Emma RAMIREZ

Reporting Docker

**Docker**

**Docker est un système de virtualisation qui consiste à la création de conteneurs. Ces conteneurs virtualisent le système d’exploitation d’un serveur. Ils vont contenir une ou plusieurs applications, celle-ci pourrons être facilement déployées sur n’importe quel système qui exécute docker.  
Cet outil offre un système de tests et de démonstration complet à moindre cout.**

Les manipulations suivantes se feront sur un powershell.

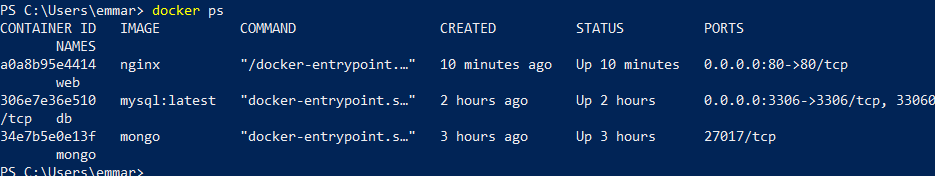
Création de conteneur :

Une image contenant texte

Description générée automatiquement

Ici on crée un serveur web en utilisant les ports 80 en précisant le nom du conteneur. L’option -d permet de continuer à utiliser la fenêtre PowerShell pendant que le conteneur continue de tourner.

Une fois le conteneur créé on peut le vérifier comme ceci :

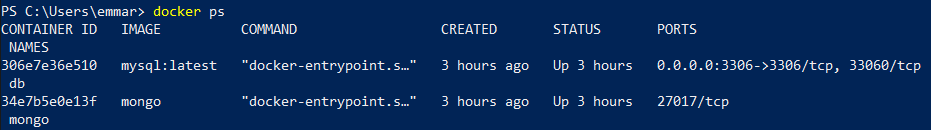


On aperçoit notre conteneur avec l’image nginx. En plus de voir ce qui s’exécute, il est également possible de voir les conteneurs arrêtés.

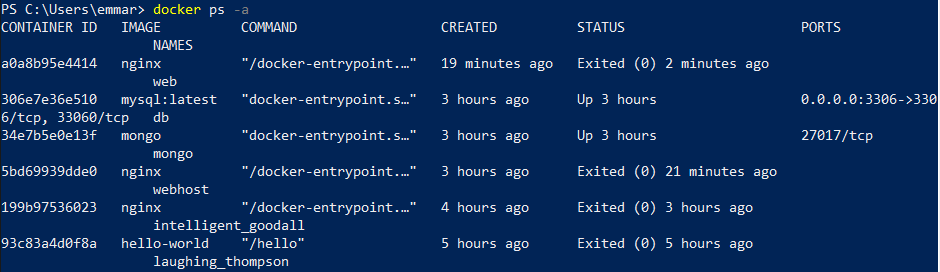
Nous allons donc arrêter notre conteneur :



Vérifions qu’il n’apparait plus en tant que conteneur en marche :

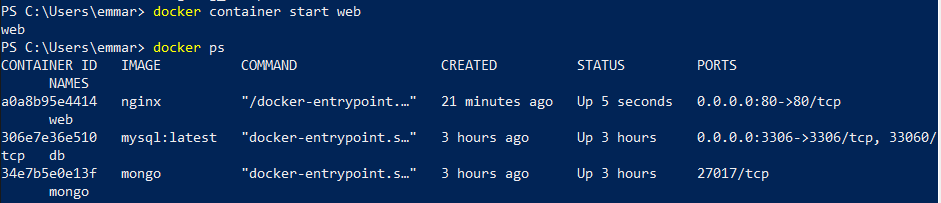


On voit donc ici qu’il n’est plus présent.



Grâce a l’option -a, nous voyons l’état de tous les conteneurs. On peut donc voir celui que nous venons tout juste d’arrêter, ainsi que la précision en termes de temps « Exited 2 minutes ago »

Nous allons maintenant le redémarrer :



Nous allons maintenant supprimer le conteneur :

Une image contenant texte

Description générée automatiquement

On peut voir dans un premier temps qu’on ne peut pas supprimer un conteneur qui s’exécute. Je l’ai donc arrêté et ensuite supprimé. On voir grâce a la dernière commande qu’il a bien disparu de la liste.

