

24/10/2018

Léo Guilpain – ESIR 3 - IoT

Docker – TD1

*« J’atteste que ce travail est original, qu’il indique de façon appropriée tous les emprunts, et qu’il fait référence de façon appropriée à chaque source utilisée »*

Table des matières

[Introduction 2](#_Toc528073942)

[Partie 1 : Initiation 2](#_Toc528073943)

[Question 01 2](#_Toc528073944)

[Question 02 2](#_Toc528073945)

[Question 03 3](#_Toc528073946)

[Question 04 3](#_Toc528073947)

[Question 05 4](#_Toc528073948)

[Question 06 et 07 4](#_Toc528073949)

[Question 08 4](#_Toc528073950)

[Question 09 4](#_Toc528073951)

[Question 10 5](#_Toc528073952)

[Question 11 5](#_Toc528073953)

[Partie 2 : Personnalisation d’une image 5](#_Toc528073954)

[Montage de volume 5](#_Toc528073955)

[Question 01 5](#_Toc528073956)

[Question 02 6](#_Toc528073957)

[Question 03 6](#_Toc528073958)

[Question 04 6](#_Toc528073959)

[Question 05 7](#_Toc528073960)

[Question 06 7](#_Toc528073961)

[Question 07 7](#_Toc528073962)

[Création d’une image 7](#_Toc528073963)

[Question 09 7](#_Toc528073964)

[Question 10 8](#_Toc528073965)

[Question 11 8](#_Toc528073966)

[Question 12 et 13 8](#_Toc528073967)

[Question 14 8](#_Toc528073968)

[Question 15 10](#_Toc528073969)

[Question 16 11](#_Toc528073970)

[Question 17 13](#_Toc528073971)

[Conclusion 13](#_Toc528073972)

Introduction

Le but de ce TP est de découvrir le fonctionnement de docker. Docker permet de gérer un ensemble de conteneurs. Nous allons donc également découvrir le fonctionnement des conteneurs.

Partie 1 : Initiation

Question 01

Je ne possédais pas docker-machine, j’ai donc dû l’installer.

Question 02

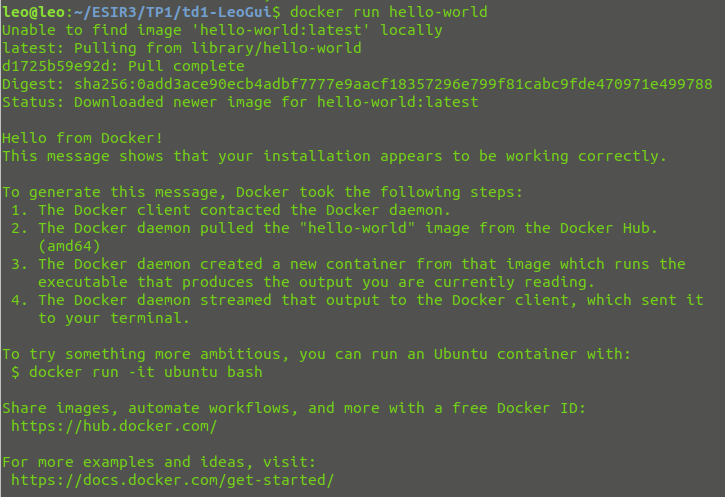


Figure 1 : hello world

Dans un premier temps, on récupère l’image « hello world ». Une fois l’installation terminée, un message s’affiche. Pour générer ce message, il a fallu :

