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.**

**Conteneur :** Un conteneur, est une instance qui exécute une image. Il permet l’isolation des services.

**Image :** Une image est comme un snapchot d’une machine. Elle s’exécute par le conteneur.

Les manipulations suivantes se feront sur un powershell.

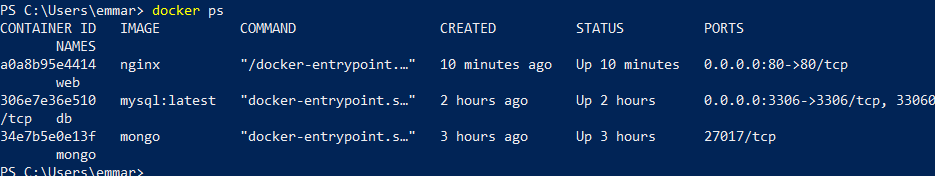
Création de conteneur :

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 :

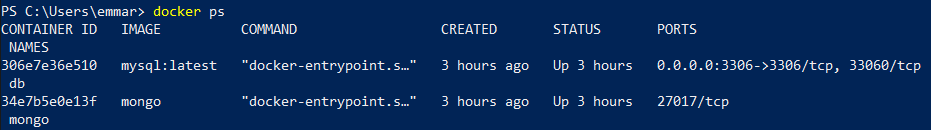


On aperçoit notre conteneur avec l’image nginx. En plus de voir ce qui s’exécute, il est également possible de voir les conteneurs arrêtés.

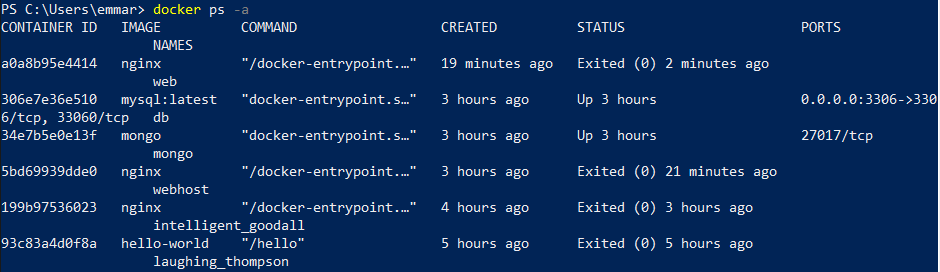
Nous allons donc arrêter notre conteneur :



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

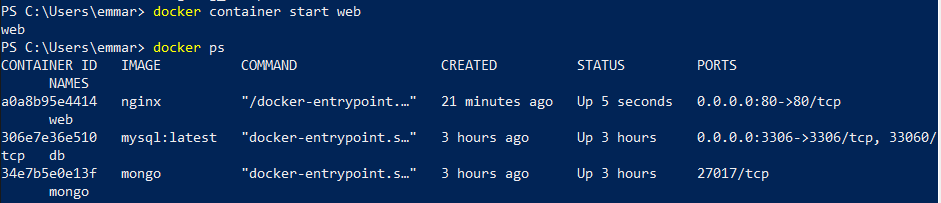


On voit donc ici qu’il n’est plus présent.



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 voit 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.

**Docker Hub**

**Le docker hub est une plateforme web qui contient toutes sortes d’images que l’on peut télécharger.**

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

Ici on récupère l’image alpine en version 3.14 et par une ligne de commande on veut faire s’afficher « Hello World ».

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 :



Mon conteneur ne veut pas se lancer … Je n’arrive pas à savoir pourquoi, il semblerait que cela vienne de ma machine car j’ai transmis les fichiers à quelqu’un de mon groupe pour qu’il teste, et cela a marché chez lui.

J’ai essayé plus tard d’en faire un nouveau. Moins complexe mais en utilisant seulement mes sources. Mon objectif était de créer une image Ubuntu avec le paquet vim et nginx d’installé.

Une image contenant texte

Description générée automatiquement

Dans un premier temps, je n’avais pas noté le « -y ». J’avais donc cette erreur au moment de build.