Création d’image :

Une image contenant texte

Description générée automatiquement

Ici j’ai créé une image Debian, on peut le vérifier, qu’elle s’affiche dans la liste ci-dessous.

Une image contenant texte

Description générée automatiquement

On peut la supprimer de la manière suivante :

Une image contenant table

Description générée automatiquement

L’image n’apparaît plus dans la liste, elle est donc bien supprimée.

Maintenant je souhaite créer un conteneur avec une image kali linux. Pourtant, je n’ai pas d’image en local. Je vais donc allée la récupérer sur docker hub directement :

Une image contenant texte

Description générée automatiquement

Je lance un conteneur de la distribution Parrot récupérée sur le docker hub puis, pour rentrer en commande prompt directement en lançant le conteneur, il faut ajouter l’option -it

Une image contenant texte

Description générée automatiquement

Etant donné qu’on a exit juste avant, il est désactivé il faut le relancer ensuite, pour rentrer en commande prompt quand le conteneur est déjà exécuté, il faut utiliser exec -it.

Une image contenant texte

Description générée automatiquement

Variable d’environnement

Lors de la création d’un conteneur, il est possible d’y inclure une variable d’environnement

Une image contenant texte

Description générée automatiquement

Grâce à l’option -e on définit le mot de passe root de notre conteneur mySQL.

De plus docker offre la possibilité de d’observer les ressources utilisées par les conteneurs comme un gestionnaire des tâches, de la manière suivante :

Une image contenant texte, extérieur, tableau de points

Description générée automatiquement

On peut afficher les process en cours avec la commande top : Une image contenant texte, périphérique, jauge, mètre

Description générée automatiquement

On peut avoir diverses informations sur le conteneur grâce à cette commande : Une image contenant texte

Description générée automatiquement

On peut également y retrouver sa configuration réseau.

Voici comme faire s’afficher les logs d’un conteneur :

Une image contenant texte

Description générée automatiquement

Création d’un nouveau réseau qui s’appellera LAN1. On affichera ensuite sa configuration :

Une image contenant texte

Description générée automatiquement

Création d’un conteneur dans le réseau :



On peut le vérifier de la manière suivante :

Une image contenant texte

Description générée automatiquement

Je vais installer un paquet sur mon conteneur :

Une image contenant texte

Description générée automatiquement

Parrot étant une distribution ultra complète, il est compliqué de trouver des paquets qui ne sont pas encore installés, cependant, cela se fait de cette manière.

J’ai créé un deuxième conteneur de la même façon.

Sur ce nouveau conteneur, je fais un Docker container inspect pour avoir son adresse IP.

Je peux ensuite faire un ping :

Une image contenant texte

Description générée automatiquement

Le ping à réussi, les conteneurs sont sur le même réseau et réussissent à communiquer.

**DOCKERFILES**

Nous pouvons créer des images plus complètes et personnalisé en utilisant un dockerfile.

Sous un éditeur comme Visual Studio Code, on crée un fichier nommé dockerfile comme dans cet exemple :

Une image contenant texte

Description générée automatiquement

**Puis dans le terminal on crée l’image en lui donnant un nom**

Une image contenant texte

Description générée automatiquement

Comme on peut le voir, l’image est maintenant créée :

Une image contenant texte

Description générée automatiquement

On lance donc le conteneur :



**Test de dockerfile plus complexe**

**Le but est de créer une image debian pour mettre en place un serveur web avec apache.**

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

**J’ai donc cette arborescence : 4**



**Mon conteneur ne veut pas se lancer … je n’arrive pas à savoir pourquoi, il semblerait que cela vienne de ma machine.**

Maintenant, je vais créer une autre image, puis je vais lui donner un volume

Une image contenant texte

Description générée automatiquement

On voit donc grâce a *docker volume inspect* que notre volume data-test a été créé et que notre dossier data est à l’intérieur, comme stipulé dans le dockerfile.

