1 - être capable de créer un conteneur et de gérer des images (container) docker (pull, start, stop, rm, rmi)

La première étape est d’installer docker depuis docker desktop

Une image contenant texte

Description générée automatiquement

Ensuite créer le premier conteneur et le lancer grâce à la commande docker run -d -p 1000:1000 --name test -e MYSQL\_RANDOM\_ROOT\_PASSWORD=yes mysql:latest

Argument -d pour lancé le container en arrière-plan

Argument -p pour donner le port d’écoute

Argument --name pour donner au container, le nom désiré.

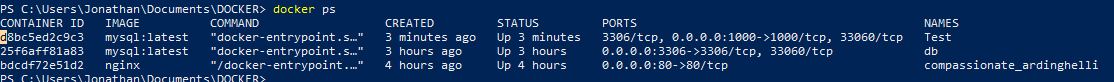
Argument -e pour donner la distribution donnée au container.

Le type de distribution:latest pour la version de la distrib

Dans l’exemple : Mon container s’appellera test.



On voit que le container a été créer avec sous ma ligne de commande sont ID unique.

Suite à la création, il est possible de vérifier que le container est lancé

Dans cette capture nous avons :

* La commande : docker ps qui permet de lister tous les containers de lancé
* Le Container ID qui est l’indentité de celui-ci, il est unique
* L’image installé sur le container : Ici Mysql en dernière version ou NGINX
* Depuis combien de temps la VM est créer, et si elle est lancée ou non et depuis combien de temps.
* Le port d’ecoute lié
* Et le nom du container.

Il est ensuite possible d’agir sur ses container la grâce à différente commande :

* Stopper et redémarrer le container : docker Start / Stop Nom\_du\_Container

****

À la suite de l’arrêt, le container n’est plus présent dans les containers lancés en arrière-plan

Pour relancé le Container : docker start Nom\_du\_container

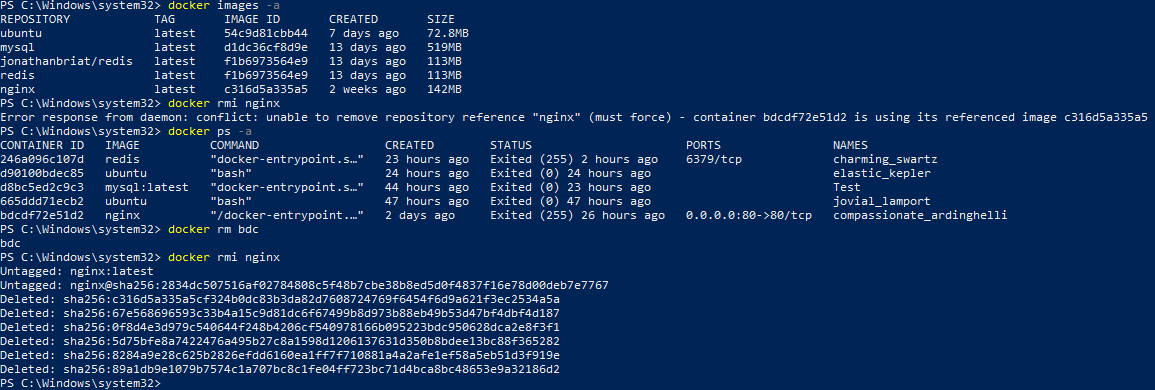
Pour la suppression complète d’un container : docker rm CONTAINER\_ID

Une image contenant texte

Description générée automatiquement

On voit ici que le container mysql :latest 25f n’est plus présent.

Pour supprimé une image : docker rmi NOM\_DE\_I’IMAGE



Impossible de supprimer une image si elle est en cours d’utilisation sur un container

Suppression du container en question et suppression de l’image.

Accéder au Shell d’un conteneur

Une image contenant texte

Description générée automatiquement

Le container doit être lancer

Il est important de mettre que l’on veut accéder au Bash après la commande docker exec -ti NOM\_DU\_CONTAINER /accès au bash

Pour avec le monitoring des container lancés

Avec docker stats, nous avons comme le gestionnaire de tache windows avec l’utilisation du CPU, la mémoire, le nom de la machine et son ID

Une image contenant table

Description générée automatiquement

Mettre en ligne sur hub une image docker.

1. Créer un compte sur docker HUB
2. Pousser l’image sur docker HUB depuis powershell (L’image que je pousse s’appelle redis)
3. Créer un lien entre l’image que nous avons en local et Docker Hub



Tag pour tagger le nom de l’image et la version

Jonathanbriat/…. Pour lié l’image a notre compte Hub précédemment créer .

