Cours Docker

Le conteneur d’applications

Par Cyrille MBIA

Sécurité systèmes numériques

Site web : https://itdreamtech.com

PLAN DU COURS

1. Comprendre Docker et les applications conteneurisées
2. Premiers pas avec l'utilisation de Docker CLI et l'utilisation d'applications conteneurisées
3. Dockerisez votre propre application dans une image Docker personnalisée

01

Comprendre Docker et les applications

MOUVEMENT VERS LE CLOUD, POURQUOI ?

- **Migrer la puissance dans le Cloud**
- **Changement d'environnement/plateforme simplifié**
- **Pas de « Bloquage / Fermeture » d'un constructeur conteneurisées**

LE CLOUD POUR VOUS C’EST QUOI ?

Le Cloud est une infrastructure informatique dématérialisée qui permet aux utilisateurs d'accéder à des ressources informatiques telles que des serveurs, du stockage, des bases de données et des logiciels via Internet. Plutôt que de disposer de ces ressources physiquement sur site, les utilisateurs peuvent les exploiter à distance, souvent moyennant des frais basés sur la consommation ou l'abonnement.

QUELQUES CHIFFRES

- 14 millions de « machines »
- 900 000 images Docker
- 12 milliards d’images téléchargées
- 3300 contributeurs au projet

Grâce au fait que Docker soit open source…

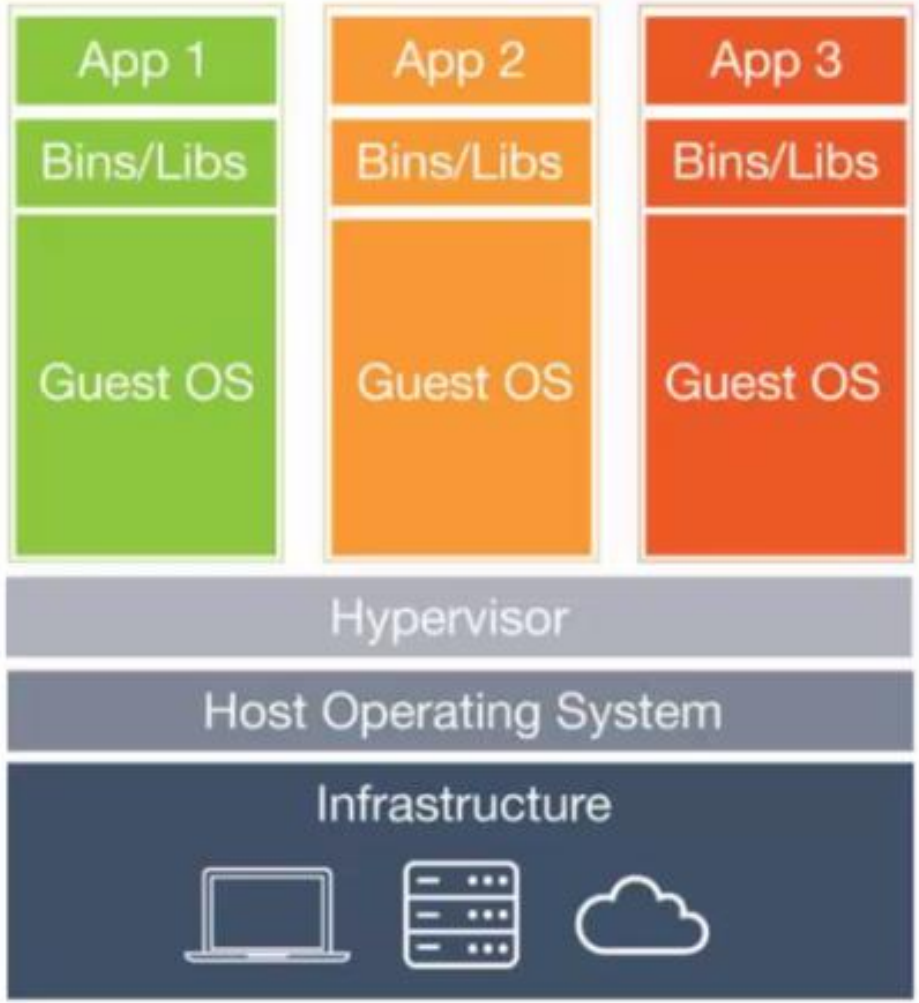
Le modèle de machines virtuelles

Une machine virtuelle est une représentation logique d'une machine physique.

Un des avantages est de Pouvoir installer tout et n'importe quoi.

