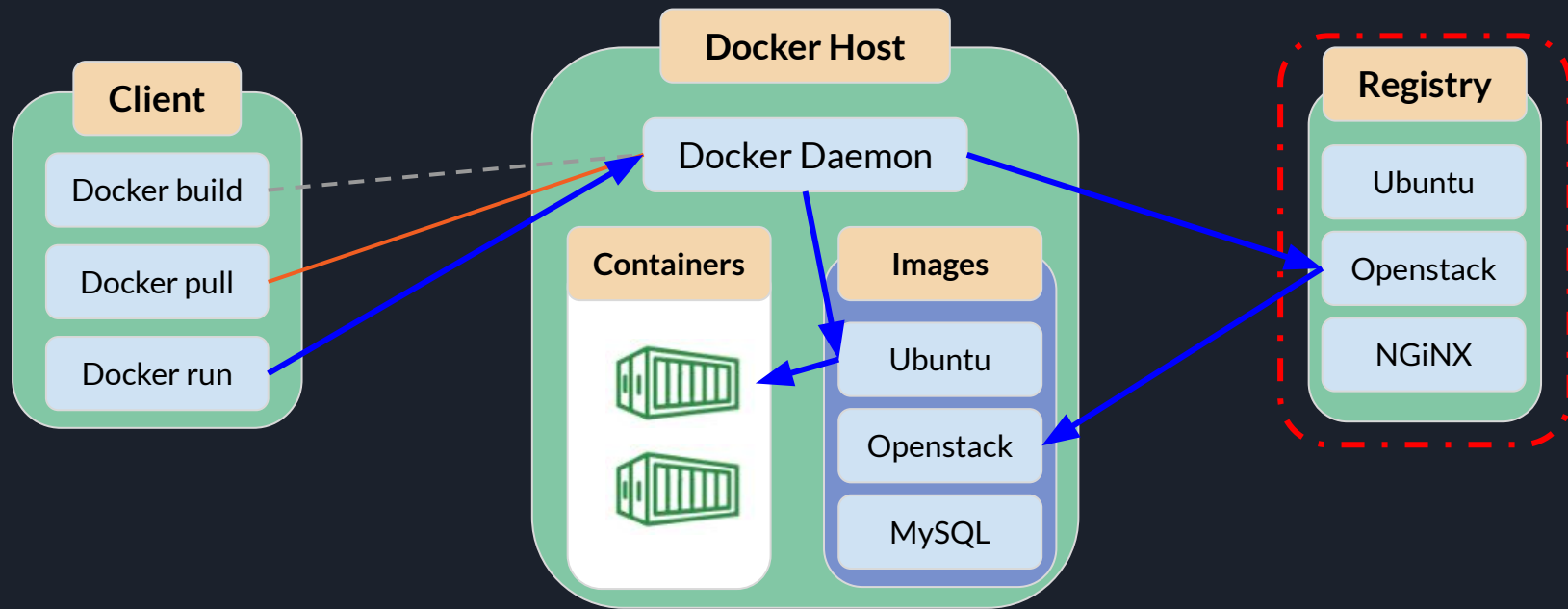
Les composants de Docker



Le Docker daemon traite les requêtes API afin de gérer les différents aspects de l'installation comme les images, les conteneurs ou les volumes de stockage.