Utilisation docker-compose

On crée un fichier docker-compose.yml

Une image contenant texte, portable, capture d’écran

Description générée automatiquement

Une image contenant texte, signe

Description générée automatiquement

On tape l’adresse dans la barre de navigation et on tome sur l’installation de notre wordpress.

Une image contenant texte

Description générée automatiquement

On peut ajouter une image sur le docker hub.

Il faut tout d’abord se connecter sur le docker hub et créer un document.

Une image contenant texte

Description générée automatiquement

On se connecte via le terminal pour se lier à notre compte:

Une image contenant texte

Description générée automatiquement

Je veux mettre sur le docker hub l’image qui s’appelle image\_test.

Une image contenant texte

Description générée automatiquement

On voit maintenant qu’elle est en ligne :

Une image contenant texte

Description générée automatiquement

**Projet Plex**



Tout d’abord nous partons de la distrib Ubuntu. 

On créer donc un fichier dockerfile qui va nous servir à créer notre image.

On copie l’adresse de téléchargement de l’iso.

Une image contenant texte

Description générée automatiquement

On met à jour la version, on installe wget, et on se place dans le dossier téléchargement pour lancer le téléchargement l’exécutable plex.

On test d’abord nos lignes en local via powershell.   
On a récupéré une image ubuntu, mis à jour, puis tenter d’installer les paquets nécessaires tels que :   
- dkpg  
- wget  
- gnung

Plex n’a pas l’air d’être compatible avec la dernière version d’ubuntu.  
On teste avec une version antérieure d’ubuntu, la 16.04. Mais nous avons encore une erreur comme on peut le voir ci-dessous

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Autre test :

Depart de l’image ubuntu 20.04.   
On commence tout simplement par mettre à jour : *sudo apt update && sudo apt upgrade -y*

On installe les paquets sudo, gnupg, gnupg2, gnupg1, curl …

*sudo apt install apt-transport-https curl -y*

On crée un fichier de référentiel extrait directement du référentiel Plex. Ce qui garantit que l’installation et la mise à jour sont fait à partir de la source officielle : *curl https://downloads.plex.tv/plex-keys/PlexSign.key | sudo apt-key add –*

On importe le référentiel :   
*echo deb https://downloads.plex.tv/repo/deb public main | sudo tee /etc/apt/sources.list.d/plexmediaserver.list*

On installe plex : sudo apt install plexmediaserver  
On souhaite connaitre le status : *systemctl status plexmediaserver*

Et là on a une erreur, comme si le service n’existe pas.

Comme on ne voyait pas d’où pouvait provenir l’erreur, nous avons essayé de faire ces mêmes commandes sur une machine virtuelle ubuntu sur vm ware. Cette version avait déjà tous les paquets de bases d’installé. Ce test nous permettait de savoir si le problème venait des paquets à installer non.

Etant donné que sur cette machine nous n’avons pas rencontré d’erreur, nous ne avons conclu que nous avions pas les bons paquets.   
Nous avons listé tous les services installés sur cette machine pour comparer avec la nôtre, mais il y’avait plus de 1787 lignes… Nous ne nous sommes donc pas attardés dessus.

Autre test :

Depart de l’image ubuntu 20.04.   
On commence tout simplement par mettre à jour : sudo apt update && sudo apt upgrade -y

On installe les paquets sudo, gnupg, gnupg2, gnupg1, curl …

En essayant également de trouver des commandes qui peuvent permettre d’installer tous les paquets de bases pour ressembler au maximum à notre version sous Vm Ware comme :

* Apt install gdebi
* sudo apt install build-essential

sudo apt install apt-transport-https curl -y

curl https://downloads.plex.tv/plex-keys/PlexSign.key | sudo apt-key add –

echo deb https://downloads.plex.tv/repo/deb public main | sudo tee /etc/apt/sources.list.d/plexmediaserver.list

On installe plex : sudo apt install plexmediaserver

Cependant le résultat est le même, nous bloquons sur la commande : systemctl status plexmediaserver qui nous retourne ce message :

*System has not been booted with systemd as init system (PID 1). Can't operate.*

*Failed to connect to bus: Host is down*