LES LIMITATIONS DES MACHINES VIRTUELLES

- ❖ Des ressources allouées pour chaque machine (CPU, Disque, Ram)
- ❖ Un OS complet sur chaque machine (virtuelle)
- ❖ Plus il y a de machines plus il faut de puissance (ressources perdues)
- ❖ Ressources perdues par… des parties de l'OS virtualisées pour rien



Le modèle de containers

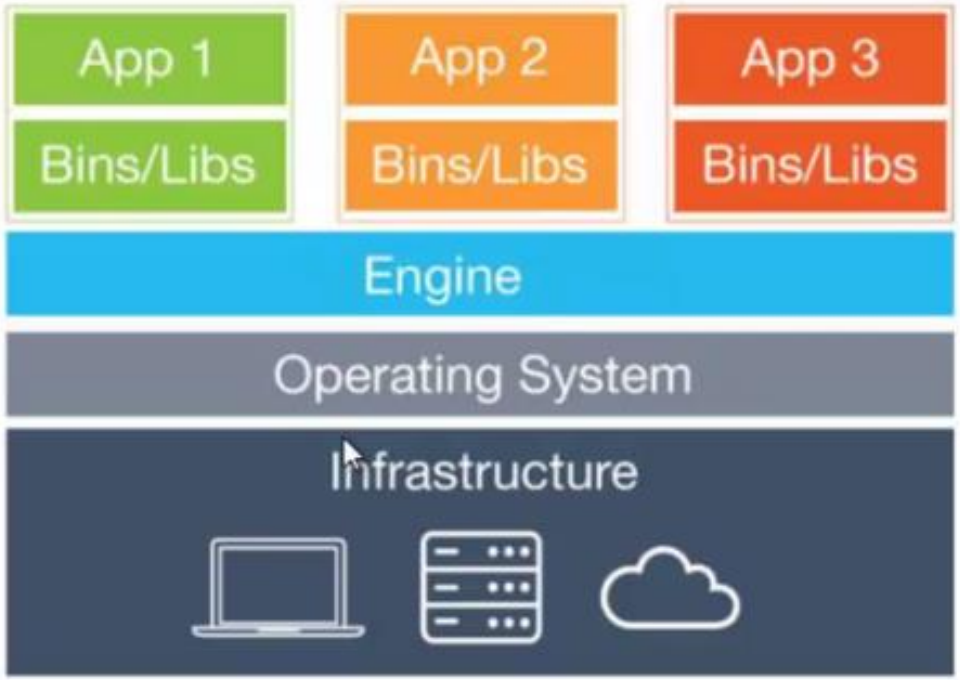
Un container est une forme de virtualisation, au mieux une isolation.

Un des avantages est que c'est extrêmement léger et rapide.

DOCKER… LES CONTAINERS / CONTENEURS À LA RESCOUSSE

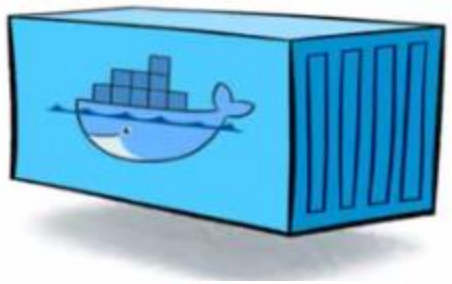
UN CONTENEUR C'EST…

- Un moyen standardisé de packager l'application
- Un moyen d'isoler les applications entres elles
- Un partage du noyau avec la machine physique



Types de containers linux

- Les systèmes d’exploitation
- Les services d’application



Les namespaces

Fonctionnalité du noyau linux qui a pour but d'isoler :

Les process namespaces (les processus)

Network namespaces (les cartes réseau)

Mount Namespaces (systèmes de fichiers)

UTS Namespaces (noms d'hôtes & noms de domaines)

IPC Namespaces (inter process communication)

