BTS SIO SISR

Situation professionnelle numéro 2

Mise en place et configuration d'un container Linux sous Docker

C'est lors de cette situation que je vais essayer de répondre à la problématique.

BTS SIO SISR CFA INSTA 75005

Léo LE CORRE

Plan de situation :

La demande

L'équipe de développement de la KLLT Bank me contacte à propos d'une demande. Ils veulent une solution fiable et simple d'utilisation pour héberger un serveur WEB localement. Je propose l'utilisations de conteneurs avec Linux et en utilisant Nginx. Pour déployer facilement la solutions aux demandeurs.

La différence entre les machines virtuelles et les containers

Une VM offre une isolation totale (avec son propre système d'exploitation complet), contrairement aux conteneurs qui partagent un noyau du système d'exploitation de l'hôte. Étant donné que la VM possède un système d'exploitation complet, elle demande plus de ressources comme le CPU, la RAM, et la taille du disque, contrairement au conteneur qui ne possède pas d'OS complet. De ce fait, la vitesse (/ vitesse de démarrage) est impactée. Une VM nécessite un hyperviseur tel que Hyper-V et Vmware tandis que les containers demande des outils d'orchestration tels que Docker, Kubernetes et Docker compose.

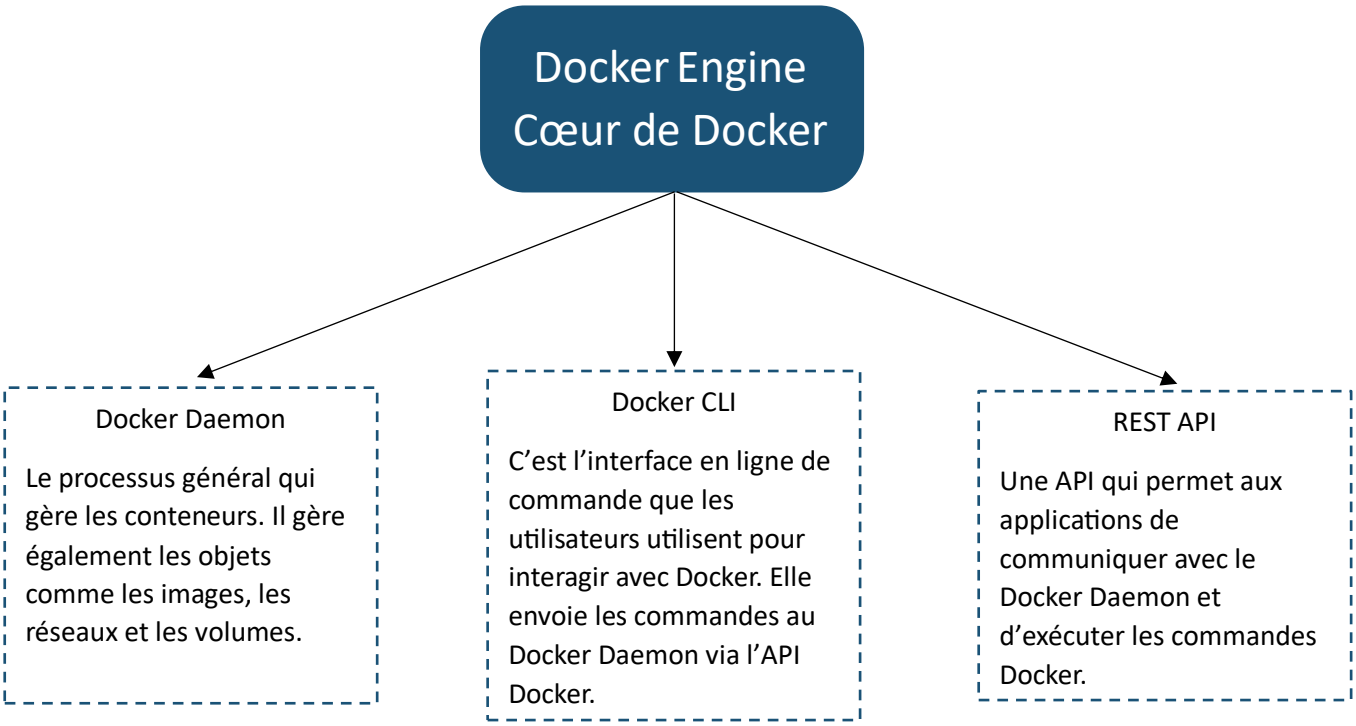
Pourquoi utiliser les VM et pourquoi utiliser les conteneurs