1. Pousser le fichier/l’image sur le HUB Docker.



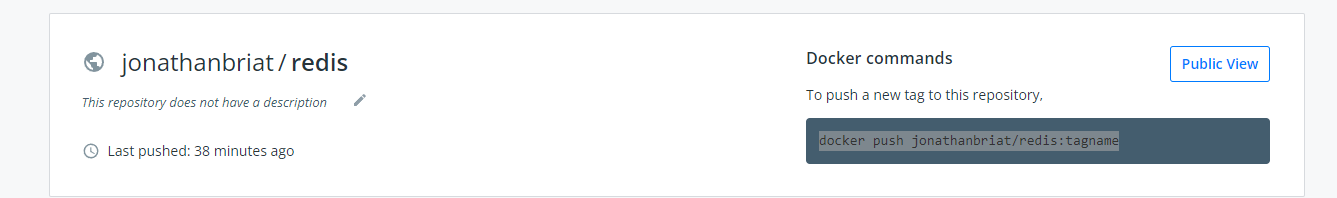
Push pour envoyer l’image en ligne.

Le rester pour savoir sur qu’elle compte envoyer le fichier.

1. Vérifier que l’image est bien uploader

Une image contenant texte

Description générée automatiquement



Création d’un docker files et création d’un container suivant cette image

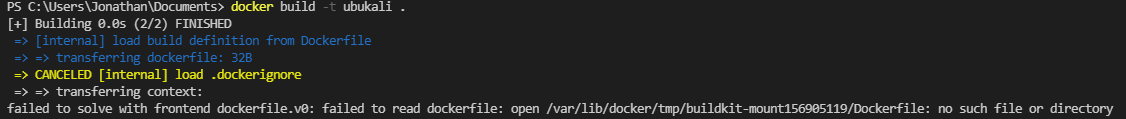
1. Création de l’image

Une image contenant texte

Description générée automatiquement

1. Construire l’image pour ensuite la renseigner

Docker build -t NOM\_DE\_L’IMAGE .



Erreur lors de la création : Il faut bien être dans le dossier ou ce situe le dockefile pour lancer la création.

Une image contenant texte

Description générée automatiquement

Vérification que la création de l’image a bien eu lieu avec une docker images

Une image contenant texte

Description générée automatiquement

Ubu kali est bien installer



Commande pour créer un nouveau container depuis l’image créer

--name pour donner un nom au container

--publish pour les ports d’écoute utilisé

Le nom de l’image

Création de Volume

1. Creation du volume

Il faut dans 1 premier temps, créer un volume vierge ici appeler : VolumeTest



Si on regarde sur Docker Desktop : Le nouveau volume est créer mais vide

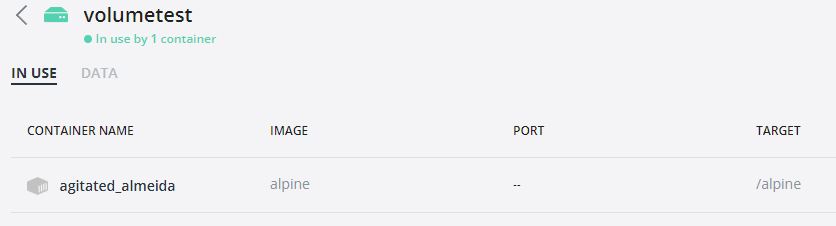
Une image contenant texte

Description générée automatiquement

Ensuite, créer dans ce volume, une image

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 :   
Pour celui-ci nous nous sommes inspirés de ces articles :

<https://fr.linuxcapable.com/how-to-install-plex-media-server-on-ubuntu-20-04/>

<https://www.delftstack.com/fr/howto/linux/how-to-install-plex-media-server-on-ubuntu/>

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’installés. 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 en avons conclu que nous n’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*

Une image contenant texte

Description générée automatiquement

Autre test :

On décide donc de procéder autrement, on ne va plus s’attarder sur ce systemctl qui ne fonctionne pas.   
On va créer notre dockerfile en précisant un port. On build notre image et si en se connectant à localhost : port un résultat s’affiche alors cela voudra dire que notre conteneur est bien en route.   
On décide dans un premier temps de partir d’une version antérieure d’Ubuntu. Une version 16.04 qui est une version stable.

Voici le début de notre dockerfile. On crée une première variable qui contient la version de Plex que nous souhaitons installer. On en crée un deuxième qui contient le lien de téléchargement de l’exécutable.

Ensuite on met à jour les paquets et on installe le paquet wget qui va nous servir pour télécharger plex grâce au lien qu’on a récupéré précédemment, comme inscrit à la ligne 16.