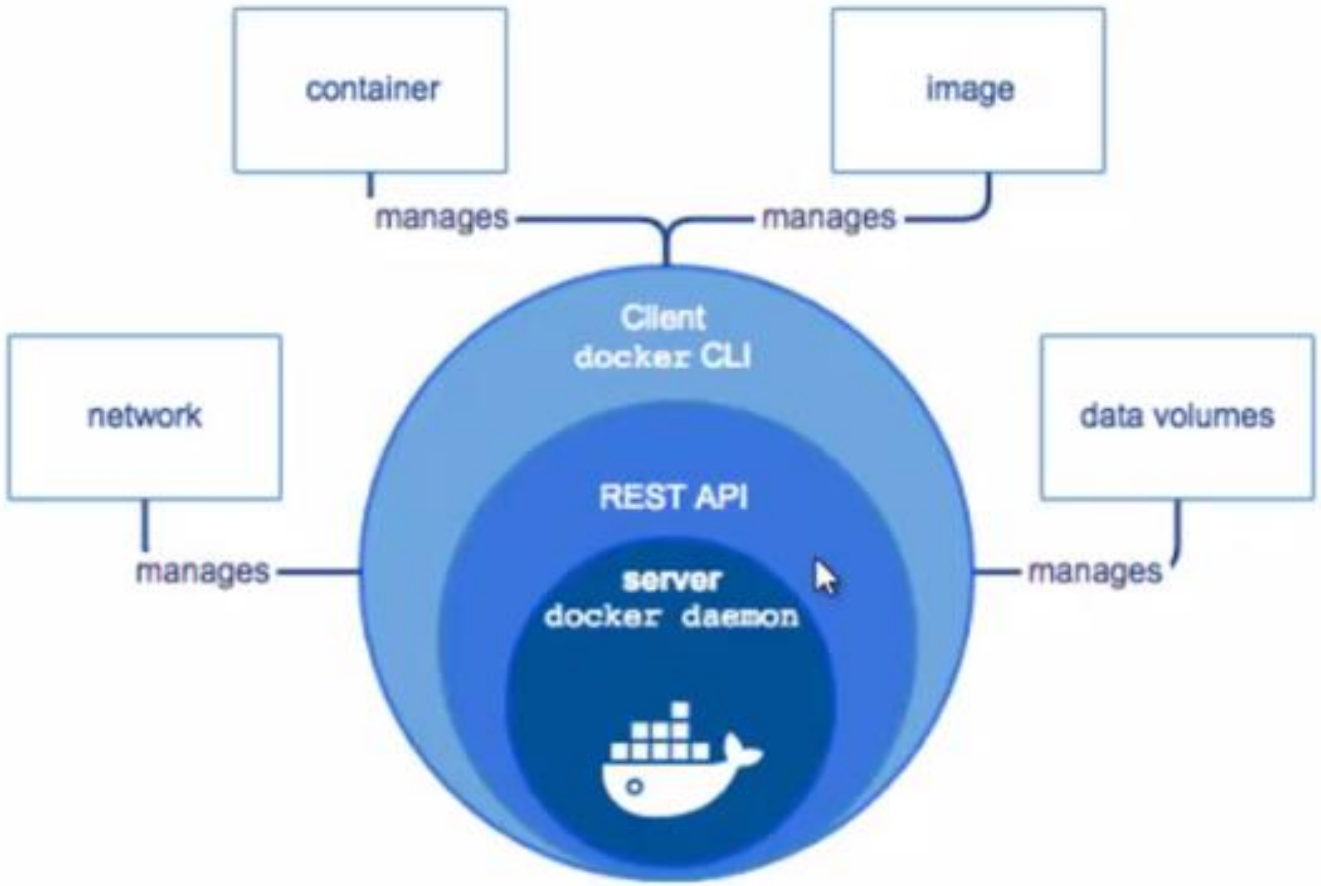
User Namespaces (Les utilisateurs)