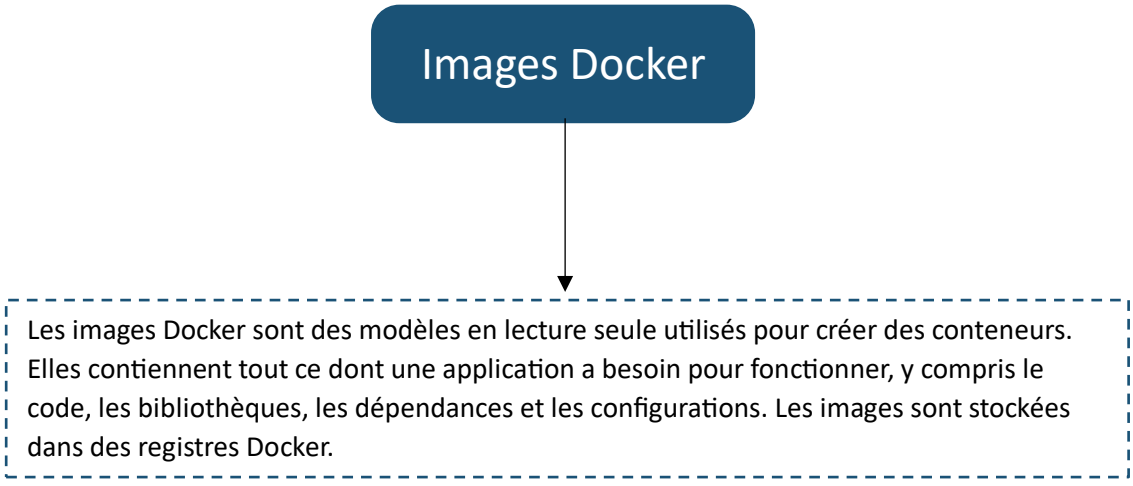
On utilise généralement une VM quand on a besoin d'une isolation complète (souvent dans le domaine du hacking). Quand on doit exécuter des applications nécessitant des OS différents.

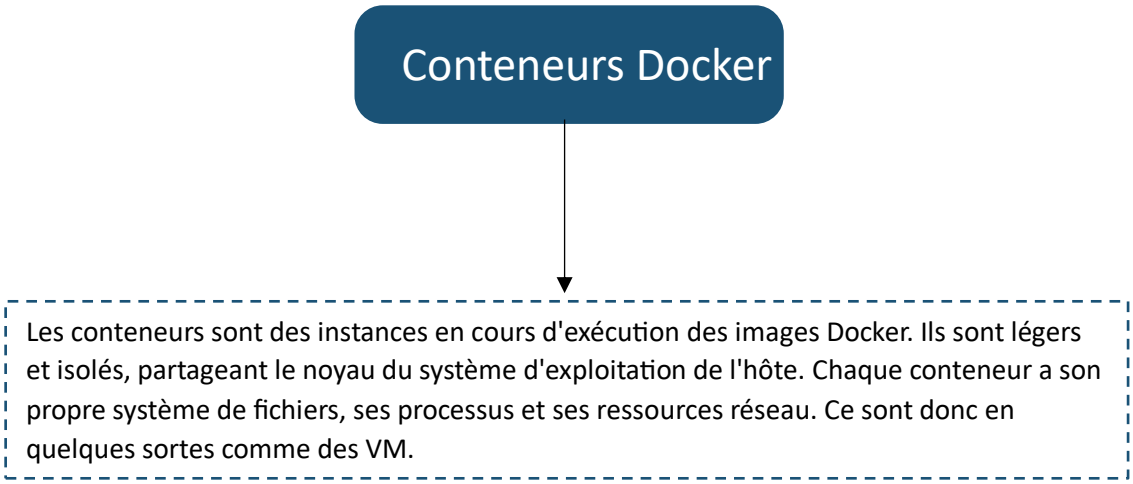
On utilise généralement les conteneurs quand on à besoin d'un déploiement rapide et efficace. Pour des services plus petits et quand on a pas besoin d'un cloisonnement.

