07 février 2022

Kerem COSKUN

Compte-Rendu b3 rpi

1. Projet Perso
2. Commandes pour gérer les conteneurs et les images

Docker permet créer, déployer et exécuter des conteneurs de manière efficace. Un [conteneur](https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/?gi=ab8480e95fe3) enveloppe l’application d’un logiciel dans une boîte invisible avec tout ce dont il a besoin pour s’exécuter. Ils sont surtout légers, portables et permettent aux développeurs de créer, déployer et exécuter efficacement des applications distribuées.

Pour créer un conteneur il faut taper la commande suivante :



Ici, nous créons un conteneur qui utilisera le port 80 ayant pour nom webhost nginx

Nous avons plusieurs commandes qui permettent de manipuler ces conteneurs :

La commande Docker PS qui permet de lister les conteneurs qui sont en train de tourner.



Docker container start [argument] ou l’argument est le nom du conteneur permet de démarrer ton conteneur.



Docker container stop [argument] permet de d’arrêter ton conteneur.



Docker rm [argument] supprime le conteneur serene\_hypatia.



Docker rmi [Argument] permet de supprimer une image.



1. Variables d’environnement



1. Monitoring des conteneurs.

Docker top [argument] permet d’afficher les processus en cours du conteneur indiqué

Une image contenant texte, bleu

Description générée automatiquement

Docker inspect [argument] est une commande permet d’avoir les informations bruts du docker, on peut notamment y trouver les paramètres réseaux.

Une image contenant texte

Description générée automatiquement

Docker logs [argument] permet de voir les journaux de connexions et d’exécutions du conteneur indiqué.

Une image contenant texte

Description générée automatiquement

Docker exec -it [Argument] /bin/bash permet d’accéder à la commande shell du conteneur.



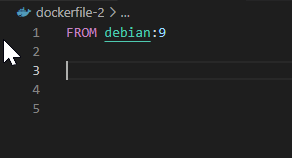
1. Création d’un dockerfile :

Pour créer un dockerfile il suffit d’ouvrir un dossier sur Visual Studio Code et cliquer de créer un fichier et le nommer dockerfile

Une image contenant texte

Description générée automatiquement

Une fois celui-ci créer, commencez par indiquer l’image et la version que vous voulez utiliser.



Une fois ceci fait, sauvegardez à l’aide des raccourcis « CTRL + S »

Vous verrez une petite balène bleu à côté de votre nom de fichier comme sur le screen ci-dessus.

1. Création d’un volume :

Nous avons créé le volume volume-test dans le dossier data de l’image ubuntu.



Pour inspecter le volume créer faire la commande suivante :

Une image contenant texte

Description générée automatiquement

Le volume volume-test a bien été créé dans le dossier data d’ubuntu.

1. Sauvegarder dockerfile sur dockerhub

Pour sauvegarder le dockerfile, il faudrait que nous créions notre image personnalisée en lançant la commande docker build -t [lenomdetonimage] [choisir le dockerfile].

Une image contenant texte

Description générée automatiquement

Une fois ceci fait lorsque l’on fait « docker images » nous constatons que notre image est créée.

Une image contenant texte

Description générée automatiquement

Il faudrait ensuite créer un compte sur docker hub puis créer un répertoire.

1. Cliquer sur Repositories
2. Créer le répertoire
3. Le chemin pour déposer le répertoire ici sera vastlordkerem/devoir



Une fois ceci fait, il faudrait créer le tag avec la commande suivante :

Le dockerfile :latest sera le nom de notre depot et latest le tag puis vastlordkerem/devoir sera le chemin du dépôt avec le tag latest.



Puis on le pousse avec la commande suivante.



Une fois ceci fait, nous devrions avoir dans notre dépôt de docker Hub ceci :

Une image contenant texte

Description générée automatiquement

1. Pull notre image personnalisé

On va essayer de pull notre image publié sur docker hub, notre chemin est « vastlordkerem/devoir »

En fesant « docker images » nous constatons que nous n’avons pas encore l’image.

Une image contenant texte

Description générée automatiquement

On pull l’image du docker hub en tapant la commande suivante :

Une image contenant texte

Description générée automatiquement

Le fait qu’il indique Already exists est tout à fait normal, on peut voir que le status est OK.