L’écosystème docker

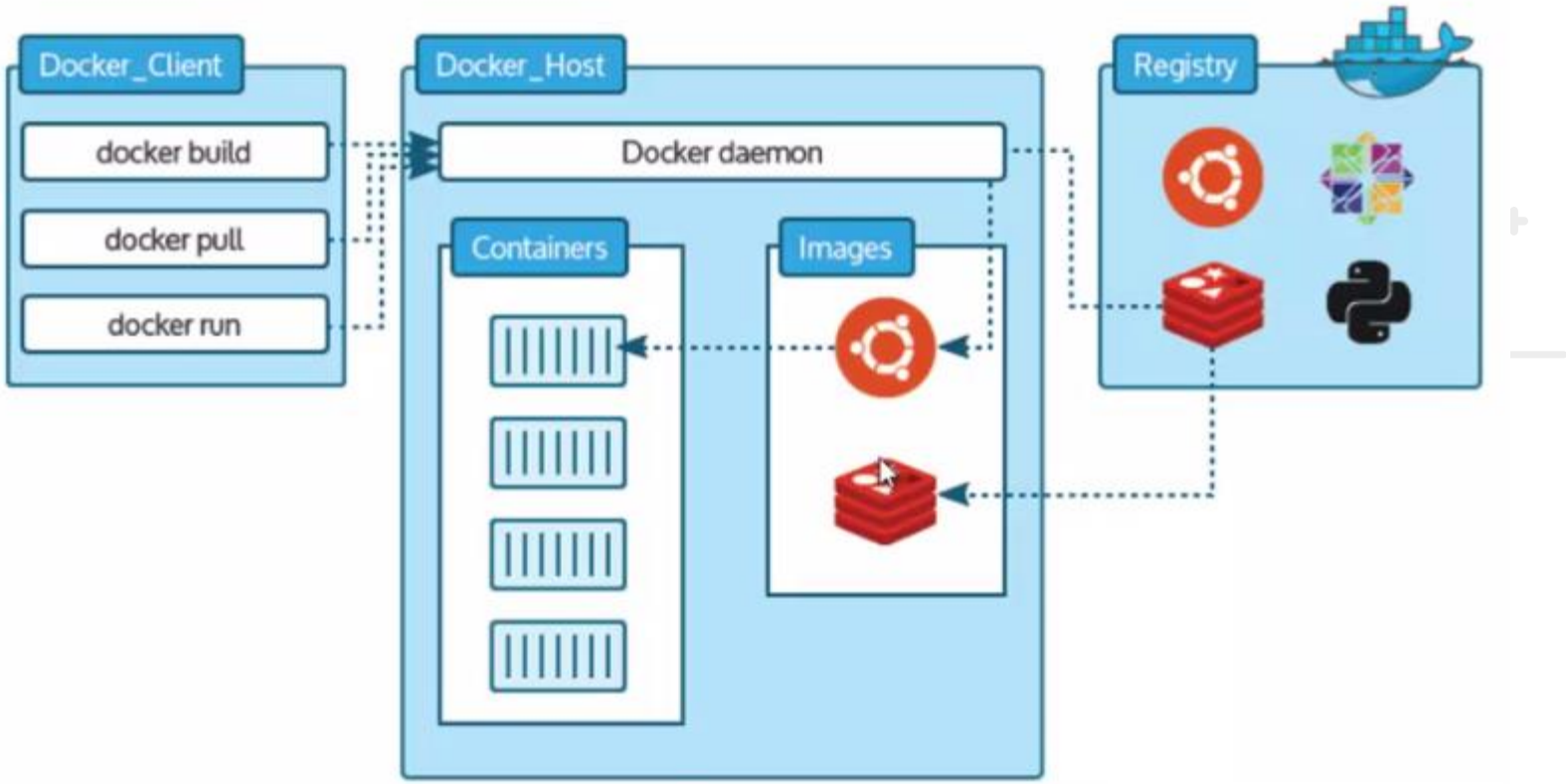
La devise de Docker est "Build, Ship and Run Any App, Anywhere", soit "Développer, Déployer et Exécuter n'importe quelle application, n'importe où".

Docker est une plateforme ouverte pour les développeurs et les administrateurs système permettant de développer, déployer, et exécuter des applications distribuées. Composé de Docker Engine, un outil d'exécution portatif et léger et Docker Hub, un service dans le cloud de partage d'images de conteneurs, Docker permet le développement rapide d'applications à partir de composants et élimine les risques d'incompatibilité entre environnements de développement, d'assurance qualité et de production. En conséquence, l'IT peut déployer plus rapidement et exécuter la même application, de la même manière, sur des ordinateurs portables, des machines virtuelles, ou dans le cloud.

