UNIVERSITÉ DE RENNES 1

ESIR ECOLE SUPERIEURE D'INGENIEURS DE RENNES

24/10/2018

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 »*

Léo Guilpain – ESIR 3 - IoT

Table des matières

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

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 :

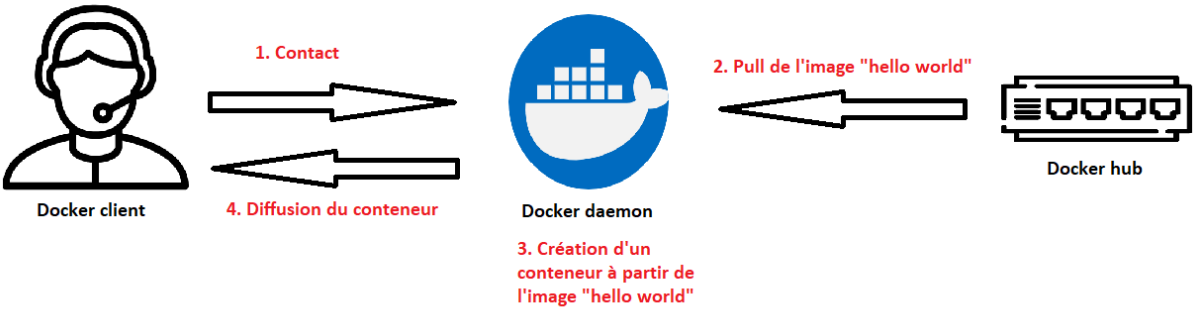


Figure 2 : Schéma du fonctionnement

Question 03

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker run -d -p 80:80 --name webserver nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
f17d81b4b692: Pull complete
d5c237920c39: Pull complete
a381f92f36de: Pull complete
Digest: sha256:b73f527d86e3461fd652f62cf47e7b375196063bbbd503e853af5be16597cb2e
Status: Downloaded newer image for nginx:latest
08c9dc3ba49b275e994c434d1f9765a192154525203a05ce690fdde03c124a94
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
nginx               latest              dbfc48660aeb        2 days ago
109MB
hello-world         latest              4ab4c602aa5e        5 weeks ago
1.84kB
<none>              <none>              248792239874        5 months ago
601MB
tradfri-dev         latest              c3cb7a1f49fe        6 months ago
601MB
debian              latest              2b98c9851a37        7 months ago
100MB
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker search node
NAME                              DESCRIPTION                                     STARS               OFFICIAL            AUTOMATED
node                              Node.js is a JavaScript-based platform for s…   6376                [OK]
mhart/alpine-node                 Minimal Node.js built on Alpine Linux           392
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker pull node
Using default tag: latest
latest: Pulling from library/node
61be48634cb9: Pull complete
fa696905a590: Pull complete
b6dd2322bbef: Pull complete
32477089adb4: Pull complete
febe7209ec28: Pull complete
4364cbe57162: Pull complete
437859acfd49: Pull complete
d8268e1e433b: Pull complete
Digest: sha256:00a7fb3df8e94ed24f42c2920f132f06e92ea5ed69b1c5e53c4bb3d20e85a3e2
Status: Downloaded newer image for node:latest
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
node                latest              a2b9536415c2        2 days ago          674MB
nginx               latest              dbfc48660aeb        2 days ago          109MB
hello-world         latest              4ab4c602aa5e        5 weeks ago         1.84kB
<none>              <none>              248792239874        5 months ago        601MB
tradfri-dev         latest              c3cb7a1f49fe        6 months ago        601MB
debian              latest              2b98c9851a37        7 months ago        100MB
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
08c9dc3ba49b        nginx               "nginx -g 'daemon of…"   7 minutes ago       Up 7 minutes        0.0.0.0:80->80/tcp  webserver
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker stop webserver
webserver
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker search nginx
NAME                                                DESCRIPTION                                     STARS               OFFICIAL
AUTOMATED
nginx                                               Official build of Nginx.                        9915                [OK]
```

Figure 10 : docker search nginx

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:b73f527d86e3461fd652f62cf47e7b375196063bbbd503e853af5be16597cb2e
Status: Image is up to date for nginx:latest
```