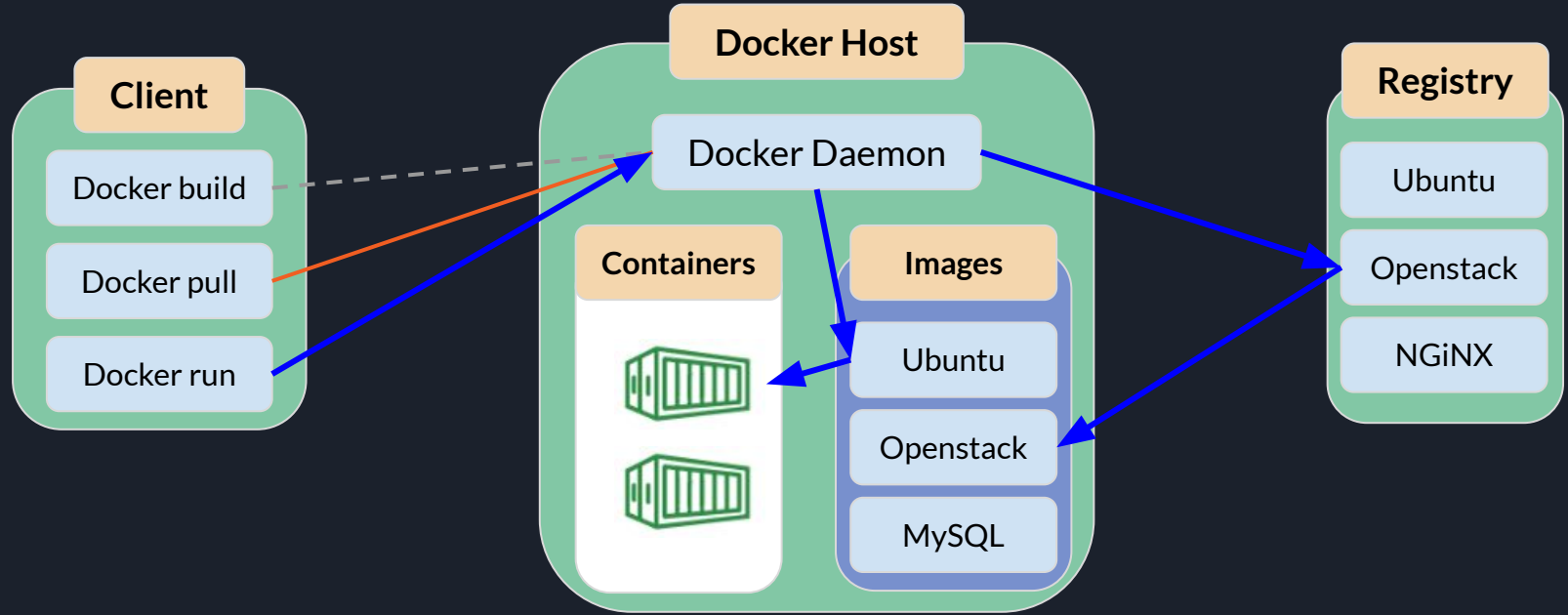
Les images Docker sont des modèles en lecture seule, utilisées pour créer des conteneurs Docker. Elles sont composées de plusieurs couches empaquetant toutes les installations, dépendances, bibliothèques, processus et codes d'application nécessaires.

Les composants de Docker



Le registre Docker est un système de catalogage permettant l'hébergement et le "push and pull" des images Docker. Vous pouvez utiliser votre propre registre local ou l'un des nombreux services hébergés par des tiers comme Amazon ECR ou Google Container Registry.

Les composants de Docker





Les avantages de Docker

Docker offre plusieurs avantages par rapport aux méthodes de déploiement traditionnelles. Tout d'abord, il permet de créer des conteneurs légers et portables qui peuvent être déployés rapidement et facilement sur différentes plateformes. De plus, Docker permet d'isoler les applications et leurs dépendances, ce qui facilite la gestion et le déploiement des applications. Enfin, Docker offre une grande flexibilité et facilite le travail en équipe en permettant aux développeurs de travailler sur des applications de manière indépendante et de partager leurs images et leurs conteneurs avec d'autres développeurs.



Travailler avec des conteneurs Docker

Une fois que vous avez une compréhension de base de Docker, vous pouvez commencer à travailler avec des conteneurs Docker. Cette section explique comment créer des images Docker et exécuter des conteneurs.

Attention vous allez avoir besoin d'installer docker sur votre machine pour faire les commandes de cette partie du cours.



Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
WORKDIR /app
COPY requirements.txt
RUN pip install - --no-cache-dir -r requirements.txt
COPY ..
CMD ["python", "app.py"]
```

Dockerfile crée une image qui exécute une application web Python

Dockerfile
(.dockerfile)

Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster  
WORKDIR /app  
COPY requirements.txt  
RUN pip install - --no-cache-dir -r requirements.txt  
COPY .  
CMD ["python", "app.py"]
```

FROM spécifie l'image de base que vous allez utiliser (celle présente sur DockerHub)

Dockerfile crée une image qui exécute une application web Python

Dockerfile
(.dockerfile)

Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
WORKDIR /app
COPY requirements.txt
RUN pip install - --no-cache-dir -r requirements.txt
COPY .
CMD ["python", "app.py"]
```

WORKDIR permet de spécifier le répertoire dans lequel seront basées les instructions qui suivront son appel

Dockerfile crée une image qui exécute une application web Python

Dockerfile
(.dockerfile)

Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - -no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

COPY permet d'importer des documents locaux mais pas ceux provenant d'une URL. (contrairement à ADD)

Dockerfile crée une image qui exécute une application web Python

Dockerfile
(.dockerfile)

Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - --no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

RUN permet d'exécuter des commandes supplémentaires à l'intérieur du build du dockerfile

Dockerfile crée une image qui exécute une application web Python

Dockerfile
(.dockerfile)

Créer une image Docker

Pour créer une image Docker, vous devez écrire un Dockerfile, qui est un fichier texte qui contient toutes les instructions pour créer une image.