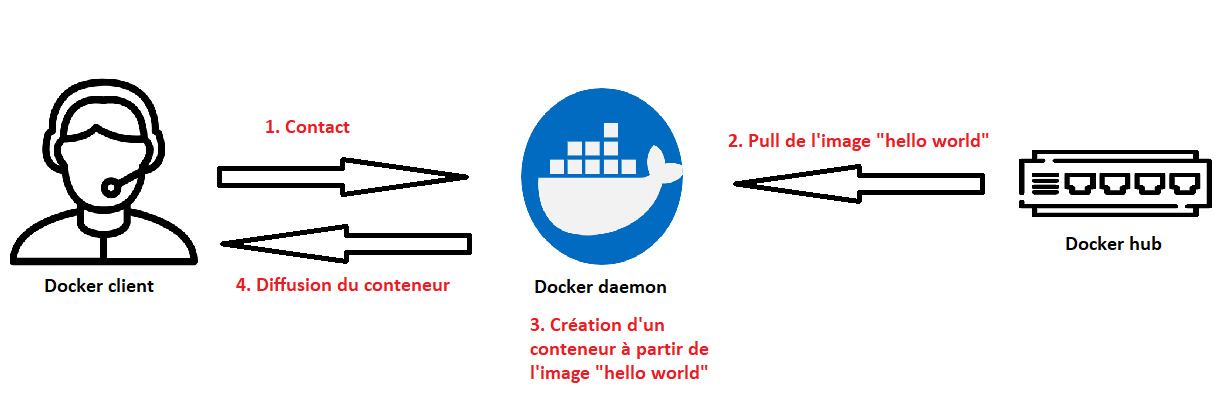


Figure 2 : Schéma du fonctionnement

Question 03

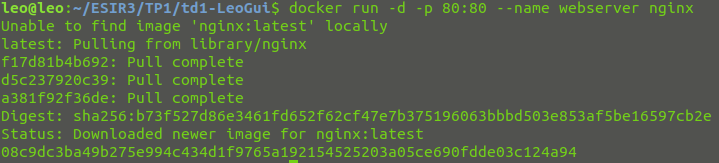


Figure 3 : Lancement du serveur

Dans un premier temps, on essaye de lancer un serveur web à partir de l’image « nginx ». Or, cette dernière n’est pas présente sur le répertoire local donc on fait un “pull” de l’image « nginx » pour la récupérer et la télécharger. Une fois l’image récupérée, le serveur web que l’on a nommé “webserver” est lancé à l'intérieur du conteneur.

Question 04

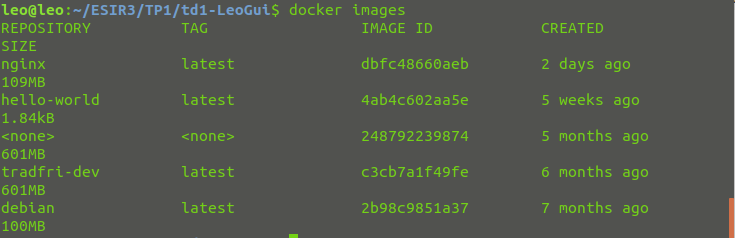


Figure 4 : Images docker

L’image « nginx » que l’on vient de récupérer est bien présente. De plus, on peut voir que d’ancienne images sont présentes, cela provient d’utilisations passées de Docker. Par exemple, tradfri-dev m’a servi pour allumer la lampe Ikea.

Question 05



Figure 5 : Recherche de l'image node

La première image et l’image officielle que l’on doit récupérer. On peut également voir grâce aux « stars » que c’est l’image qui a été la plus téléchargée.

Question 06 et 07

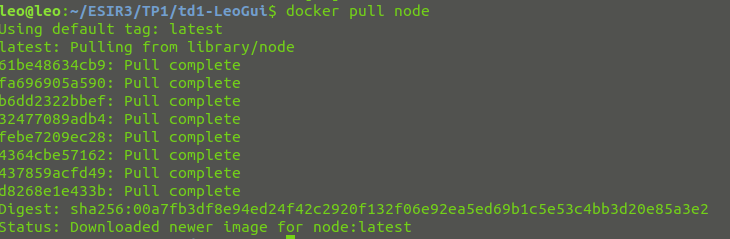


Figure 6 : docker pull node

Idem que pour nginx, il a fallu faire un “pull” pour pouvoir récupérer la dernière image de node.

Question 08

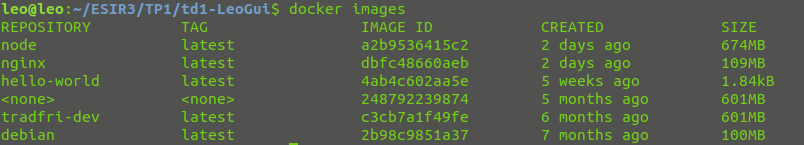


Figure 7 : docker images

On peut voir que l’image node et l’image nginx sont bien présentes sur l’ordinateur à la suite des manipulations précédentes.