Lorsque nous faisons un « docker images » de nouveau, nous constatons que l’image est bien là

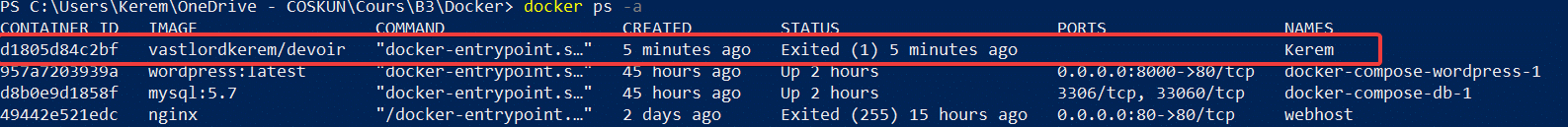
Une image contenant texte

Description générée automatiquement

Nous allons maintenant créer notre conteneur à partir de notre image personnalisée en faisant la commande suivante.



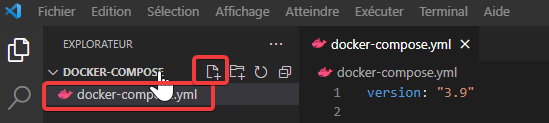
Une fois avoir fait cette manipulation, nous constatons que notre conteneur a bien été créer à partir de notre image vastlordkerem/devoir.



1. Docker-compose

Docker-Compose est un outil qui permet d’associer plusieurs conteneurs ainsi que des paramètres pour y créer toute une configuration personnalisée.

Il suffit de créer une fichier suivi de « Docker-compose.yml » l’extention étant très importante, vous y verrez une petite baleine rose.



Ainsi, nous avons créer notre docker-compose il nous faut maintenant y associer les paramètres.

Une image contenant texte

Description générée automatiquement

Nous pouvons aussi le personnaliser à souhait, ci-dessus est un serveur SQL suivi d’un wordpress ( création d’un site )

Ci-dessous il s’agit d’un serveur plex, serveur multimédia que j’ai concocté moi-même.

Une image contenant texte

Description générée automatiquement

1. Projet PLEX

**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

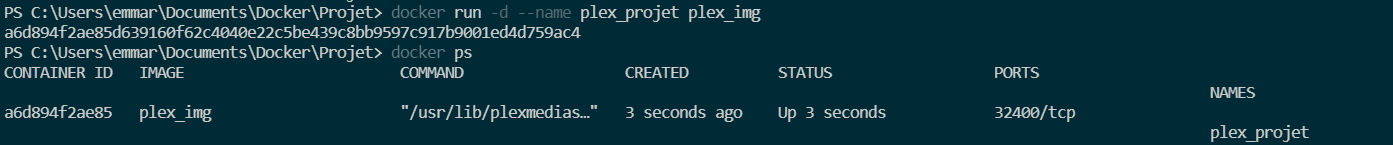
Description générée automatiquement

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 :