```
FROM python:3.8-slim-buster
```

```
WORKDIR /app
```

```
COPY requirements.txt
```

```
RUN pip install - --no-cache-dir -r requirements.txt
```

```
COPY ..
```

```
CMD ["python", "app.py"]
```

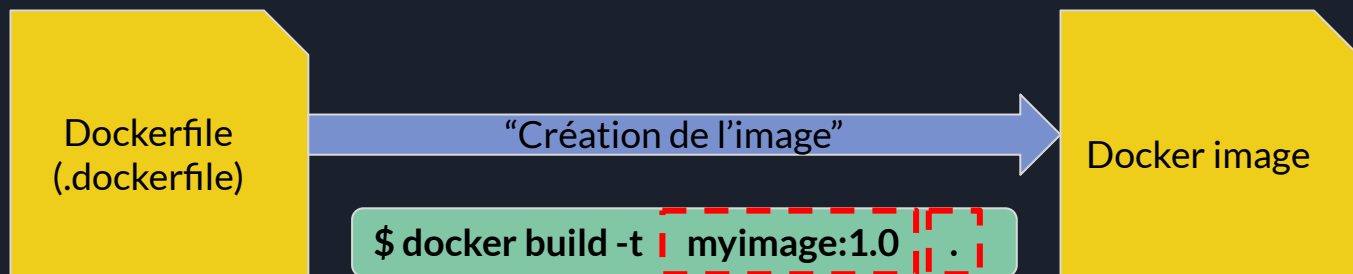
CMD sert à mentionner les instructions qui seront exécutées en premier lors du lancement du conteneur. Permet une exécution sans paramètre supplémentaire

Dockerfile crée une image qui exécute une application web Python

Dockerfile
(.dockerfile)

Créer une image Docker

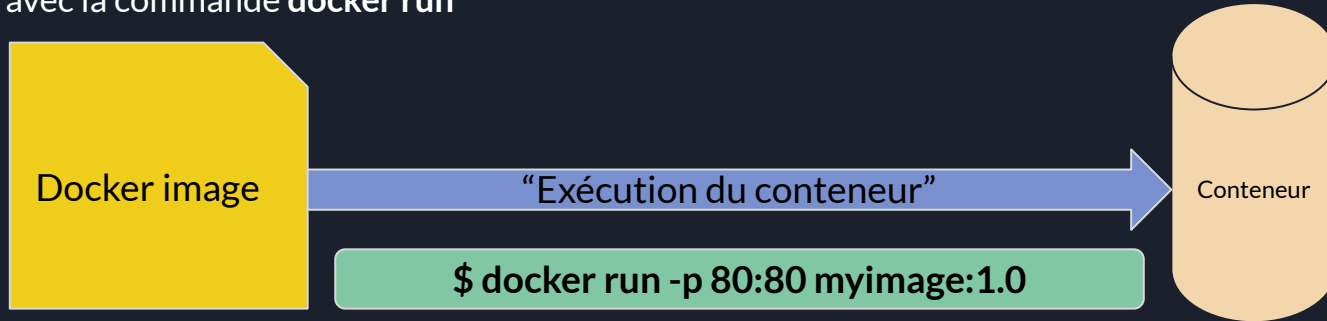
Une fois que vous avez écrit le Dockerfile, vous pouvez créer l'image avec la commande **docker build**. Par exemple :




Cette commande crée une image avec le tag `myimage:1.0` à partir du Dockerfile situé dans le répertoire courant `(.)`.

Exécuter un conteneur Docker

Une fois que vous avez créé une image Docker, vous pouvez l'exécuter en tant que conteneur avec la commande **docker run**



Cette commande exécute un conteneur à partir de l'image **myimage:1.0** et mappe le port 80 (HTTP) du conteneur sur le port 80 de l'hôte.



Configuration de l'environnement d'un conteneur Docker

Vous pouvez configurer l'environnement d'un conteneur Docker en utilisant les variables d'environnement ou les fichiers de configuration. Par exemple, vous pouvez définir une variable d'environnement dans un Dockerfile :

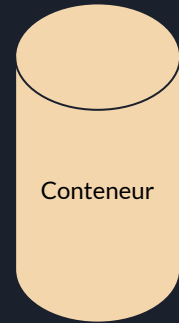
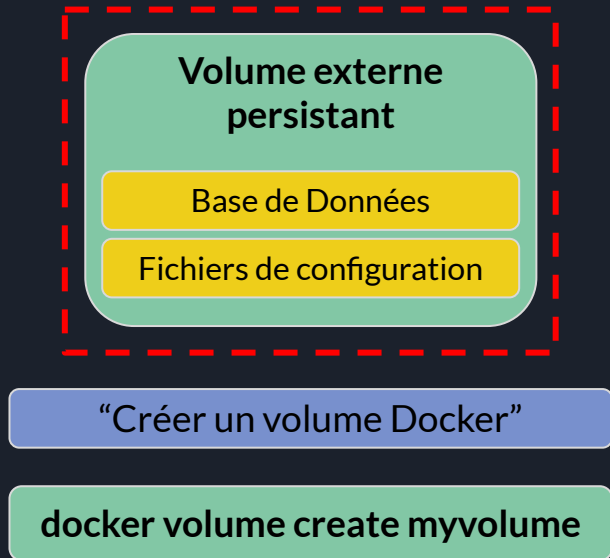
```
ENV MY_VAR=myvalue
```

Vous pouvez également spécifier des variables d'environnement au moment de l'exécution du conteneur avec la commande **docker run**. Par exemple :

