Introduction à Docker

Docker est un projet open source (Apache 2.0) écrit en GO et hébergé sur GitHub: https://github.com/docker (https://github.com/docker).
Initialement porté par la startup DotCloud (renommée depuis Docker) fondée par deux français anciens de l'Epitech.
Docker est composé de trois éléments :

une API de type REST qui permet de communiquer avec le daemon Le client en CLI (command line interface) : commande docker
Par défaut, le client communique avec le daemon Docker via un socket Unix (/var/run/docker.sock) mais il est possible d'utiliser un socket TCP.

le daemon Docker qui s'exécute en arrière-plan et qui s'occupe de gérer les conteneurs (Containerd avec runC)

en CLI (command line interface) : commande docker

Introduction à Docker

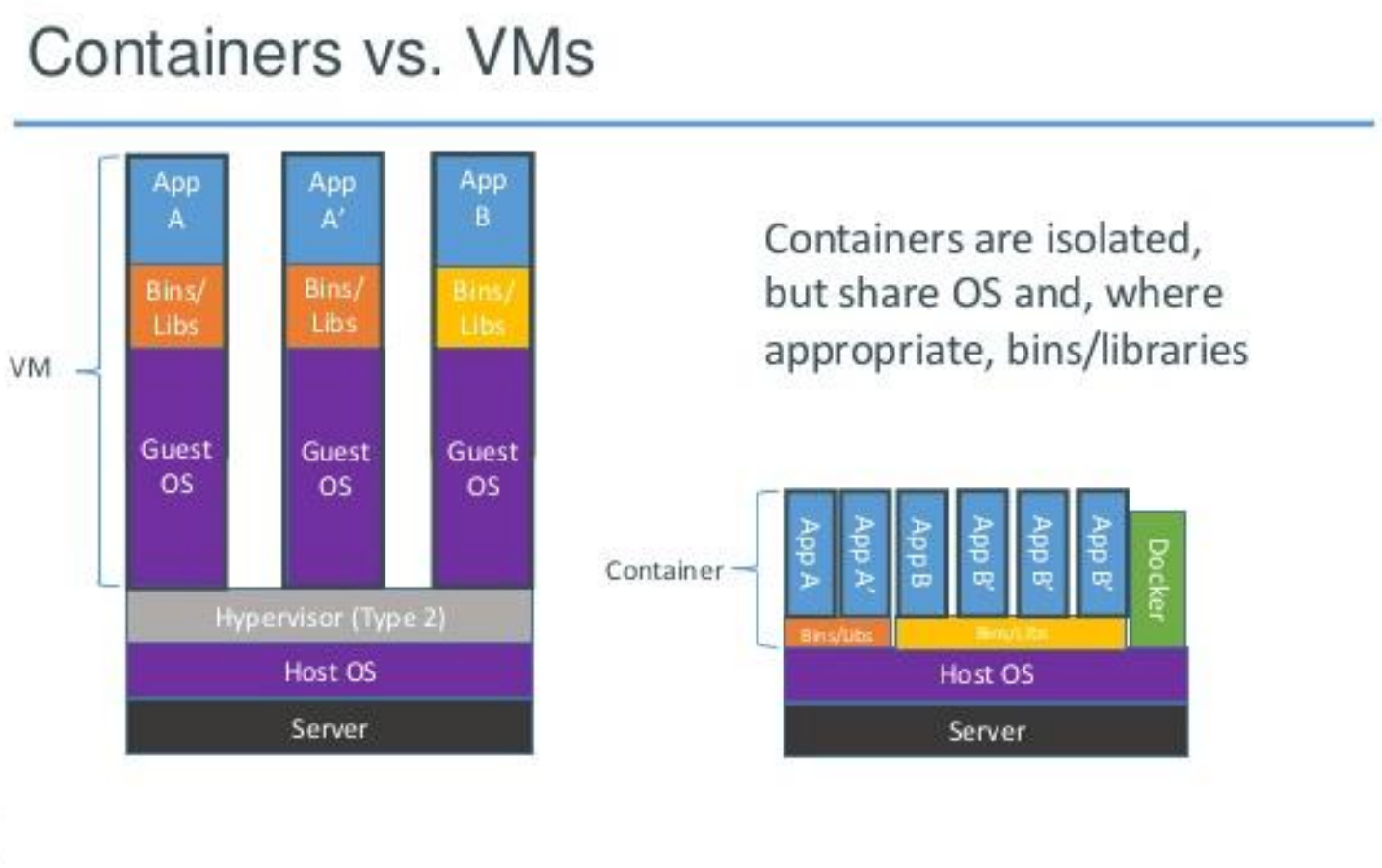
Par défaut, le client communique avec le daemon Docker via un socket Unix (/var/run/docker.sock) mais il est possible d'utiliser un socket TCP.