Question 09

https://lh6.googleusercontent.com/wNYcmv5pNF57zqXhb5wTxUjlJ_so5kSnDDJPFXhDhUCe7VCSMvouD9AxYThw2WIv59tzYlETtAJ5H1ODigLsLDOgkr4g8xT5puFvD-7NYmkq1tTYKMMMjPhHqQ1ieBDlJDGUato

Figure 8 : docker ps

Le seul conteneur qui s'exécute en tâche de fond sur le port 80 est le webserveur. Ce dernier s'exécute contrairement à node puisque nous avons utilisé la commande “docker run … nginx”. Pour que node s'exécute en tâche de fond il faudrait faire la même commande en remplaçant “nginx” par “node”.

Question 10

https://lh6.googleusercontent.com/doERJ7aBpWdvNSKDArj1SRJF4L6BoE67-aU92aoeDFOHBJYyaWeuujpzfgNKV-GnUM0dLlLEUwFilRrv_F6eUSpO1KGyXIxsIvgeCnUqw5eN4hO-E7gFxYp1iz7UqMyX9hFliAk

Figure 9 : docker stop

Le conteneur a bien été stoppé ici.

Question 11

**Docker rm :** Permet de supprimer un ou plusieurs conteneurs

**Docker rmi :** Permet de supprimer une ou plusieurs images

Partie 2 : Personnalisation d’une image

Montage de volume

Question 01

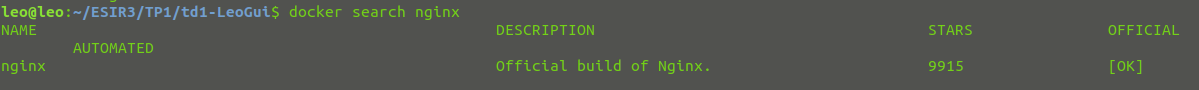


Figure 10 : docker search nginx

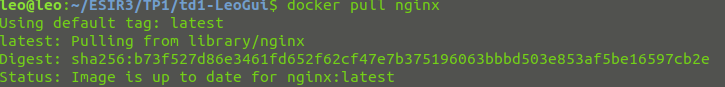


Figure 11 : docker nginx

Idem que pour les questions précédentes. On récupère la dernière image de nginx. Cette commande suffit car la dernière image est en version 1.15.5

Question 02

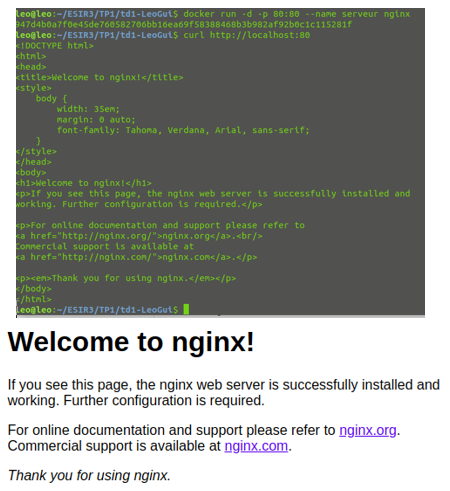


Figure 12 : run serveur nginx

En effectuant ces commandes, on peut voir que le conteneur est bien lancé.

Question 03

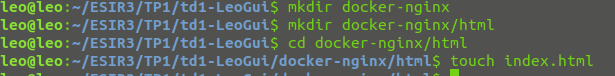


Figure 13 : Création des dossiers

Grâce à ces commandes, nous avons créé les dossiers et les fichiers requis.

Question 04

https://lh4.googleusercontent.com/v-eSuGRQiP1rgeShQcuv1dgeqq59p7vWo0zZ0qF0yDAKmRQgxonO3fC81WcgATTFzc_MHIIWCC8SOmkZ-MHocksGZP6KliCDD7NuxOIQvZeeU047q3I9KEDHYODCTMyZUgyLY1E

Figure 14 : Notion de volume

On a ici créé un lien entre l'espace de stockage de l'hôte, et celui du conteneur.