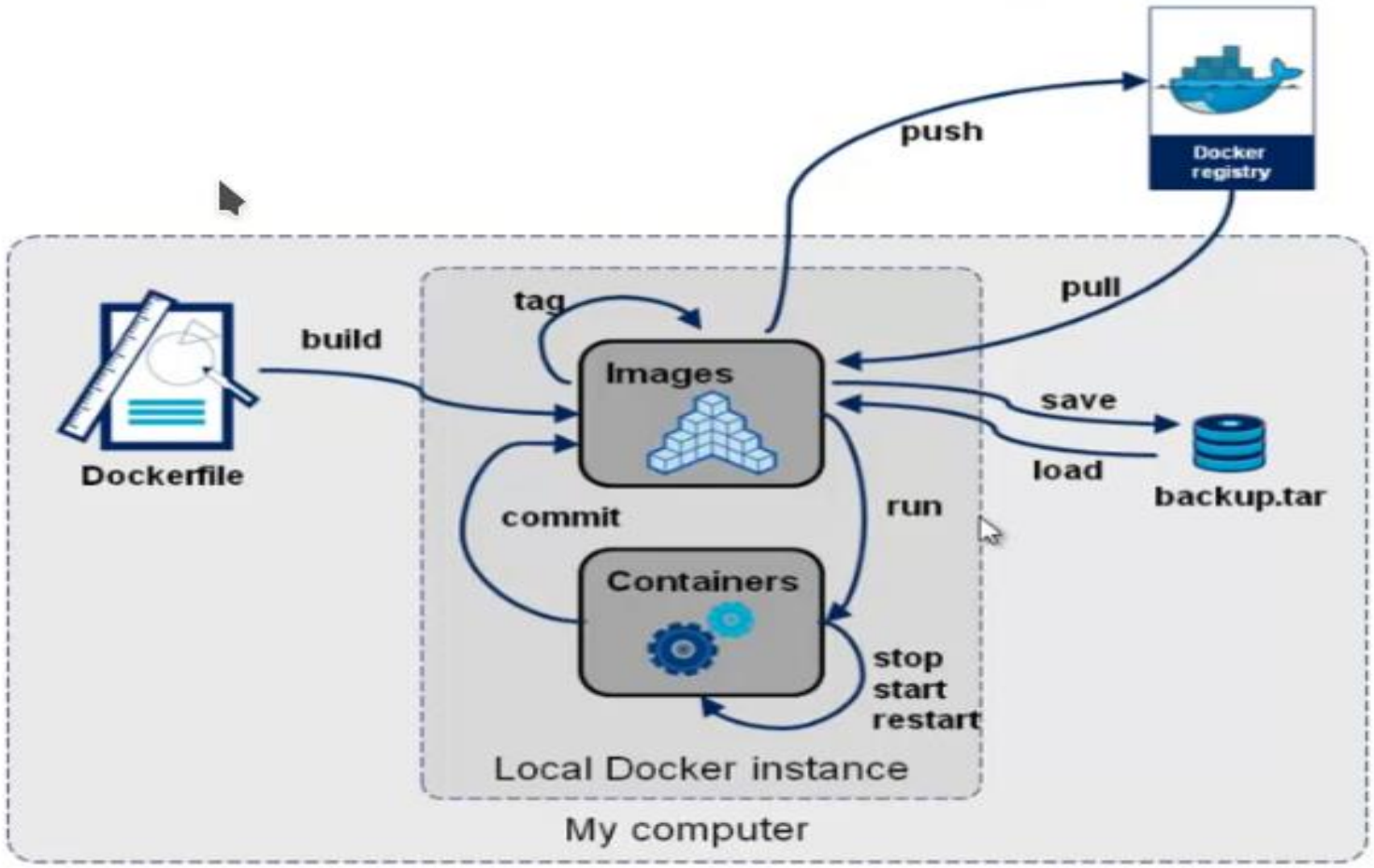
Architecture



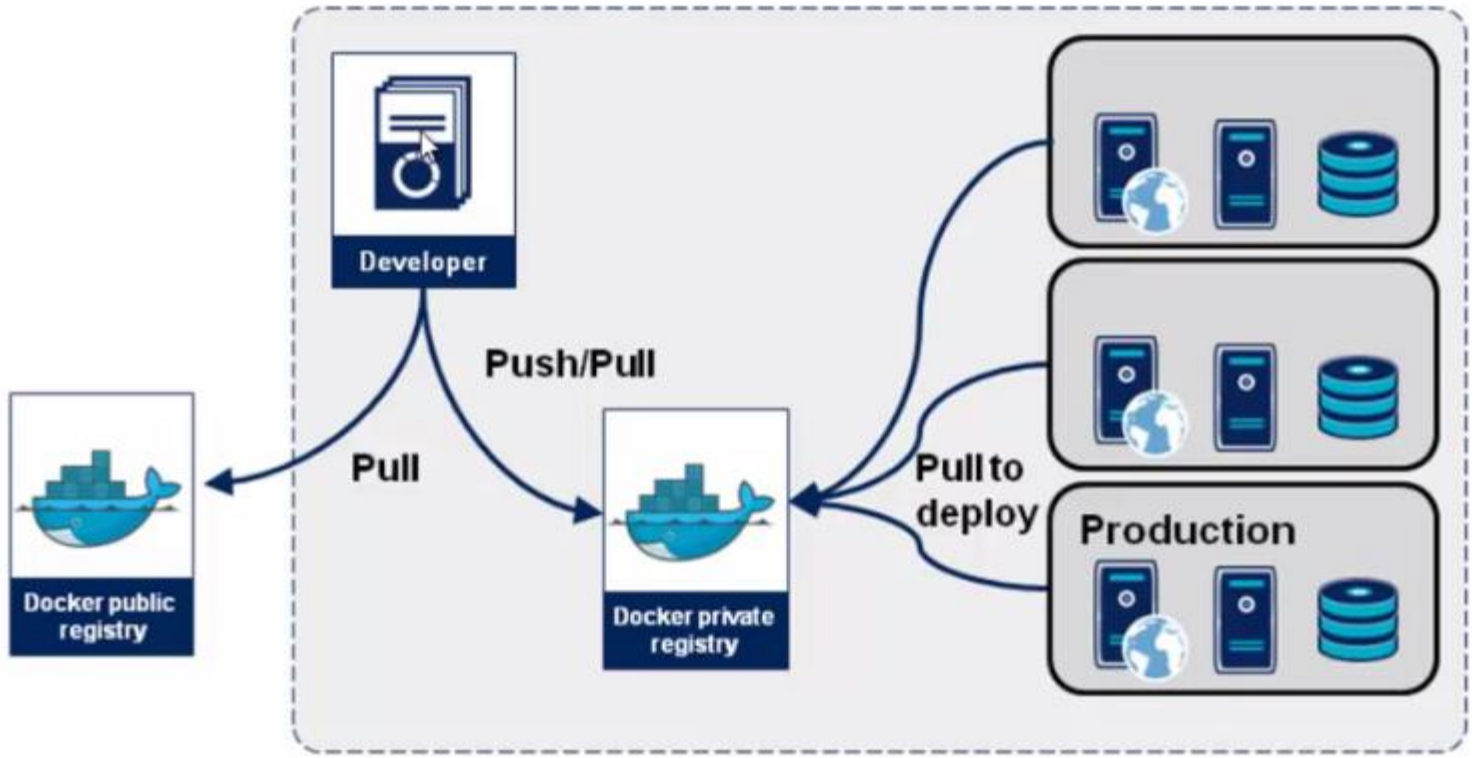
Architecture



« runner » une image



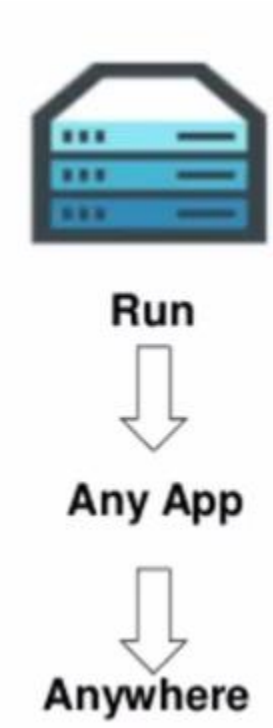
Les registres



Docker container

Les images sont ce que vous créez lorsque vous exécutez docker build; ils sont stockés dans un registre de conteneurs comme le Docker Hub et contiennent tous les fichiers et le code pour exécuter une application. Vous pouvez les considérer comme des fichiers ISO pour un système d'exploitation de machine virtuelle.

Les conteneurs sont créés à partir d'images et sont comme la véritable machine virtuelle qui exécute l'application. Vous pouvez avoir plusieurs conteneurs exécutés en parallèle sur la même image. Chaque conteneur aura son propre système de fichiers, éventuellement créé avec des « montages de volume » qui lient les données de l'hôte au conteneur.



02

Installation de docker engine

- Sur une machine ubuntu

Docker container

```
$ sudo apt-get update

$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

- curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

- https://docs.docker.com/engine/install/ubuntu/

Télécharger une image

- Il existe de nombreuses images disponibles publiquement que nous pouvons utiliser pour travailler avec docker. L'exemple ci-dessous télécharger une image hello-world en utilisant la commande docker pull :

```
[ ~ ]docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:31b9c7d48790f0d8c50ab433d9c3b7e17666d6993084c002c2ff1ca09b96391d
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
```

Télécharger une image

- La commande docker images permet de rechercher sur un registre docker tel que docker hub, afficher la liste des images disponibles localement sur votre système :

```
[ ~ ]docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
hello-world   latest    bf756fb1ae65   12 months ago    13.3kB
```

Créer un container

- Pour créer un conteneur à partir d'une image vous pouvez utiliser docker create :

```
[ ~ ]docker create hello-world

2ffd5f2c5a7562fbf1d7b89a14c11a52e5843dd7938f380a8cd53f3952da99de
```

Lancer un container

Pour exécuter un conteneur, nous pouvons utiliser la commande docker containers start pour démarrer un conteneur. Le **-i** exécute le conteneur de manière interactive et nous permet de voir le résultat