Une image contenant texte

Description générée automatiquement

Pour la suite, à la base nous voulions passer par l’instruction USER Plex, qui veut tout simplement dire d’utiliser l’utilisateur plex.   
Après un test voici l’erreur qui est retournée :



Suite à cette erreur, nous avons trouvé une autre instruction en passant par la création d’un groupe.   
Voici notre source : Une image contenant texte

Description générée automatiquement  
  
Adapté à notre projet elle donne :   
Une image contenant texte

Description générée automatiquement

On s’attaque ensuite à plex directement. Après de nombreuses recherches, nous avons récupéré quelques variables d’environnement qui semblent obligatoire au bon fonctionnement. Voici quelques pages qui nous ont aidé :

**Sources variables d’environnement :**

Une image contenant texte

Description générée automatiquement  
<https://qastack.fr/ubuntu/600026/problems-starting-plexmediaserver-on-kubuntu-15-04>

Une image contenant texte

Description générée automatiquement  
<https://hub.docker.com/layers/tianon/plex/1.18.4.2171-ac2afe5f8-media-server/images/sha256-253b6886997289b77bf90f5cd08a9a1196b8351edb520a0763c4fccef96d701a?context=explore>

La difficulté s’est magnifestée par le grand nombre de variable. Nous ne savions pas lesquelles garder.

Après quelques tests, voici notre sélection.

Par exemple voici une erreur en testant le chemin /usr/lib/plexmediaserver/Plex Media Server. On obtient une erreur car le chemin n’existe pas encore, c’est pour cela qu’il lui faut sa variable d’environnement.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Celles-ci sont présentes pour le bon fonctionnement du service.   
On voit notamment sur la dernière ligne qu’on ajoute l’arborescence au PATH pour que Plex puisse être exécuter sans poser problème.

On lui attribue un port, puis un volume, et enfin se place par défaut dans le volume qu’on a créée.

Une image contenant texte

Description générée automatiquement

Enfin, on lui donne l’instruction de lancer Plex.

Une image contenant texte

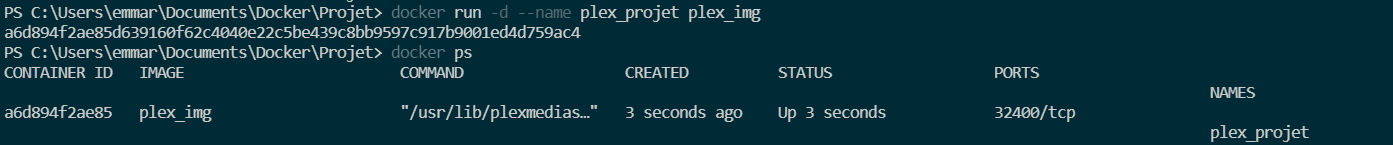
Description générée automatiquement

Ce qui nous donne le résultat suivant :

Une image contenant texte

Description générée automatiquement

On build, notre dockerfile et cela fonctionne.   
On lance notre conteneur :