Docker c'est aussi un dépôt d'images (aussi appelé registry) : https://store.docker.com (https://store.docker.com)
Il contient les images officielles maintenues par Docker mais aussi celles mises à disposition par d'autres contributeurs.

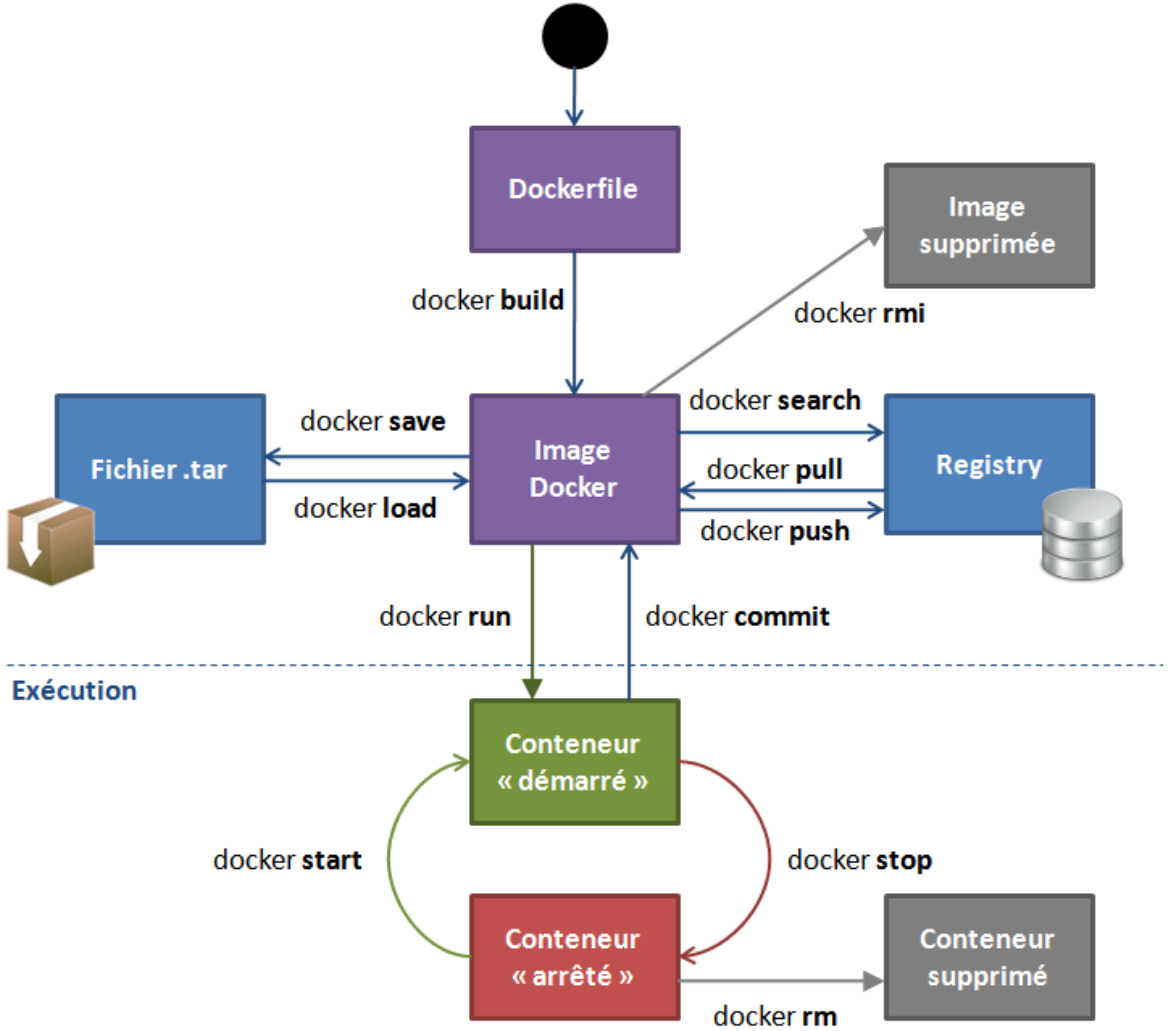
une image est un ensemble de fichiers inertes en read-only