Question 05

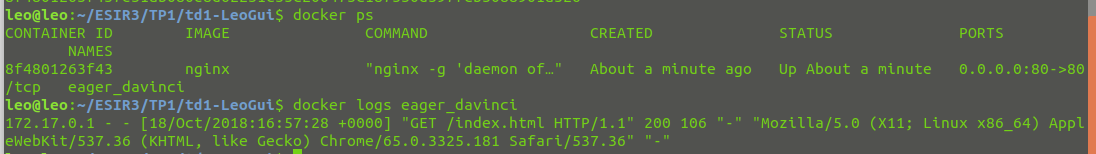
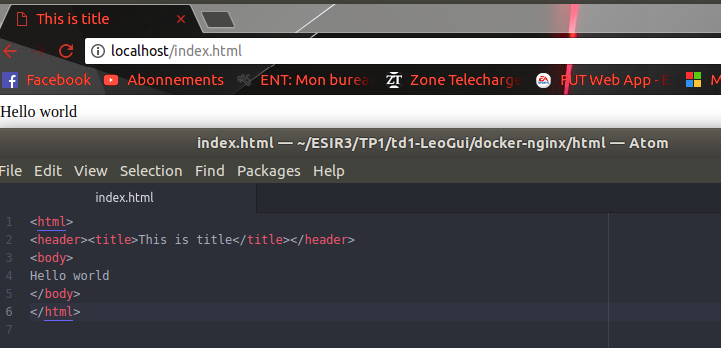


Figure 15 : docker logs

On récupère le nom du conteneur pour pouvoir ensuite observer les logs.

Question 06



On voit bien que le serveur est fonctionnel et il peut être modifié en temps réel.

Question 07

On ne peut pas se connecter correctement au conteneur car c’est l’hôte. Or les conteneurs ne peuvent pas interagir avec l’hôte.

Création d’une image

Question 09

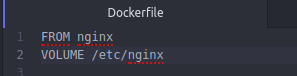


Figure 16 : Dockerfile

Après avoir créé le fichier « Dockerfile » à l’aide de la commande « touch », on le rempli comme ceci.

Question 10

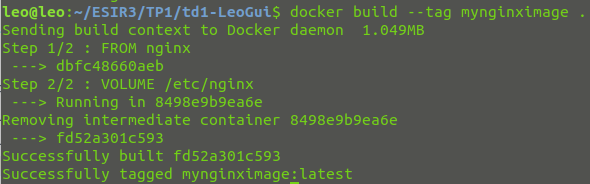


Figure 17 : docker build

Grâce à cette commande, on a « build » l’image grâce au Dockerfile.

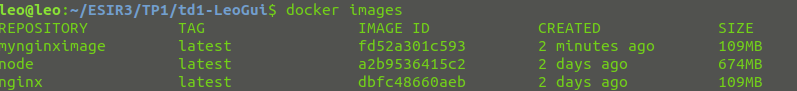


Figure 18 : image

L’image a bien été créée.

Question 11

https://lh5.googleusercontent.com/Oh_7PMu4LaM2PYvXcy1UIhE0wHvjxZMMAEs6K7KCjzkt0iSOlJTZ5xXVJyU09TDkrrGRJiQx_T32jN9nypRURCz-5JQpILY4MouRGP_D0n9zo-BIuCNrzQrZrXR9VwKJO8RT0WI

Figure 19 : docker run

A l’aide de cette commande, on a exécuté l’image « nginx2 » à partir de l’image « mynginximage ».

Question 12 et 13

https://lh4.googleusercontent.com/8VO8Z55NuiBoFAZLEZXmQ5-NYuHUAD7hAPejMuL8G305AmQgcV3xVSNwJLJIpvKhH9tuyI5zwSywcxkYaRfMYYKazouFfSo_2DecxZTmm2f-2dPWELvIm00SM51fu03svH0I5rE

Figure 20 : docker run

Après avoir exécuté la commande ci-dessous, nous avons accès à une console avec les droits root.

On a créé un conteneur debian pour pouvoir ainsi accéder aux différents fichiers de nginx2. Notamment les fichiers « default.conf ». Le choix était possible entre télécharger vim pour le modifier directement depuis le shell. J’ai décidé de le copier en local pour le modifier.