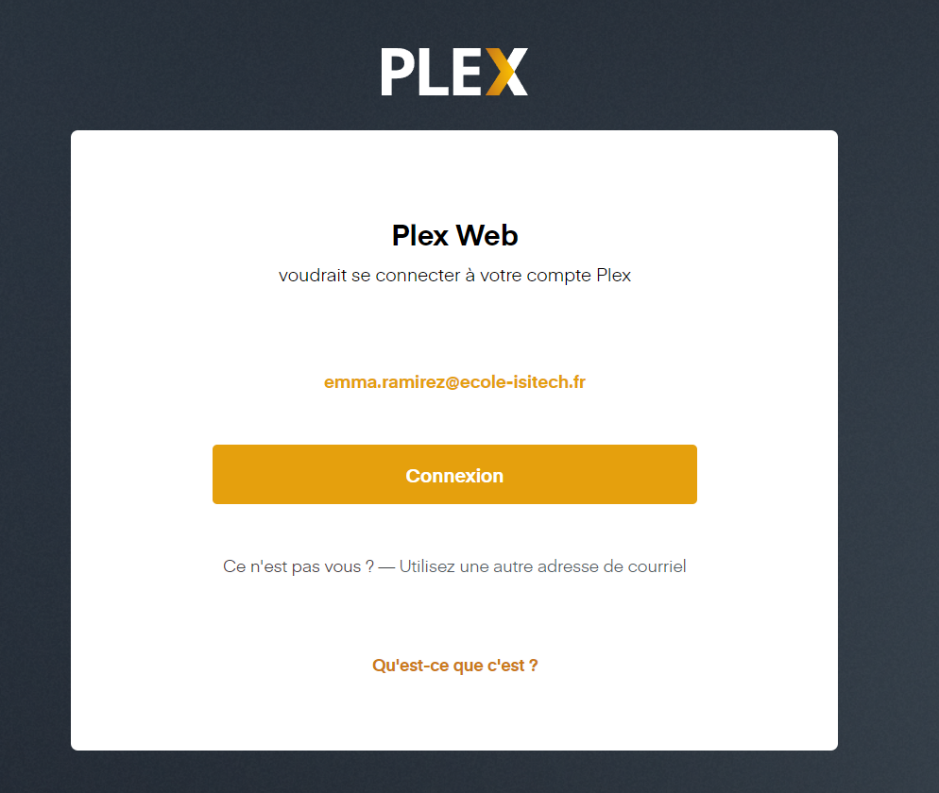
  
  
On tape l’adresse localhost :32400

Une image contenant texte

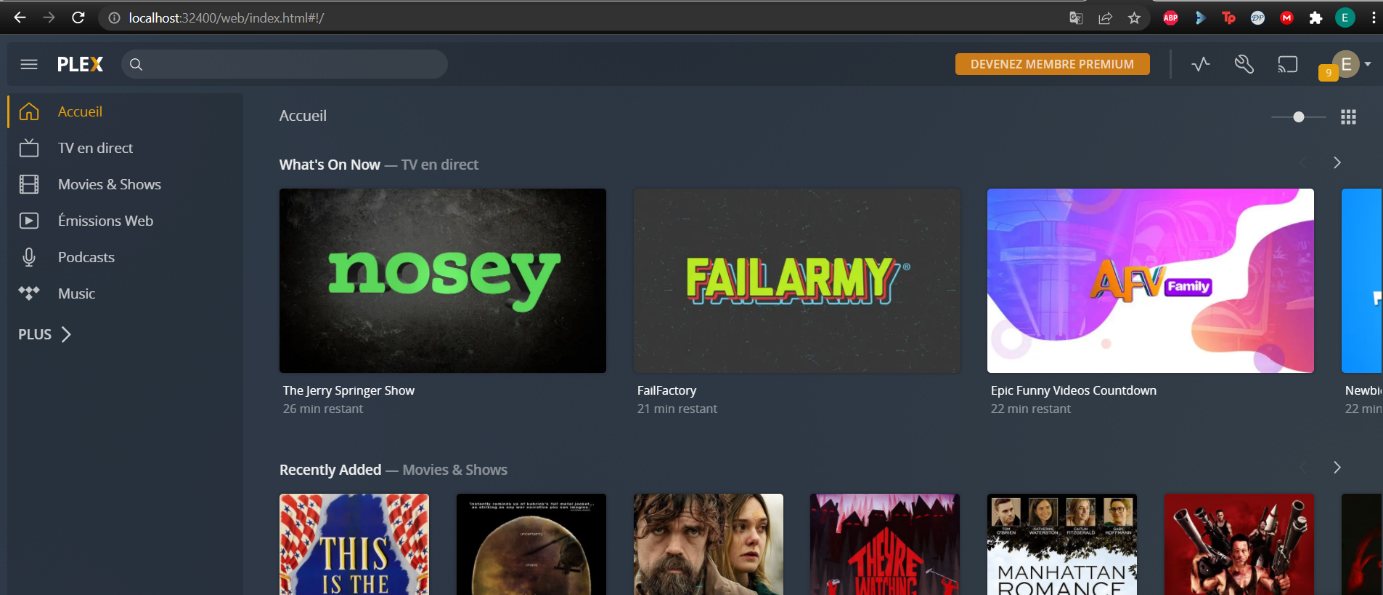
Description générée automatiquement

En recherchant l’erreur sur internet, il fallait en fait rajouter localhost :32400/**manage**

On arrive bien sur la page Plex.







On tente donc de modifier la version d’Ubuntu par une version plus récente, la 20.04. Celle-ci fonctionne également.

Voici une vision de tous les tests :



  
Une image contenant texte

Description générée automatiquement

**Docker-compose**

Une image contenant texte

Description générée automatiquement

Ici nous sommes partis de l’image plex-img créée précédemment pour lui apporter quelques modifications (sans toucher à l’image en elle-même).  
A la place de n’avoir qu’un seul volume, nous avons décider d’en créer 3 pour séparer les différentes données.   
Un volume pour la configuration, un volume pour les films et un volume pour la télévision.

On tape à nouveau localhost :32400/**manage**Et on tombe bien sur la page Plex.   
On peut observer en créant la bibliothèque nos 3 volumes stipulés dans le docker-compose

Une image contenant texte, moniteur, noir, capture d’écran

Description générée automatiquement