Un conteneur est une instance une active (started) ou inactive (stopped) d'une image. L'execution d'un conteneur n'altère jamais une image.

Docker


Containers vs. VMs
App A
App A'
App B
Bins/ Libs
Bins/ Libs
Bins/ Libs
VM
Guest OS
Guest OS
Guest OS
Hypervisor (Type 2)
Host OS
Server
Containers are isolated, but share OS and, where appropriate, bins/libraries
Container
App A
App A'
App B
App B'
App B'
App B'
Docker
Bins/Libs
Host OS
Server
docker

Docker


Dockerfile
docker build
Image supprimée
docker rmi
Fichier .tar
docker save
docker load
Image Docker
docker search
docker pull
docker push
Registry
docker run
docker commit
Exécution
Conteneur « démarré »
docker start
docker stop
Conteneur « arrêté »
Conteneur supprimé
docker rm

Lexique

Conteneur : Image exécutable d'un environnement complet incluant code, librairies, outils et configuration

Image : template de conteneur en read-only contenant un systeme de base et une application.

Docker HUB : Dépôt public d'images mises à disposition par Docker

DockerHub (https://store.docker.com)

Dockerfile : fichier texte de description d'une image

Les images Docker

Faire une recherche sur le Docker Hub (commande : **docker search**)

```
$ docker search debian
NAME                DESCRIPTION                                     STARS ubuntu          Ubuntu is a Debian-based Linux operating s...   6898 debian           Debian is a Linux distribution that's comp...
2356 google/debian                                                  52    armhf/debian    Debian is a Linux distribution that's comp...   29
```

Ici “ubuntu” et “debian” sont des images officielles (les autres sont de la forme user/nom_image).

Elles sont maintenues par l’équipe docker et considérées comme plus “fiables”.

La colonne STARS donne une indication sur la popularité de l’image (mise en favoris).

Les images Docker

Télécharger une image (commande : **docker pull**)

```
$ docker pull debian Using default tag: latest
latest: Pulling from library/debian
3e17c6eae66c: Pull complete
Digest: sha256:26b2647845d66e20eeadf73d1c302a4ffd2cc9a74c39a52f2aced4f8 Status: Downloaded newer image for debian:latest
```

Lancer un conteneur (commande : **docker run**)

La commande docker run qui permet de lancer un conteneur peut également télécharger l'image si elle n'est pas disponible localement

```
$ docker run debian:stretch docker run debian:stretch
Unable to find image 'debian:stretch' locally stretch: Pulling from library/debian
Digest: sha256:26b2647845d66e20eeadf73d1c302a4ffd2cc9a74c39a52f2aced4f8 Status: Downloaded newer image for debian:stretch
```

Les TAGS

Lister les images présentent localement (commande : **docker images** ou **docker image ls**)

```
docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
hello-world         latest              f2a91732366c        2 weeks ago
debian              latest              6d83de432e98        4 weeks ago
debian              stretch             6d83de432e98        4 weeks ago
```

Ici les images debian ont le même ID (6d83de432e98) : c'est la même image mais avec un TAG différent

Exemple d'images et de TAGS proposés sur le dépôt officiel Debian :
https://hub.docker.com/r/library/debian/tags/ (https://hub.docker.com/r/library/debian/tags/)

Les TAGS

Ajouter un tag à une image (commande : **docker image tag**)

```
$ docker image tag debian:stretch debian:levasbr1
$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
debian              levasbr1            6d83de432e98        4 weeks ago
debian              stretch             6d83de432e98        4 weeks ago
```

Supprimer une image (commande : **docker rmi)**

Cette commande permet de supprimer une image à condition de ne pas avoir de conteneur lié.