Le Lexique de Docker

Le Docker Engine est le cœur de Docker. Il se compose de trois parties principales qui sont :


Docker Engine
Cœur de Docker
Docker Daemon
Le processus général qui gère les conteneurs. Il gère également les objets comme les images, les réseaux et les volumes.
Docker CLI
C'est l'interface en ligne de commande que les utilisateurs utilisent pour interagir avec Docker. Elle envoie les commandes au Docker Daemon via l'API Docker.
REST API
Une API qui permet aux applications de communiquer avec le Docker Daemon et d'exécuter les commandes Docker.


Images Docker
Les images Docker sont des modèles en lecture seule utilisés pour créer des conteneurs. Elles contiennent tout ce dont une application a besoin pour fonctionner, y compris le code, les bibliothèques, les dépendances et les configurations. Les images sont stockées dans des registres Docker.


Conteneurs Docker
Les conteneurs sont des instances en cours d'exécution des images Docker. Ils sont légers et isolés, partageant le noyau du système d'exploitation de l'hôte. Chaque conteneur a son propre système de fichiers, ses processus et ses ressources réseau. Ce sont donc en quelques sortes comme des VM.

Registres Docker

Les registres Docker sont des dépôts où les images Docker sont stockées et partagées. Le Docker Hub est le registre public par défaut, mais il est possible de configurer des registres privés pour des besoins spécifiques. Il y a une très grande diversité d'images.

Réseaux Docker

Avec les VM, sous Hyper-V, il existe des commutateurs virtuels interne et externes. Chez VmWare, Bridge, Host etc… Tout comme ce dernier, Docker permet de créer et de gérer des réseaux pour les connecteurs pour choisir comment vont communiquer les conteneurs.

Bridge :

Un réseau par défaut pour les conteneurs sur un seul hôte.

Host :

Les conteneurs partagent le réseau de l'hôte.

Overlay :

Permet de connecter des conteneurs sur plusieurs hôtes.

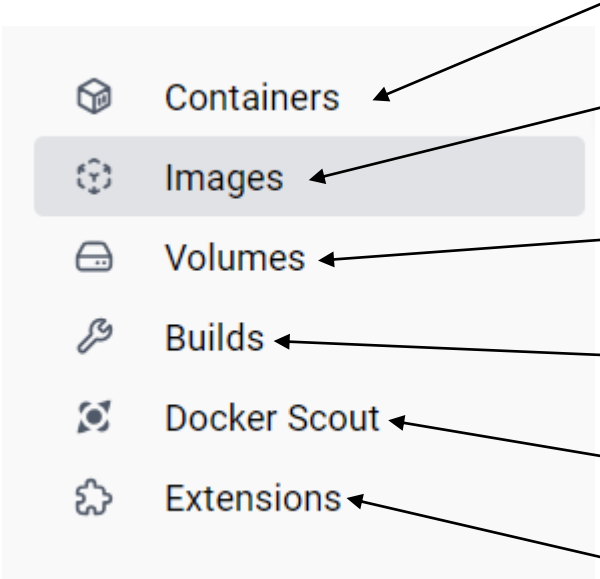
Macvlan :

Attribue une adresse MAC unique à chaque conteneur pour qu'il apparaisse comme un périphérique physique sur le réseau.