Question 14



Figure 21 : Répertoire conteneur

En naviguant dans les répertoires et en effectuant la commande « cat », on a pu trouver ce qu’il y avait dans le fichier « default.conf ». Comme expliqué précédemment, ce fichier va nous permettre d’avoir toutes les configurations nécessaires à ajouter dans notre fichier « default.conf ».

De plus, cela nous a permis d’observer et de comprendre ce qu’il y avait dans le fichier « nginx.conf ».

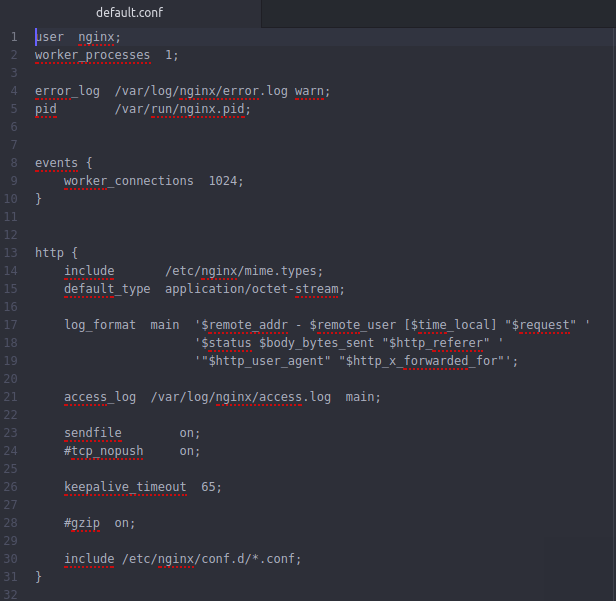


Figure 22 : nginx.conf

Question 15

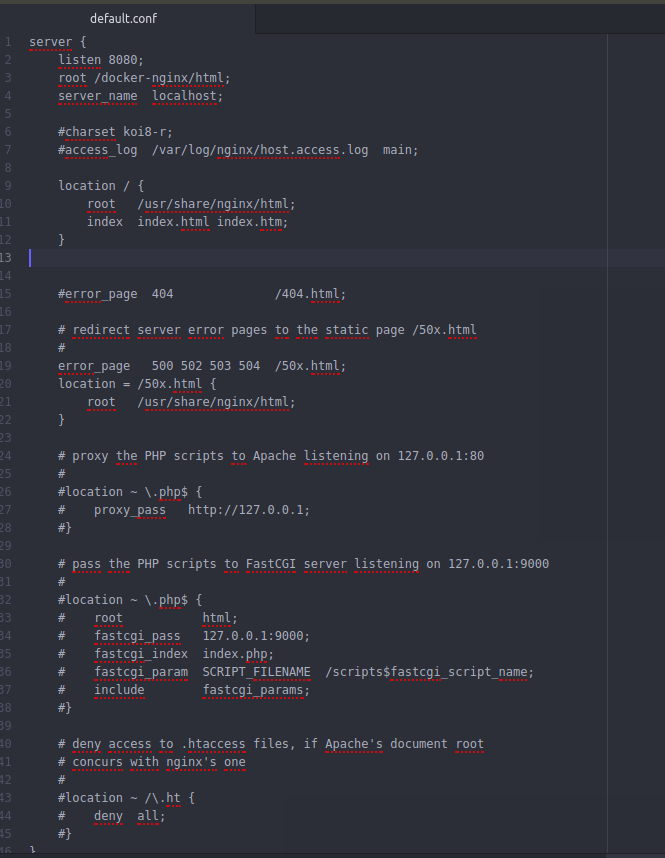


Figure 23 : default.conf

Après avoir créé le fichier « default.conf », nous avons copié l’intégralité du fichier et donc nous obtenons les informations suivantes.

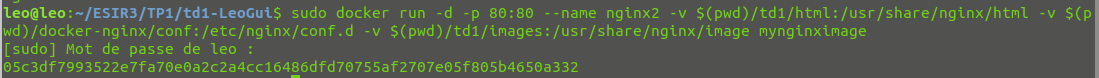
Question 16

Nous avons modifié le fichier « default.conf » comme ci-dessous :



Figure 24 : default.conf

Nous avons donc créé un serveur écoutant sur le port 80 et nommé “localhost”. Comme demandé dans l’énoncé, nous avons créé un proxy-pass à l’adresse “<http://localhost:8080>”. De plus, si dan l’url nous rajoutons l’extension .gif ou .jpg ou .png, alors une redirection est faite vers le dossier image.