On peut aussi utiliser l'ID abrégé (ex: f2A) pour désigner une image (en l'absence d'ambiguité)

Les conteneurs Docker

Lancer un conteneur à partir d'une image (commande : **docker run**)

```
$ docker run debian:latest cat /etc/issue Debian GNU/Linux 9 \n \l
```

L'état d'un conteneur dépend de la commande qui est lancée. Ici, le conteneur execute la commande cat et s'arrête dès quelle est terminée.

La commande docker ps qui permet de lister les conteneurs en cours d'executions ne retourne effectivement rien :

```
$ docker ps
CONTAINER ID   IMAGE   COMMAND   CREATED   STATUS   PORTS   NAMES
```

Les conteneurs Docker

Pour obtenir la liste complete des conteneurs, il faut utiliser l'option docker ps –a :

```
$ docker ps -a
CONTAINER ID       IMAGE          COMMAND          CREATED     STA
396767b854a9       debian:latest   "cat /etc/issue"    44 minutes  Exi
```

Le conteneur possède un identifiant unique (96767b854a9) et un nom généré aléatoirement (quizzical_easley).

Nommer un conteneur (option : **--name** ou **-n**)

On peut utiliser l'option --name pour nommer un conteneur de manière plus explicite :

```
$ docker run --name cmd_cat debian:latest cat /etc/issue Debian GNU/Linux 9 \n \
```

Cette commande a créé un nouveau conteneur :

```
docker ps -a
CONTAINER ID    IMAGE          COMMAND          CREATED          2770f96e4a3d    debian:latest   "cat
/etc/issue"    About a minute ago ef6b6f1c64a1    debian:latest   "cat /etc/issue"    4 minutes ago
```

Les conteneurs Docker

Obtenir une session intéractive (option : **-it**)

On peut obtenir une session intéractive (option -i) en se branchant sur l'entrée standard du conteneur et en connectant un pseudo terminal TTY (option -t) :

```
$ docker run -it  debian:latest /bin/bash root@eae2cce2669d:/#
```

Le prompt reprend le CID du conteneur (utiliser la commande exit pour quitter le conteneur).

Lancer un conteneur en mode daemon (option : **-d**)

On peut lancer un conteneur en mode daemon pour qu'il tourne en tâche de fond (le mode intéractif tourne au premier plan).

Les conteneurs Docker

```
$ docker run -d --name test_daemon nginx
4d81f9903afe1b777de6389954c762122b5aeea847f5b4f8953ad308bbc5203d // on affiche la
liste des conteneurs en cours d'execution :
$ docker ps
CONTAINER ID        IMAGE                         COMMAND
4d81f9903afe        nginx                         "nginx -g 'da
// on stoppe ce conteneur
$ docker stop test_daemon
```

Cycle de vie

Obtenir la configuration détaillée d'un conteneur

$ docker inspect <nom_conteneur> ou <CID>

Récupérer la sortie standard d'un conteneur

$ docker logs <nom_conteneur> ou <CID>

Afficher les processus en cours dans un conteneur

$ docker top <nom_conteneur> ou <CID>

Suspendre (freezer) et réactiver un conteneur

$ docker pause / unpause <nom_conteneur> ou <CID>

Arrêter / démarrer / tuer / redémarrer un conteneur

$ docker stop / start / kill / restart <nom_conteneur> ou <CID>

Exporter l'ensemble du système de fichier d'un conteneur dans une archive TAR

$ docker export <nom_conteneur> ou <CID> > archive.tar

Cycle de vie

Afficher les ports réseaux exposés par un conteneur

```
$ docker port <nom_conteneur> ou <CID>
```

