LYCÉE ARISTIDE BERGÈS
SIO

Découverte

321 - Fonctionnement Docker

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision

Sommaire

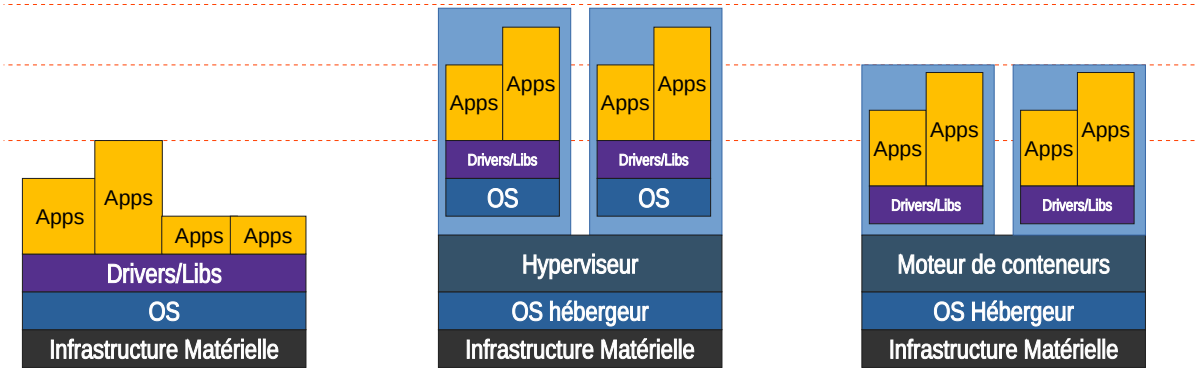
A Introduction

Docker est devenu incontournable depuis quelques années.

C'est une réponse à l'augmentation des ressources liées à la virtualisation, elle-même en réponse aux augmentations de besoins matériels.

A.1 Généralités

Le schéma ci-dessous permet d'avoir une compréhension globale des trois architectures existantes :



Architecture classique | Architecture virtualisée | Architecture conteneur

Une **architecture classique** nécessite une machine par service : un serveur web ne doit pas héberger la base de données ou les services de messagerie, etc. pour des raisons de sécurité.

Une **architecture virtuelle** enraille ce problème en utilisant une seule machine pour héberger plusieurs systèmes d'exploitation et leurs services associés : les machines virtuelles (VM) communiquent par le vSwitch (commutateur virtuel) et peuvent donc être sécurisée individuellement. Cependant, la contrepartie est une machine matérielle surdimensionnée comptant de 16 à 256 microprocesseurs et une mémoire vive de plusieurs téraoctets.

L'**architecture par conteneur** s'appuie sur un seul OS (moteur de conteneurs, par exemple Docker) et sur un fonctionnement sous forme de conteneur : ce sont uniquement les services qui communiquent entre eux.

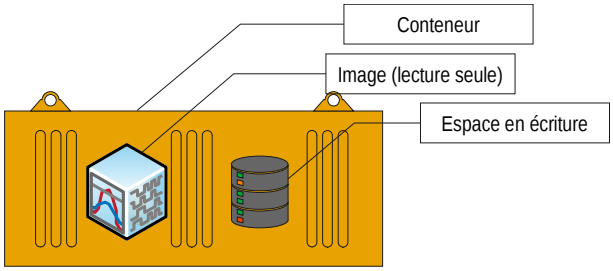
A.2 Fonctionnement de Docker

Docker n'est pas une solution de virtualisation : un conteneur peut tenir dans 200 Mo là où une machine virtuelle aura besoin de quelques gigaoctets.

Le moteur Docker optimise les ressources en réduisant les images des conteneurs au strict minimum (et notamment, des accès non-graphiques mais uniquement par CLI).

Cette optimisation est une des raisons pour laquelle on n'utilise pas les conteneurs en production : il s'agit donc d'une solution pour faciliter le développement et les tests de solutions applicatives ou de services.

Docker ne remplace donc pas les architectures virtuelles, mais permet d'en simuler le fonctionnement pour un coût en ressource moins élevé.

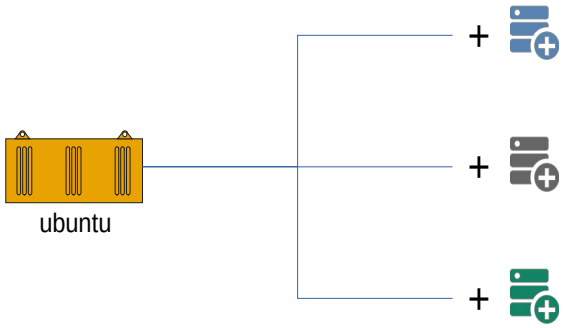


Le terme conteneur décrit une image de l'application ou du service (en lecture seule), à laquelle on ajoute un espace accessible en écriture. Toutefois, l'arrêt du conteneur efface les données écrites.