```
[ ~ ] docker container start -i 2ffd5f2c5a7562fbf1d7…
```

Hello from Docker !

Ce message montre que votre installation semble fonctionner correctement.

Lancer un container

Il existe un raccourci pour créer un conteneur à partir d'une image et l'exécuter avec le docker run. Cela créera un nouveau conteneur pour une image et l'exécutera :

```
[ ~ ] docker run hello-world
```

Hello from Docker !

Ce message montre que votre installation semble fonctionner correctement.

Liste des images

Pour voir quelles images sont déjà installées sur votre ordinateur, vous pouvez utiliser docker image ls. Nous pouvons voir notre image hello-world ci-dessous :

```
[ ~ ]docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
ubuntu        latest    f63181f19b2f   13 hours ago     72.9MB
hello-world   latest    bf756fb1ae65   12 months ago    13.3kB
```

Répertorier les conteneurs

Pour lister les conteneurs que nous avons construits, nous pouvons utiliser la commande docker containers ls.

L'indicateur **-a** nous permet de voir les conteneurs arrêtés et en cours d'exécution. Il y a deux conteneurs ci-dessous, celui qui a été construit avec la commande docker create et l'autre qui a été construit avec **docker run** :

```
[ ~ ]docker container ls -a

CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                       PORTS     NAMES
5017fd2b94c2   hello-world   "/hello"   7 minutes ago    Exited (0) 7 minutes ago               stoic_nobel
5f0cea57eacf   ubuntu        "bash"     10 minutes ago   Exited (127) 8 minutes ago             condescending_neumann
2ffd5f2c5a75   hello-world   "/hello"   14 minutes ago   Exited (0) 13 minutes ago              hungry_mclaren
```

Exécuter de manière interactive

L'exécution interactive de conteneurs vous permet d'exécuter des commandes à l'intérieur du conteneur si cela est nécessaire.

Nous pouvons utiliser l'image openjdk que nous avons utilisée auparavant : Cela nous permet d'exécuter des commandes Java ligne par ligne dans un shell Java

```
[ ~ ]docker run -it openjdk
Unable to find image 'openjdk:latest' locally
latest: Pulling from library/openjdk
a73adebe9317: Pull complete
8b73bcd34cfe: Pull complete
1227243b28c4: Pull complete
Digest: sha256:7ada0d840136690ac1099ce3172fb02787bbed83462597e0e2c9472a0a63dea5
Status: Downloaded newer image for openjdk:latest
Jan 21, 2021 4:48:58 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
|  Welcome to JShell -- Version 15.0.2
|  For an introduction type: /help intro

jshell> System.out.println(”hello world");
Hello world
```

Répertorier les processus en cours d'exécution

Pour voir quels conteneurs sont actuellement en cours d'exécution, nous pouvons utiliser la commande docker ps. C’est utile lorsque vous exécutez des conteneurs en arrière-plan.

```
[ ~ ]docker ps
CONTAINER ID   IMAGE     COMMAND    CREATED          STATUS          PORTS     NAMES
55e4a7c3ddcc   openjdk   "jshell"   11 seconds ago   Up 10 seconds             affectionate_kowalevski
```

Shell interactif

```
[ ~ ] docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
83ee3a23efb7: Pull complete
db98fc6f11f0: Pull complete
f611acd52c6c: Pull complete
Digest:
sha256:703218c0465075f4425e58fac086e09e1de5c340b12976ab9eb8ad26615c3715
Status: Downloaded newer image for ubuntu:latest
root@5f0cea57eacf:/#
```

Dockeriser une application

Dockerisez votre propre application dans une image docker personnalisée

Ajout d'un fichier Docker

Pour créer une image Docker personnalisée, nous avons besoin de créer un Dockerfile. Le fichier docker spécifie comment notre image doit être construite.

```
FROM openjdk

RUN useradd -ms /bin/bash ojdk

RUN mkdir -p /home/ojdk/app/ && chown -R ojdk:ojdk
/home/ojdk/app

WORKDIR /home/ojdk/app

COPY *.java ./
COPY junit-* ./

USER ojdk

RUN javac -cp "junit-4.10.jar:." *.java

COPY --chown=ojdk:ojdk . .