Une image contenant texte

Description générée automatiquement

J’ai ensuite testé de mettre en commentaire l’upgrade

Une image contenant texte

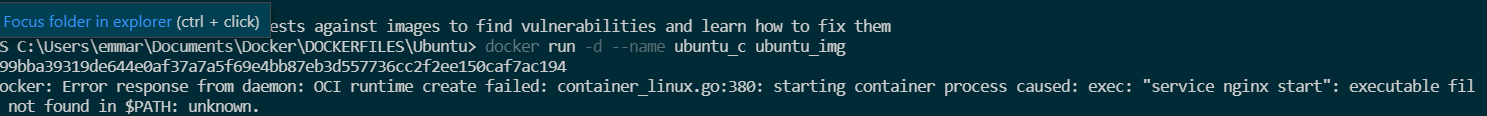
Description générée automatiquement

C’est à ce moment que je me suis rendu compte que dans une installation standard une confirmation était demandée. J’ai donc rajouté le -y, et le build a fonctionné :

Une image contenant texte

Description générée automatiquement

Je lance le conteneur et voici l’erreur :



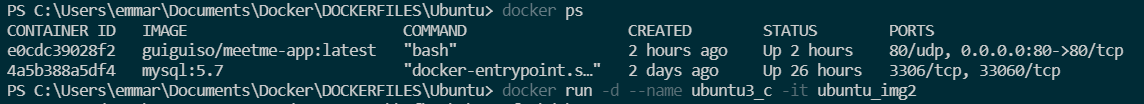
En effet la commande ne fait pas partie du PATH elle ne peut donc pas se lancer sans qu’on précise le chemin. J’ai modifié en précisant :   
Une image contenant texte

Description générée automatiquement

Je lance donc le conteneur et cette fois-ci cela fonctionne.



Cependant le conteneur ne tourne pas. Il ne s’affiche pas dans la liste docker ps.



Cela ressemble au problème que j’ai eu sur l’autre test que j’ai fait plus haut.

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 à *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.

**Docker-compose**

**Docker-compose est un outil qui permet de créer plusieurs conteneurs à partir d’images et de les lier entre eux en y ajoutant des paramètres. Cela se fait dans un fichier yaml.**

Dans cet exemple, nous allons créer à partir d’un docker-compose un site wordpress ainsi que sa base de données MYSQL.

On crée un fichier docker-compose.yml. Dans ce fichier l’indentation est extrêmement importante. Si elle n’est pas réalisée correctement le fichier sera en erreur.

Une image contenant texte, portable, capture d’écran

Description générée automatiquement

Ici on peut voir les 2 services db et wordpress. On leur affecte des volumes, des ports, ainsi que des variables d’environnement qui vont permettre par exemple de défini des mots de passe et des noms d’utilisateurs.

Maintenant on lance notre docker-compose.

Une image contenant texte, signe

Description générée automatiquement

Une fois le conteneur en route, on tape l’adresse dans la barre de navigation et on tombe sur l’installation de notre WordPress.

Une image contenant texte

Description générée automatiquement

**2ème Docker-compose**

J’ai tenté de créer un serveur plex.   
Pour cela il nous faut 2 services, un plex et un pour les certificats SSL.

On leur affecte image, ports, environnements, labels volumes et réseaux.  
Je pense que cela n’a pas fonctionné en parti car je n’avais pas d’image ubuntu

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

**Ajout d’une image sur Docker Hub**

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 :   
  
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

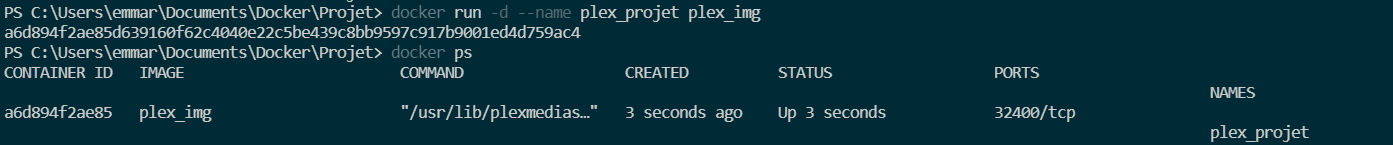
Description générée automatiquement

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 :