En effet, à l'origine, les conteneurs Docker sont conçus pour héberger des applications "sans état", donc sans persistance de données.

Un autre intérêt de la solution Docker, est la possibilité de pouvoir partager des conteneurs, par l'intermédiaire d'un catalogue appelé registry docker.

Enfin, si plusieurs images d'un même conteneur sont utilisées, chaque conteneur ne stockera que les différences entre l'image d'origine. Ce mécanisme utilise la fonctionnalité "Copy-on-Write" de l'UFS (Union File System) Docker. Moins d'encombrement mémoire et un démarrage d'une nouvelle image plus rapide.



Note : *Il existe malgré tout des possibilités pour avoir des données persistantes, en utilisant des montages de volume*.

B Utilisation

Une fois installé, Docker doit être présent dans les services.

B.1 Vérification du service

La commande pour vérifier que Docker est bien démarré est :

```
systemctl status docker
```

Résultat :

```
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
     Active: active (running) since Tue 2022-10-25 11:26:47 CEST; 15min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 7724 (dockerd)
      Tasks: 37 (limit: 18887)
     Memory: 277.2M
        CPU: 6.451s
```

B.2 Vérification du fonctionnement

La première commande permet d'afficher la version en cours.

```
docker version
```

Résultat :

```
Client:
 Version:           20.10.18
 API version:       1.41
 Go version:        go1.19.1
 Git commit:        b40c2f6b5d
 Built:             Sat Sep 10 11:31:10 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server:
 Engine:
  Version:          20.10.18
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.19.1
  Git commit:       e42327a6d3
  Built:            Sat Sep 10 11:30:17 2022
```

Si le service Docker n'est pas actif, la commande précédente n'affiche que la partie "Client" et un message : Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running?

B.3 Exécuter un conteneur

Le conteneur Hello-World est un excellent test pour valider que Docker puisse accéder aux dépôts Docker sur Internet.

```
docker run hello-world
```

La commande tente de lancer le conteneur hello-world. S'il n'est pas présent, l'image sera téléchargée. Le résultat lors de la première exécution devrait ressembler à ceci :

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:18a657d0cc1c7d0678a3fbea8b7eb4918bba25968d3e1b0adebfa71caddbc346
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Ce conteneur n'est pas interactif, il n'est pas possible d'interagir avec celui-ci.

La prochaine commande va permettre l'utilisation d'un shell bash dans un conteneur Ubuntu : pour cela, il faut ajouter le paramètre de commande `-it` pour préciser un fonctionnement interactif.

```
docker run -it ubuntu bash
```

La commande tente de lancer le conteneur Ubuntu. S'il n'est pas présent, l'image sera téléchargée. Il est intéressant de constater la taille de celle-ci (une trentaine de méga-octets), là où un environnement virtuel d'Ubuntu ferait environ deux gigaoctets :

```
~ docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
301a8b74f71f: Downloading [==========================================>        ]  28.46MB/30.43MB
```

Le résultat lors de la première exécution devrait ressembler à ceci (tapez la commande `ll` pour afficher les fichiers) :

```
root@72710ed8f06e:/# ll
total 56
drwxr-xr-x   1 root root 4096 Oct 25 09:27 ./
drwxr-xr-x   1 root root 4096 Oct 25 09:27 ../
-rwxr-xr-x   1 root root    0 Oct 25 09:27 .dockerenv*
lrwxrwxrwx   1 root root    7 Oct 20 11:49 bin -> usr/bin/
drwxr-xr-x   2 root root 4096 Apr 18  2022 boot/
drwxr-xr-x   5 root root  360 Oct 25 09:27 dev/
```

Ne quittez pas le conteneur, nous allons vérifier la persistance des données.

B.4 Non persistance des données

Lancez la commande suivante, puis avec la commande `ls`, vérifiez que le fichier listing.txt a bien été généré.

```
ls > listing.txt
ls
```

le résultat suivant s'affiche :

```
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  listing.txt  media  mnt  opt  proc
root  run  sbin  srv  sys  tmp  usr  var
```

Quittez l'invite de commande avec la commande `exit` puis relancez la commande

```
docker run -it ubuntu bash
```

Essayez de lister à nouveau le répertoire : le fichier listing.txt ne doit plus être présent.