Afficher les changements effectués sur les fichiers d'un conteneur (A=Ajout, D=Delete, C=Modif)

```
$ docker run -it --name test_diff debian /bin/bash

root@c7d328b087eb:/# apt update && apt -y upgrade
Ign:1 http://deb.debian.org/debian stretch InRelease
Get:2 http://deb.debian.org/debian stretch-updates InRelease [91.0 kB]
Get:3 http://deb.debian.org/debian stretch Release [118 kB]
Get:4 http://deb.debian.org/debian stretch Release.gpg [2434 B]       Get:5
http://security.debian.org stretch/updates InRelease [63.0 kB]
Get:6 http://deb.debian.org/debian stretch-updates/main amd64 Packages

$ docker diff test_diff
C /root
A /root/.bash_history
…
```

Cycle de vie

Commiter un conteneur pour obtenir une nouvelle image

Une solution pour “persister” les données ou les configurations consiste à commiter le conteneur pour créer une nouvelle image (clairement pas une bonne pratique).
Dans cet exemple, l’image est associée à l’utilisateur “xxxxxx” correspondant à un compte sur le docker-hub pour faciliter la publication des images sur la plate-forme public.

```
$ docker commit test_diff  <user>/<image>:<tag>
$ docker commit test_diff  xxxxxx/debian:latest
```

Supprimer un conteneur (il doit être arrêté…)

```
$ docker rm contener_name
```

Supprimer plusieurs conteneurs en utilisant les CID abrégés

```
$ docker rm 331 c7d
```

Registry

Pousser une image locale sur une registry privée ou public

On commence par “tagger” l’image locale pour la registry privée. Dans l’exemple le serveur qui héberge la registry se nomme “registry”

```
$ docker tag  <user>/debian:latest registry.domain.tld:<port/<user>/deb
$ docker tag levasbr1/debian:latest registry.inow.fr:5000/xxxxxx
$ docker images
REPOSITORY                TAG             IMAGE ID
registry.inow.fr/xxxxxx/debian   latest          55f2b1db89
```

Pousser l’image dans la registry :

```
$ docker push registry.inow.fr:5000/xxxxxx/debian:latest
The push refers to repository [registry.inow.fr:5000/xxxxxxx/deb
6737a6f8cc10: Pushed e1df5dc88d2c: Pushed latest: digest: sha256:56bbd19006e1d1166d56715e01771cf22fb2ca67335009b7
```

DockerFiles

Un dockerfile est un fichier texte (donc versionable) de description qui permet de générer une image. Il est basé sur une image standard auquel on ajoute les éléments propres à l'application que l'on veut déployer.

Instructions de bases :
FROM permet de définir depuis quelle base votre image va être créée
LABEL permet de définir l'auteur de l'image
RUN permet de lancer une commande, mais aura aussi pour effet de créer une image intermédiaire.
COPY permet de copier un fichier depuis la machine hôte ou depuis une URL
EXPOSE permet d'exposer un port du container vers l'extérieur
CMD détermine la commande qui sera exécutée lorsque le container démarrera
ENTRYPOINT permet d'ajouter une commande qui sera exécutée par défaut
WORKDIR permet de définir le dossier de travail pour toutes les autres commandes (par exemple RUN, CMD, ENTRYPOINT et ADD)
ENV permet de définir des variables d'environnements qui pourront ensuite être modifiées grâce au paramètre de la commande run --env <key>=<value>
VOLUMES permet de créer un point de montage qui permettra de persister les données. On pourra alors choisir de monter ce volume dans un dossier spécifique en utilisant la commande run -v

Volumes Docker

Par essence, les conteneurs Docker sont éphémères. Ils doivent être conçus pour pouvoir être supprimés sans perdre les données. Pour assurer cette persistance, on utilise les volumes Docker.