Pour lier les informations, il est nécessaire de monter 3 volumes : le dossier html, le dossier image et le fichier default.conf

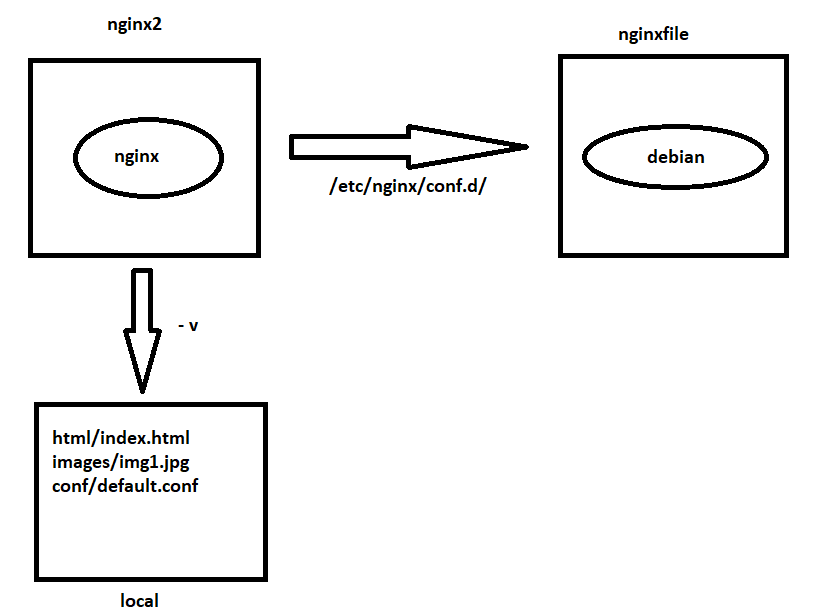


Figure 25 : Explication du montage

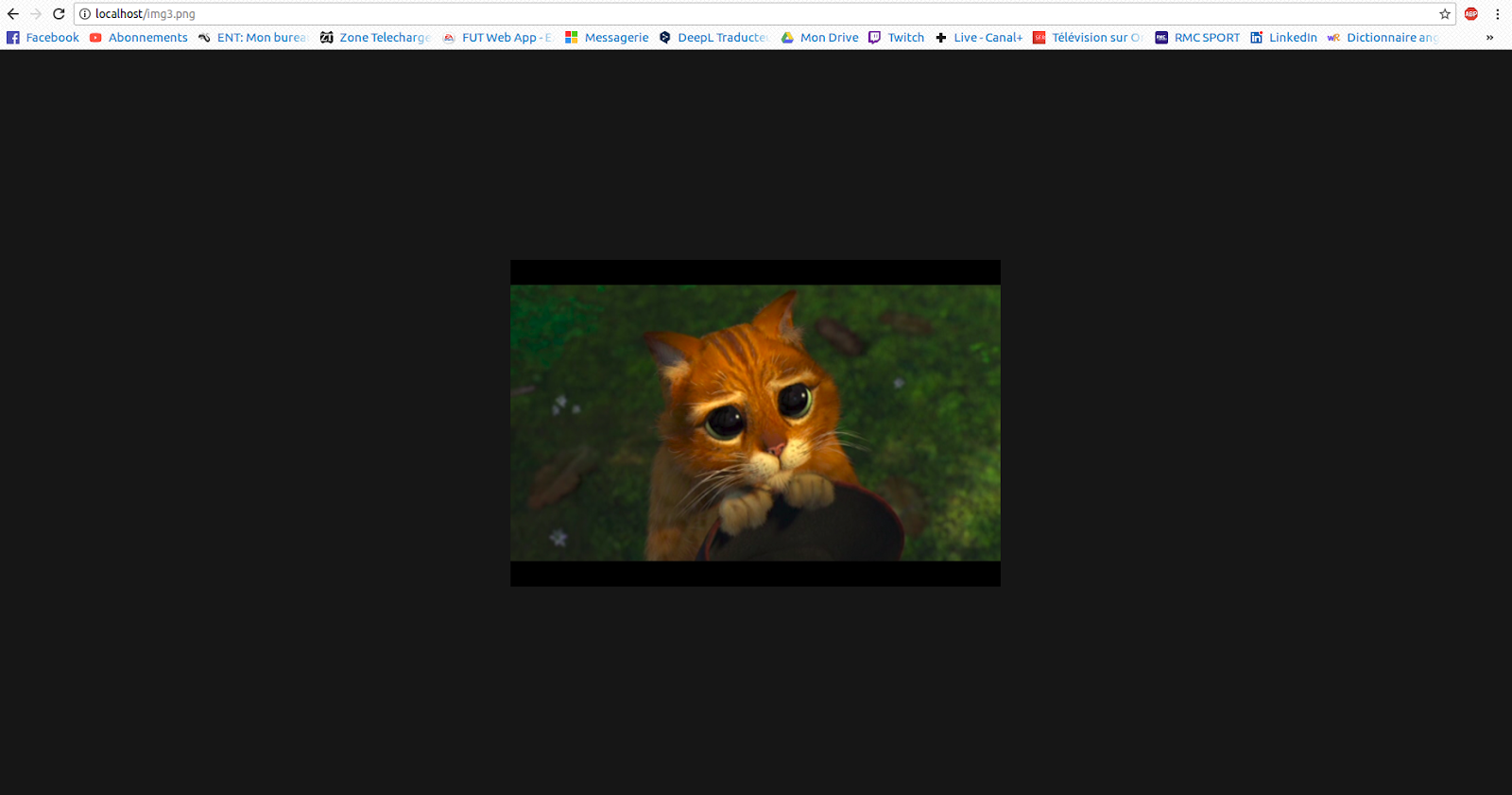


Figure 26 : img1.jpg

Lorsque dans la barre de recherche nous tapons « localhost:80/img1.png », nous obtenons ceci. Cela signifie donc que le volume a bien été monté correctement.

Question 17

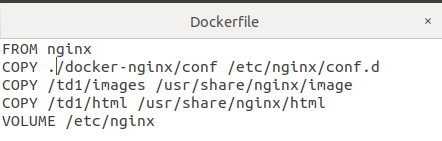


Figure 27 : Dockerfile

Via l’image nginx, on va créer la nouvelle image « nginxcustom »