B.5 Chercher un conteneur

Comme cela a été précédemment expliqué, il existe un catalogue des conteneurs en ligne, appelé Docker Hub. Il est donc pratique de pouvoir rechercher un conteneur ayant les caractéristiques souhaitées.

```
docker search php
```

La commande retourne la liste des conteneurs officiels ou créés par des utilisateurs enregistrés :

```
NAME                              DESCRIPTION                                     STARS     OFFICIAL
AUTOMATED
php                               While designed for web development, the PHP …   6848      [OK]
composer                          Composer is a dependency manager written in …   934       [OK]
adminer                           Database management in a single PHP file.       772       [OK]
phpmyadmin                        phpMyAdmin - A web interface for MySQL and M…   666       [OK]
mediawiki                         MediaWiki is a free software open source wik…   452       [OK]
webdevops/php-nginx               Nginx with PHP-FPM                              240                 [OK]
php-zendserver                    Zend Server - the integrated PHP application…   205       [OK]
yourls                            YOURLS is a set of PHP scripts that will all…   198       [OK]
webdevops/php-apache-dev          PHP with Apache for Development (eg. with xd…   151                 [OK]
bitnami/php-fpm                   Bitnami PHP-FPM Docker Image                    151                 [OK]
webdevops/php-apache              Apache with PHP-FPM (based on webdevops/php)    129                 [OK]
bitnami/phpmyadmin                Bitnami Docker Image for phpMyAdmin             37                  [OK]
circleci/php                      CircleCI images for PHP                         35
webdevops/php-nginx-dev           PHP with Nginx for Development (eg. with xde…   29                  [OK]
bitnami/phpbb                     Bitnami Docker Image for phpBB                  25                  [OK]
```

De manière intuitive, on constate que les conteneurs qui ne proviennent pas directement de l'équipe Docker ont pour nom l'identifiant du concepteur suivi d'un `/` suivi du nom du conteneur.

bitnami/phpbb est donc un conteneur de forum **phpBB** conçu par **Bitnami**.

B.6 Gérer les conteneurs

Les conteneurs peuvent être instables ou bien nécessiter un arrêt forcé. Les commandes suivantes permettent de faire cela.

La simulation consiste à utiliser 3 invites de commandes différentes :

- Dans la première console, lancez la commande `docker run -it ubuntu bash`
- Dans la deuxième console, lancez la commande `docker run -it ubuntu bash`

B.6.1 Affichage des conteneurs

Dans la troisième console, nous allons vérifier quels sont les conteneurs en cours d'exécution :

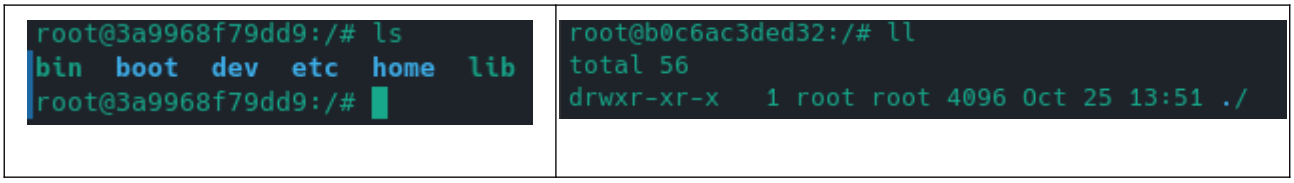
```
docker ps
```

Le résultat devrait être similaire à ceci :

```
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS          PORTS     NAMES
b0c6ac3ded32   ubuntu    "bash"    6 seconds ago    Up 6 seconds              zealous_panini
3a9968f79dd9   ubuntu    "bash"    13 seconds ago   Up 13 seconds             quizzical_montalcini
```

La commande ps est similaire à la commande Linux et indique les **p**roce**s**sus en cours.

La **première colonne** donne l'ID du container (vous pouvez le vérifier, car c'est l'ID utilisé dans la commande bash de chaque container Ubuntu :



Chaque conteneur a une image et peut avoir une commande.

Le status du conteneur est indiqué dans la **cinquième colonne**.

Enfin, dans la **dernière colonne**, on constate qu'un nom est automatiquement créé par Docker, humainement plus pratique pour atteindre un conteneur.