Configuration du conteneur

Pour commencer, installons Docker Desktop. Pour ce faire, aller sur leur site et cliquer sur « Docker Desktop for Windows – x86_64 »

Puis suivre l'installation par défaut. Une fois le logiciel installé, on y retrouve plusieurs sous catégories :



Là où l'on retrouve les containers crées en cours d'utilisation

Là où l'on retrouve les images (registres docker comme spécifier ci-dessus)

Là où l'on retrouve les « disques » de nos conteneurs, ils permettent de stocker les données persistantes.

Permet de créer des images à partir de docker files

Analyse des images, tableau de bord, logs, solution de correction de problèmes de sécurités

Ajouter des extensions

Une fois fait, nous allons pouvoir commencer la création du conteneur linux. Pour ce faire, allons dans le cmd (Win + R, et taper CMD) et écrire :

```
C:\Users\leole>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
de44b265507a: Download complete
Digest: sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

C:\Users\leole>
```

Cette commande vas installer l'image spécifiée (Ubuntu) sur la machine hôte.

Puis écrire :

```
C:\Users\leole>docker run -it ubuntu
root@abe5a6937047:/#
```

Cette commande nous permet d'avoir accès au terminal linux du conteneur. -it permet de maintenir le conteneur en mode interactif.

Nous allons maintenant mettre à jour Linux :

```
C:\Users\leole>docker run -it ubuntu
root@abe5a6937047:/# apt update
Get:1 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1028 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [707 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [719 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [15.3 kB]
50% [5 Packages 8855 kB/19.3 MB 46%]
```

Puis entre la commande :

```
root@abe5a6937047:/# apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Mise en place du serveur Web

Notre OS est maintenant à jour, Nous allons maintenant héberger un site WEB localement avec Nginx.

Pour ce faire, nous allons créer le fichier nginx_project dans la racine Windows, puis entrer dedans :

```
PS C:\Users\leole> mkdir C:\nginx_project


    Répertoire : C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         05/01/2025     23:30                nginx_project


PS C:\Users\leole> cd C:\nginx_project
```

Puis écrire la commande echo suivis du code entre guillemets :

echo "<!DOCTYPE html>

<html>

<head>

<title>Mon Serveur Nginx</title>

</head>

<body> <h1>Bienvenue sur mon serveur Nginx !</h1> </body>

</html>" >

index.html

```
PS C:\nginx_project> echo "<!DOCTYPE html> <html> <head> <title>Mon Serveur Nginx</title> </head> <body> <h1>Bienvenue sur mon serveur Nginx !</h1> </body>
</html>" > index.html
```

Enfin, utiliser la commande :

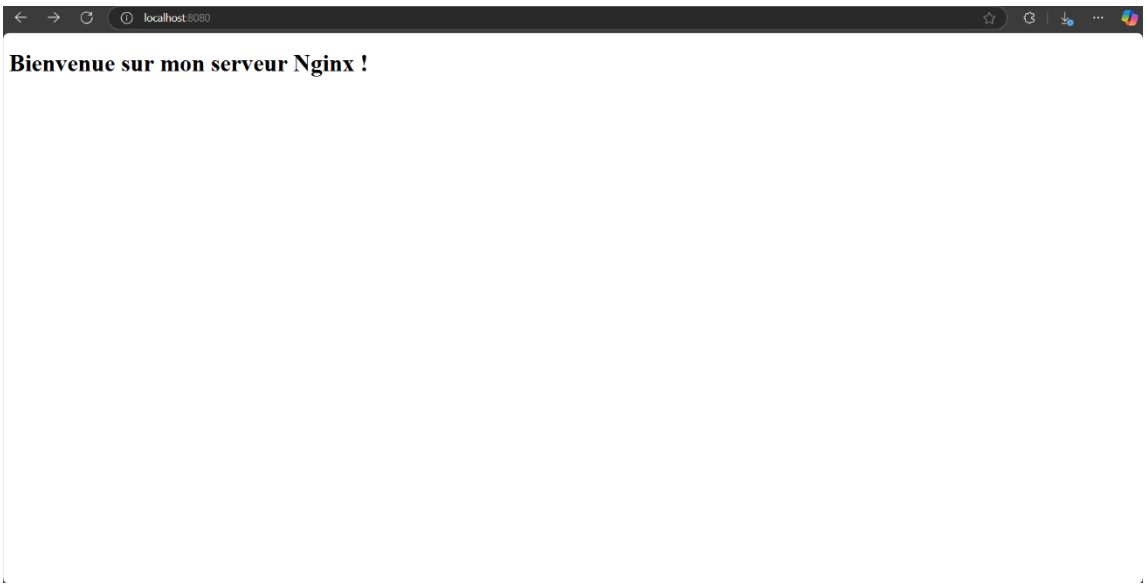
docker run -it -p 8080:80 -v C:\nginx_project:/usr/share/nginx/html nginx

Afin de créer un nouveau conteneur en mode -it, avec le chemin vers le code et enfin, en expliquant que c'est avec nginx qu'on veut.