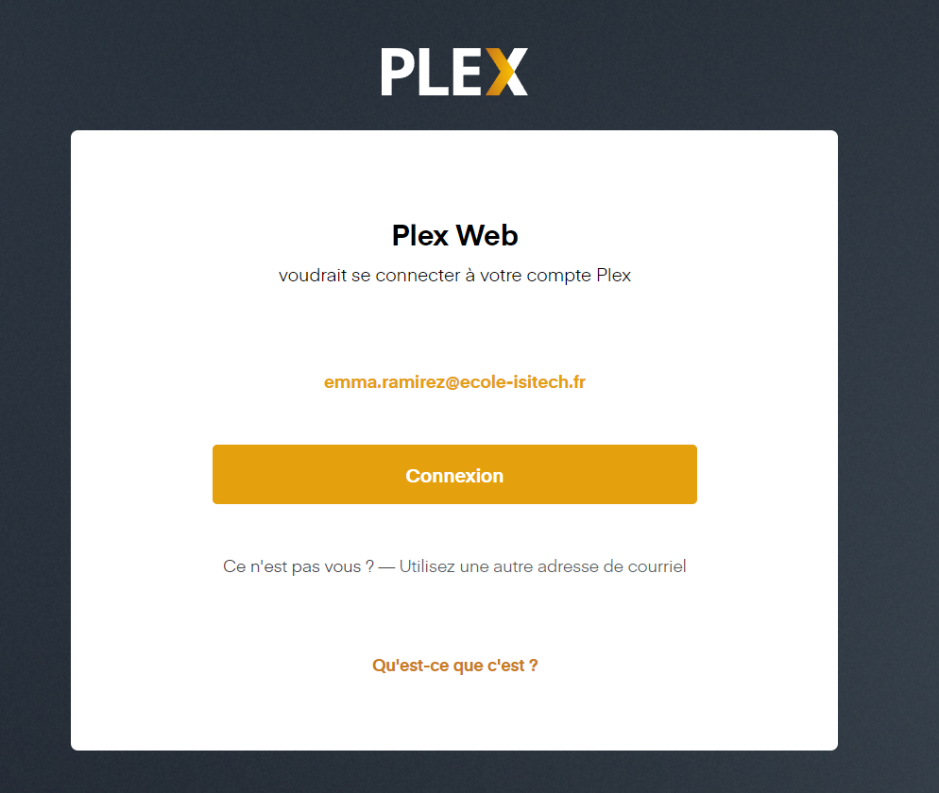
  
  
On tape l’adresse localhost :32400 et voici ce qu’on obtient :

Une image contenant texte

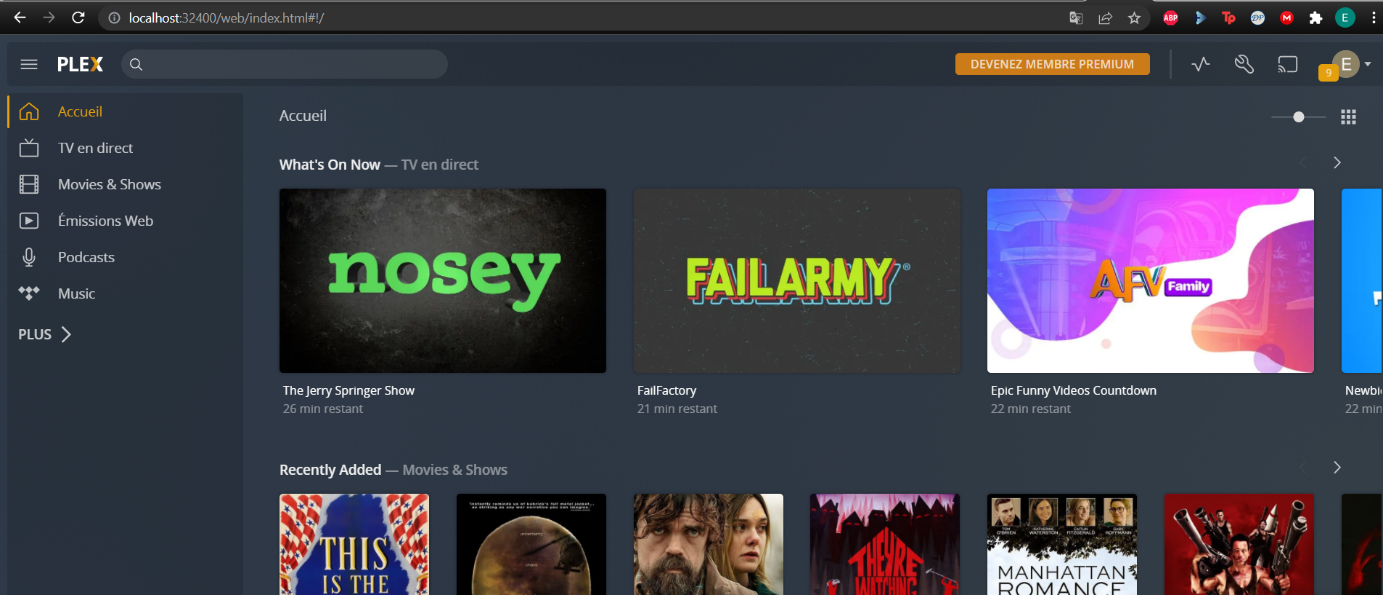
Description générée automatiquement

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 décidé 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