B.6.2 Renommage de conteneur

Si le nom simplifié d'un conteneur ne vous convient pas, vous pouvez le renommer :

```
docker rename zealous_panini capitaine_kirk
```

La commande `docker ps -a` permet de voir tous les conteneurs lancés depuis le début de session.

B.6.3 Arrêt d'un conteneur

Il est alors possible d'arrêter un conteneur de la mémoire avec la commande `docker stop` :

```
docker stop zealous_panini
```

Note : On peut remplacer le nom par l'ID.

B.6.4 Arrêt forcé d'un conteneur

Si un container refuse de s'arrêter, on peut le supprimer avec la commande `docker kill` :

```
docker kill zealous_panini
```

B.6.5 Pause d'un conteneur

Dans un cadre de débogage ou bien pour simuler un problème, il est possible de mettre en pause un conteneur avec les commandes `docker pause` et `docker unpause` :

```
docker pause gallant_gagarin
docker unpause gallant_gagarin
```

Toutefois, une partie des flux est conservée en mémoire (buffer) et lorsque le conteneur sort de pause, il traitera les données enregistrées.

La commande `unpause` est différente de la commande `restart` qui réinitialise le conteneur.

B.7 Débogage

B.7.1 logs

Il existe peu de commandes pour effectuer un débogage sur les conteneurs, mais la commande `docker logs nom_conteneur` permet d'avoir un certain contrôle sur les données.

```
docker logs star_trek
```

Voici un exemple sur un conteneur de type ubuntu avec la commande bash :

```
root@f52aa6618a6c:/#
root@f52aa6618a6c:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@f52aa6618a6c:/# pwd
/
```

La journalisation retransmet tous les échanges entre l'utilisateur et le conteneur.

B.7.2 événements

Une autre commande surveille les événements sur l'ensemble des conteneurs : `docker events`.

```
docker events
```

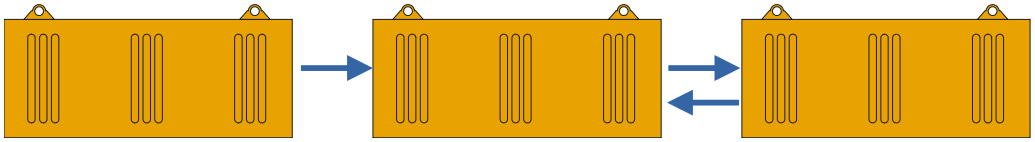
Quitter le terminal de l'image ubuntu bash donnera :

```
2022-10-25T16:55:10.527812839+02:00 container die
f52aa6618a6cc5ea2b2703f566d6de70cc8bd448fd3fb97b03d23908d270f538 (exitCode=127,
image=ubuntu, name=star_trek)
2022-10-25T16:55:10.680456229+02:00 network disconnect
a7956159f02562bfdc28f25b755ac1a995211bb1e85093a46c13977c4eabc2a6
(container=f52aa6618a6cc5ea2b2703f566d6de70cc8bd448fd3fb97b03d23908d270f538,
name=bridge, type=bridge)
```

C Utilisation avancée

Docker facilite le déploiement de plusieurs services ou applications sur un poste de travail, en utilisant des conteneurs comme briques de base.

Le fort intérêt des conteneurs, est de pouvoir communiquer avec l'extérieur et même de pouvoir communiquer avec d'autres conteneurs. La plupart du temps, cela passera par des ports réseaux (TCP ou UDP).



D'autre part, la possibilité d'avoir des données persistantes implique l'usage de point de montage entre les répertoires des conteneurs et le système d'exploitation lui-même.

Ce chapitre s'attache donc à faire découvrir ces fonctionnalités.

C.1 Persistance et volume

Avant de faire communiquer plusieurs conteneurs ensemble, nous allons utiliser un conteneur proposant plusieurs sous-services. Le conteneur https://hub.docker.com/r/tomsik68/xampp servira d'exemple.

Dans votre espace de travail, créez un répertoire www qui contiendra les pages web de votre site fictif : le chemin doit être simple (par exemple sous Linux : ~/www et sous Windows : c:\www)

La commande pour activer le conteneur est

```
docker pull tomsik68/xampp
```

Cette commande rappelle le fonctionnement de GIT et télécharge donc le conteneur.

```
Using default tag: latest
latest: Pulling from tomsik68/xampp
76dff75df4d9: Downloading [================>                                  ]
15.85MB/50.44MB
a153acca7146: Download complete
3853421bda0e: Downloading [=============>                                     ]
13.97MB/57.3MB
fd73eb793493: Downloading [>                                                  ]
6.46MB/436.1MB
9beac1daca1e: Waiting
ed620ebce689: Waiting
```