```
PS C:\Users\leole> docker run -it -p 8080:80 -v C:\nginx_project:/usr/share/nginx/html nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/01/05 22:35:43 [notice] 1#1: using the "epoll" event method
2025/01/05 22:35:43 [notice] 1#1: nginx/1.27.3
2025/01/05 22:35:43 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/01/05 22:35:43 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
2025/01/05 22:35:43 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/01/05 22:35:43 [notice] 1#1: start worker processes
2025/01/05 22:35:43 [notice] 1#1: start worker process 29
2025/01/05 22:35:43 [notice] 1#1: start worker process 30
2025/01/05 22:35:43 [notice] 1#1: start worker process 31
2025/01/05 22:35:43 [notice] 1#1: start worker process 32
2025/01/05 22:35:43 [notice] 1#1: start worker process 33
2025/01/05 22:35:43 [notice] 1#1: start worker process 34
2025/01/05 22:35:43 [notice] 1#1: start worker process 35
2025/01/05 22:35:43 [notice] 1#1: start worker process 36
2025/01/05 22:35:43 [notice] 1#1: start worker process 37
2025/01/05 22:35:43 [notice] 1#1: start worker process 38
2025/01/05 22:35:43 [notice] 1#1: start worker process 39
2025/01/05 22:35:43 [notice] 1#1: start worker process 40
```

Le serveur héberge correctement le code, pour vérifier, on peux aller dans le navigateur internet et chercher :

Localhost :8080



Pour arrêter le serveur, il faut chercher son ID :

```
PS C:\Users\leole> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS                  NAMES
c08159425fb6   nginx     "/docker-entrypoint.…"   4 minutes ago    Up 3 minutes    0.0.0.0:8080->80/tcp   hungry_shamir
abe5a6937047   ubuntu    "/bin/bash"              34 minutes ago   Up 34 minutes                          friendly_heyrovsky
PS C:\Users\leole>
```

On peux voir que l'ID est c08159425fb6 .

On peut donc l'arrêter :

```
PS C:\Users\leole> docker stop c08159425fb6
c08159425fb6
PS C:\Users\leole>
```

Conclusion

La solution proposée répond à la problématique posée et les tests sont concluants. En utilisant Docker et Nginx, nous avons réussi à créer un environnement d'hébergement efficace et sécurisé pour servir des fichiers HTML. Les étapes de création du répertoire, de configuration du fichier index.html, et de lancement du conteneur Docker ont été suivies avec succès. Enfin, les tests d'accès au serveur via http://localhost:8080 ont confirmé que le serveur Nginx fonctionne correctement et sert le fichier HTML comme prévu. Nous avons également fermée le conteneur après. Cette solution offre une méthode fiable et reproductible pour déployer des applications web conteneurisées.