CMD [ "java", "-cp", "junit-4.10.jar:.",
"org.junit.runner.JUnitCore", "TestAdd", "TestSub"]
```

Dockeriser une application

Syntaxe du fichier Docker

From - L'image de base à utiliser

Run - Exécute les commandes lors de la construction l'image du docker file

Workdir - Spécifie le répertoire dans lequel les commandes sont exécutées à partir de

User - Change d'utilisateur

Copy - Copie les fichiers

CMD - Exécute des commandes lorsque le conteneur est exécuté

Dockeriser une application

Dockerfile expliqué

1. Utilisez l'image OpenJDK pour avoir un environnement Java préconfiguré
2. Ajouter un nouvel utilisateur « ojdk » que nous serons pour exécuter des scripts
3. Créez un répertoire qui contiendra nos fichiers et donner la permission à notre utilisateur
4. Changez le répertoire de travail en répertoire que nous avons créé
5. Copiez les fichiers Java et Junit
6. Passer à l'utilisateur ojdk
7. Compiler tout notre code
8. Copiez les fichiers dans le répertoire de travail et donner des autorisations à ojdk
9. Exécutez tous les tests

Dockeriser une application

Créer une image Docker

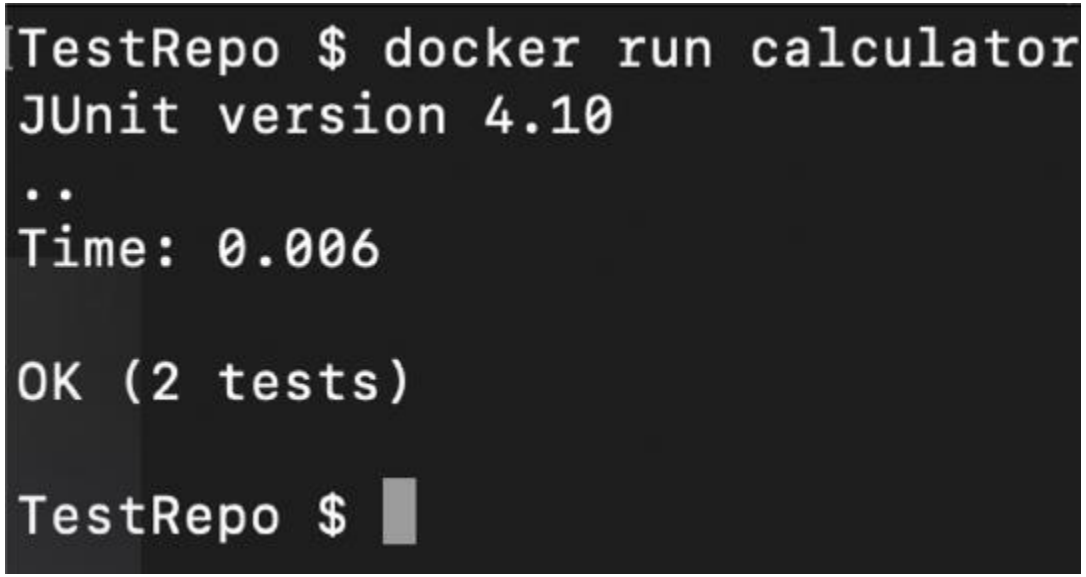
Pour créer une image Docker à l'aide d'un Dockerfile, nous pouvons utiliser la commande docker image build et fournissez-lui le répertoire où se trouve le Dockerfile. La balise --tag nous permet de nommer et de baliser l'image du docker.

```
TestRepo $ docker image build . --tag "calculator:latest"
Sending build context to Docker daemon  590.3kB
Step 1/9 : FROM openjdk
 ---> e105e26a0a75
Step 9/10 : COPY --chown=ojdk:ojdk . .
 ---> 47cff2b55e3c
Step 10/10 : CMD [ "java", "-cp", "junit-4.10.jar:.", "org.junit.runner.JUnitCore", "TestAdd", "TestSub"]
 ---> Running in c8395bc770b4
Removing intermediate container c8395bc770b4
 ---> 6b345c94e511
Successfully built 6b345c94e511
Successfully tagged calculator:latest
```

Dockeriser une application

Exécutez notre image personnalisée

Nous pouvons utiliser la commande docker run pour exécuter notre image et nous pouvons voir que nos tests sont en cours :



Dockeriser une application

Exécuter de manière interactive

Nous pouvons utiliser docker run -it pour exécuter notre image de manière interactive et ouvrir un bash

shell dans notre répertoire de travail :

```
[TestRepo $ docker run -it calculator bash
[[ojdk@419a727a1ca1 app]$ ls
 Calculator.class  README.md      TestSub.class
 Calculator.java   TestAdd.class  TestSub.java
 Dockerfile        TestAdd.java   junit-4.10.jar
 [ojdk@419a727a1ca1 app]$
```