Dans un premier temps on lance le fichier de configuration → default.conf

Puis on copie les images et l’index.html.

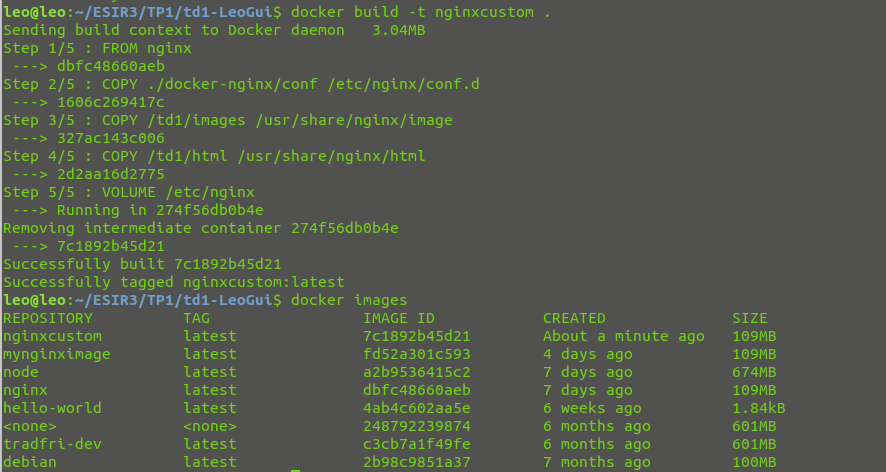


Figure 28 : docker build puis docker images

En effectuant la commande “docker build” on créer la nouvelle image puis on vérifie que l’image a bien été créée.

Conclusion

Dans ce TP sur Docker, nous avons donc pu découvrir son fonctionnement. Nous avons appris à gérer les images et les conteneurs. Après ceci, nous avons été capable de créer notre propre image.