C.1.1 Montage simple

Lorsque le téléchargement est terminé, nous pouvons analyser le conteneur et déterminer les ports accessibles.

```
docker inspect tomsik68/xampp
```

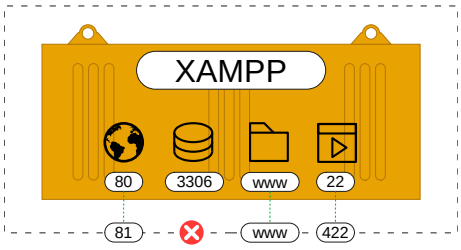
La commande affiche de nombreux renseignements, mais nous porterons notre attention sur la partie concernant les ports exposés :

```
"ExposedPorts": {
    "22/tcp": {},
    "3306/tcp": {},
    "80/tcp": {}
},
```

Hormis le port SSH (22), le port HTTP (80) correspond au serveur web (Apache) et le port MySQL (3306) à la base de données.

Les paramètres de la commande `docker run` permettent de démarrer le conteneur avec de nombreuses options, notamment l'affectation des ports des conteneurs aux ports réels de l'hôte, ou bien la création d'un montage pour rendre des données locales accessibles par le conteneur

```
docker run --name myXampp -p 422:22 -p 81:80 -d -v ~/www:/www tomsik68/xampp:8
```



L'option `-p` associe un port réel à un port du conteneur. L'option `-v` associe un chemin réel à un chemin du conteneur. L'option `--name` permet de donner immédiatement un nom au conteneur.

Le montage du dossier est intéressant mais implique de connaître le dossier présent dans le conteneur à l'avance. C'est pourquoi on l'appelle montage en aveugle.

Docker télécharge très rapidement le complément manquant (la version 8 de PHP) puis lance le service. Pour le vérifier, connectez-vous sur http://localhost:81/dashboard/ avec votre navigateur.

La page http://localhost:81/www/ pointe désormais sur le contenu de votre répertoire local www.

Pour effectuer un test, il suffit de créer une page index.php qui contient le code suivant :

index.php

```
<html>
<head>
     <title>Page test Docker XAMPP</title>
</head>
<h1>Vérification de l'exécution de PHP</h1>
<?php
     phpinfo();
?>
</html>
```

Rafraîchissez la page.

Pour vous connecter en SSH sur le conteneur, il faut utiliser la commande SSH avec le compte 'root' (mot de passe 'root') sur le port 422 :

```
ssh root@localhost -p 422
```

C.1.2 Volume Docker

Une autre alternative, permet au conteneur d'écrire et lire des données de fonctionnement (fichiers de configuration)

On utilise alors la commande `docker volume`. On peut créer, lister et vérifier l'emplacement du volume avec les trois commandes suivantes :

```
docker volume create www
docker volume ls
docker volume inspect www
```

L'usage du volume se fait dans la commande `docker run` avec l'option `--mount` :

```
docker run -it --name star_trek --mount source=www,destination=/unDossier ubuntu bash
```

Attention à ne pas insérer d'espace à partir du mot-clé 'source', en particulier de chaque côté de la virgule.

Si vous faites une commande `ls`, un répertoire nommé `unDossier` est bien présent.

```
root@34551f414eaf:/# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run
sbin  srv  sys  tmp  unDossier  usr  var
```

Ce dossier pointe vers le volume Docker www, dont les informations sont enregistrées sur l'emplacement donné par la commande `docker volume inspect www` :

```
[
    {
        "CreatedAt": "2022-10-27T19:20:18+02:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/www/_data",
        "Name": "www",
        "Options": {},
        "Scope": "local"
    }
]
```

Attention : les volumes sont indépendants des conteneurs et il est possible d'utiliser le même volume pour plusieurs conteneurs. Pour effacer un volume, il faut utiliser la commande `docker volume rm www`.

Désormais, il est possible de modifier les fichiers de configurations d'un conteneur, en connaissant sa structure. Par exemple, pour XAMPP, les fichiers de configuration se trouvent dans /opt/lampp

Cependant, la saisie des commandes Docker est fastidieuse, nous verrons dans un autre document, comment Docker-compose simplifie la création d'un environnement complet facilement.

Il existe aussi la solution Docker Desktop (anciennement Kinematic) qui est une interface graphique pour Docker. Cependant, comme pour GIT, il est préférable de bien connaître les commandes vues précédemment, pour utiliser correctement cet outil.