Ils sont initialisés à la création du container. Si l'image de base contient des données au point de montage spécifié, celles-ci sont copiés dans le volume lors de son initialisation

Ils peuvent être partagés entre plusieurs containers (exemple : pour créer des clusters de container)

Les changements apportés aux données du volume sont pris en compte immédiatement

Les données restent sur votre hôte même si le container vient à disparaître

Volumes Docker

Initialiser un volume (option : **-v**)

```
$ docker run -d -v /var/log --name test_volume debian:latest  /bin/slee
```

Ici l'option -v /var/log permet d'initialiser un volume indépendant du conteneur qui contiendra les logs du conteneur :

```
$ docker inspect test_volume
----------------8<--------------------
"Mounts": [
   {
      "Type": "volume",
      "Name": "e566163a0978a34dc1357cb4df7c9d86ddbb308de1c0ba26393e56
      "Source": "/var/lib/docker/volumes/e566163a0978a34dc1357cb4df7c
      "Destination": "/var/log",
      "Driver": "local",
      "Mode": "",
      "RW": true,
      "Propagation": ""
----------------8<-------------
```

Volumes Docker

La commande inspect permet de localiser le volume sur l'arborescence locale de l'hôte ("Source") et le point de montage dans le conteneur ("Destination")

```
$ sudo ls /var/lib/docker/volumes/e566163a0978a34dc1357cb4df7c9d86ddbb3 apt  btmp  faillog  lastlog  wtmp
```

Partager un dossier entre l'hôte et un conteneur (bind mounts)

L'autre utilisation des volumes est de partager un dossier entre le conteneur et le systeme hôte (exemple : pour qu'un développeur Web puisse éditer directement les fichiers depuis la machine hôte).

```
$ mkdir projet-web
$ docker run -d --name test_volume2  -p 8004:80 -v ~/projet-web:/var/www
```

Links Docker

L'installation de Docker crée par défaut un réseau nommé bridge (interface Docker0). Il est utilisé par les conteneurs si aucun réseau n'est spécifié au lancement (option --network=mon_reseau)

Par défaut, tous les conteneurs peuvent communiquer entre eux sur le réseau bridge. Pour plus de sécurité, il est possible de désactiver ce comportement (option DOCKER_OPTS="-icc=false" de la configuration de daemon)

```
$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
8dd02d7ad16f        bridge              bridge              local 72244137afae        host                host                local f7ab4027acf8        none                null                local

$ docker network inspect bridge
{
    "Name": "bridge",
    "Id": "8dd02d7ad16f9e47a774f0ee5a652a606ecc23bcc15c126d3e6fbf6fd
    "Created": "2017-12-05T16:44:54.821533922+01:00",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
       "Driver": "default",
       "Options": null,
       …
```

Docker-Compose

Docker-compose est un outil officiel Docker qui permet de simplifier le déploiement d'applications complexes (avec de nombreux conteneurs) en utilisant un simple fichier texte de description au format yaml. Il reprend l'ensemble des options qui seraient normalement à fournir à un docker run

```
version: '2'

services:
    db:
      image: mysql:5.7 volumes:
- db_data:/var/lib/mysql restart: always environment:
        MYSQL_ROOT_PASSWORD: bonjour
        MYSQL_DATABASE: wordpress MYSQL_USER: wordpress
        MYSQL_PASSWORD: wordpress

    wordpress:
      depends_on:
- dbimage: wordpress:latest ports:
- "8000:80" restart: always environment:
        WORDPRESS_DB_HOST: db:3306
        WORDPRESS_DB_USER: wordpress
        WORDPRESS_DB_PASSWORD: wordpress volumes:
    db_data:
```

Docker-Compose

Pour lancer les conteneurs, on utilise la commande docker-compose up ou docker-compose up -d pour lancer en arrière plan.

La commande doit être exécutée depuis le répertoire contenant le fichier yaml

```
$ docker-compose up
Creating network "levasbr1_default" with the default driver Creating volume "levasbr1_db_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
Digest: sha256:1f95a2ba07ea2ee2800ec8ce3b5370ed4754b0a71d9d11c0c35c934e Status: Downloaded newer image for mysql:5.7
Pulling wordpress (wordpress:latest)... latest: Pulling from library/wordpress e7bb522d92ff: Already exists 75651f247827: Pull complete
93b186f8edb2: Pull complete 77c007e2f556: Pull complete bf4da9c43c0b: Pull complete 11843d906ebb: Pull complete e03cc73ddbff: Pull
complete
```