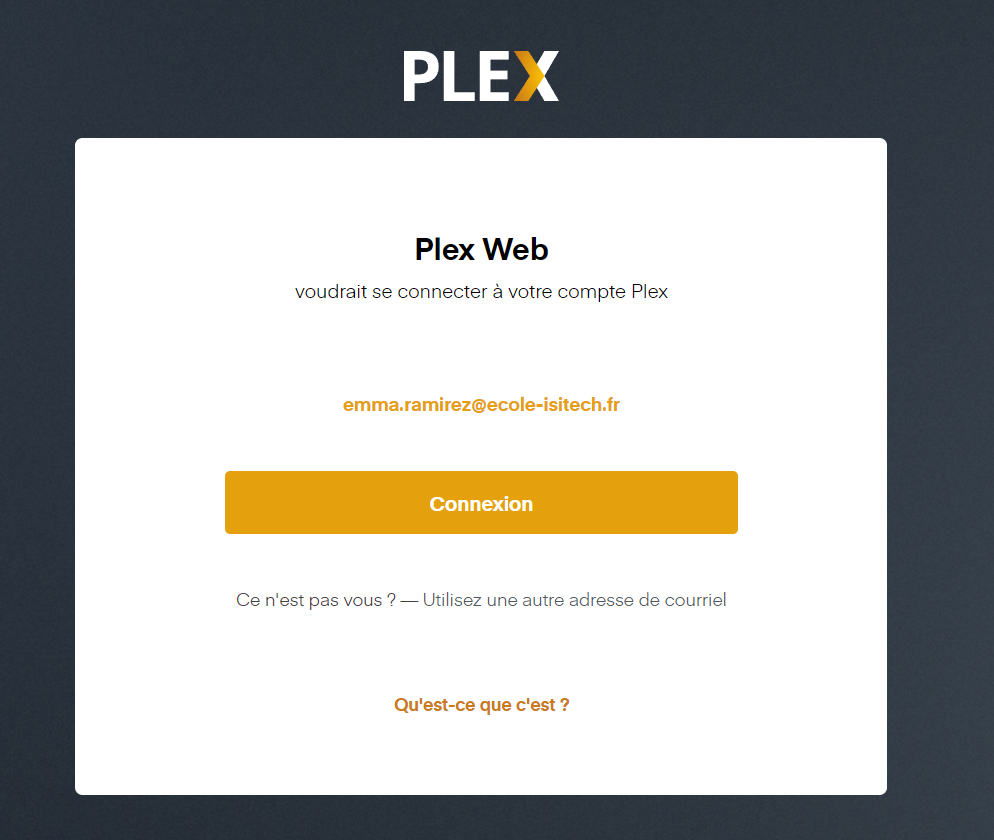
  
  
On tape l’adresse localhost :32400

Une image contenant texte

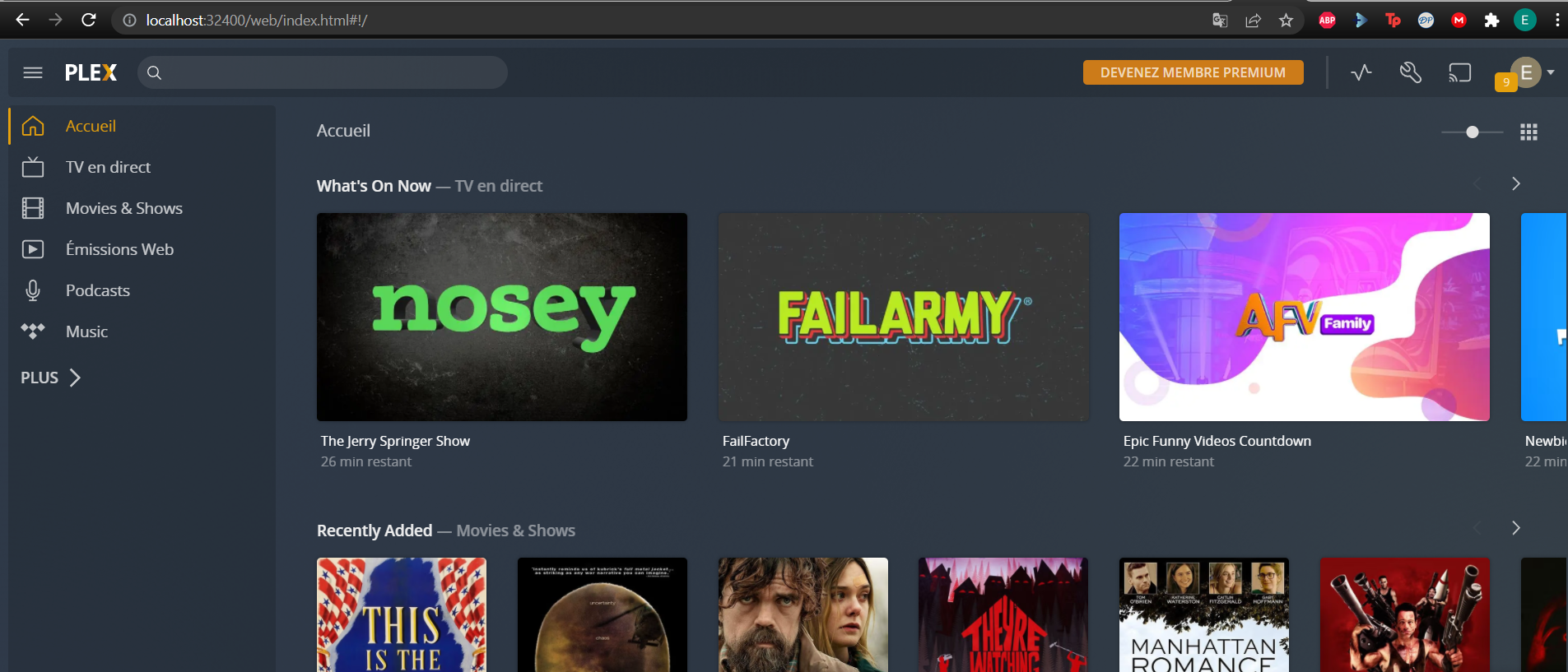
Description générée automatiquement

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

Ce qui conclut l’installation et le fonctionnement de Plex sur Docker à partir d’une image Ubuntu.