```
docker run -e MY_VAR=myvalue myimage:1.0
```

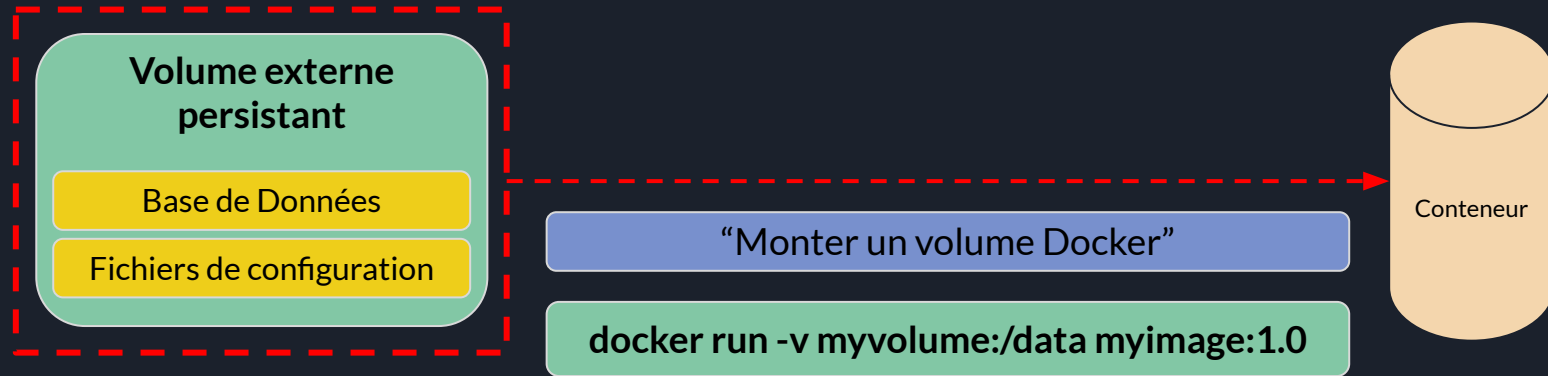
Cette commande exécute un conteneur à partir de l'image **myimage:1.0** et définit la variable d'environnement **MY_VAR** à **myvalue**.

Travailler avec des volumes Docker



Monter un volume Docker :
`docker run -v myvolume:/data myimage:1.0`

Travailler avec des volumes Docker



Cette commande monte un volume Docker "myvolume" dans le conteneur et le rend accessible dans le répertoire /data.



Travailler avec des réseaux Docker

Les réseaux Docker permettent aux conteneurs de communiquer entre eux et avec le monde extérieur. Vous pouvez créer des réseaux Docker pour isoler vos applications et les protéger contre les attaques.

Pour créer un réseau Docker, vous pouvez utiliser la commande **docker network create**. Par exemple :

```
docker network create mynetwork
```

Cette commande crée un réseau Docker appelé **mynetwork**.

Pour exécuter un conteneur Docker sur un réseau spécifique, vous pouvez utiliser l'option **--network** avec la commande **docker run**. Par exemple :

```
docker run --network mynetwork myimage:1.0
```

Cette commande exécute un conteneur à partir de l'image **myimage:1.0** sur le réseau **mynetwork**.



Limites et alternatives à Docker

Docker peut être plus difficile à mettre en place que d'autres technologies de virtualisation, en particulier pour les utilisateurs débutants.

Les conteneurs Docker partagent le même noyau de système d'exploitation, ce qui peut créer des problèmes de sécurité si le noyau est compromis.

Les conteneurs Docker peuvent être moins isolés que les machines virtuelles, ce qui peut causer des problèmes de performance ou de sécurité dans certaines situations.



Playground & TP Docker

Pour la suite et les exercices de ce cours vous devez effectuer le cours docker 101 :

<https://www.docker.com/101-tutorial/>

Puis :

<https://codedesign.fr/docker-desktop-windows-debutant/>