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

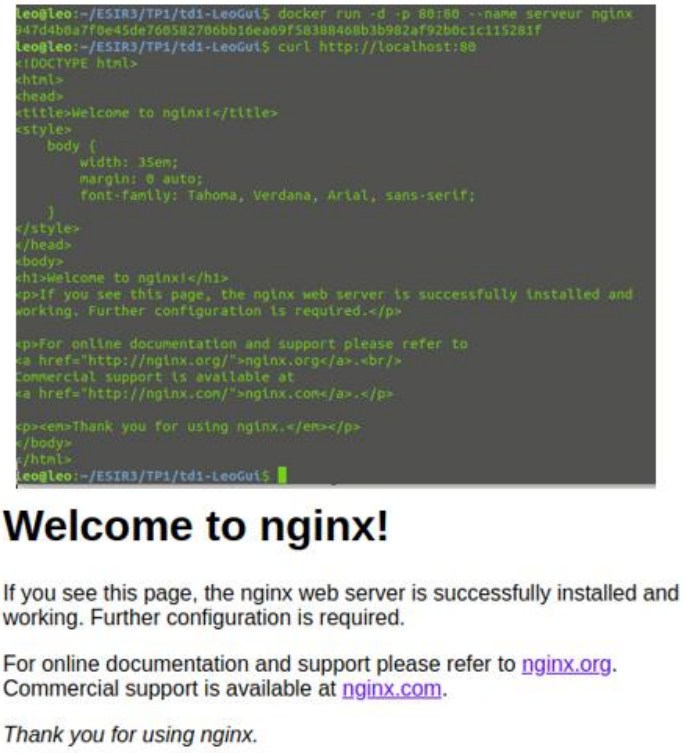


Figure 12 : run serveur nginx

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

Question 03

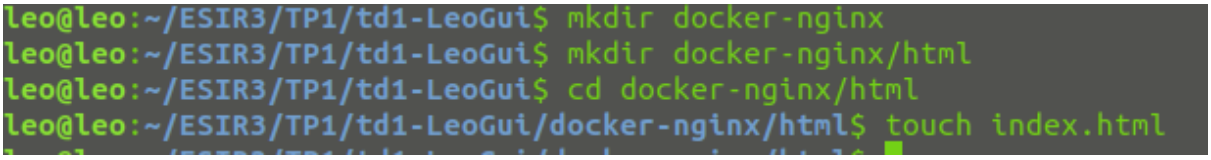


Figure 13 : Création des dossiers

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

Question 04

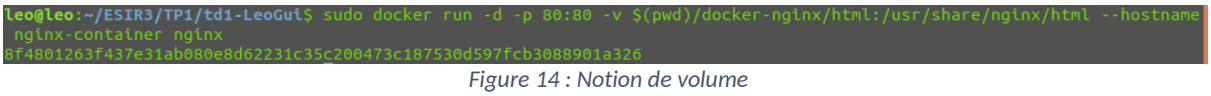


Figure 14 : Notion de volume

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

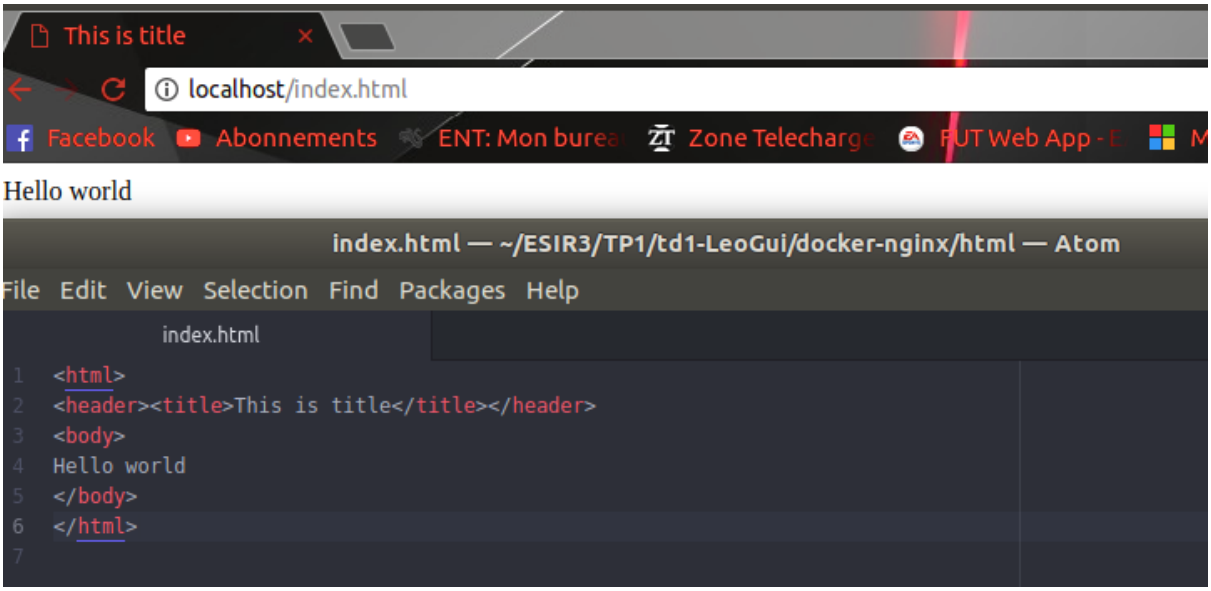
Question 05

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
      NAMES
8f4801263f43        nginx               "nginx -g 'daemon of…"   About a minute ago   Up About a minute   0.0.0.0:80->80
/tcp   eager_davinci
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker logs eager_davinci
172.17.0.1 - - [18/Oct/2018:16:57:28 +0000] "GET /index.html HTTP/1.1" 200 106 "-" "Mozilla/5.0 (X11; Linux x86_64) Appl
eWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36" "-"
```

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

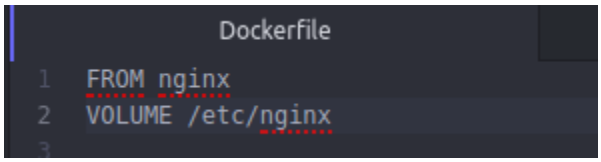


Figure 16 : Dockerfile

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

Question 10

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker build --tag mynginximage .
Sending build context to Docker daemon  1.049MB
Step 1/2 : FROM nginx
 ---> dbfc48660aeb
Step 2/2 : VOLUME /etc/nginx
 ---> Running in 8498e9b9ea6e
Removing intermediate container 8498e9b9ea6e
 ---> fd52a301c593
Successfully built fd52a301c593
Successfully tagged mynginximage:latest
```

Figure 17 : docker build

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mynginximage        latest              fd52a301c593        2 minutes ago       109MB
node                latest              a2b9536415c2        2 days ago          674MB
nginx               latest              dbfc48660aeb        2 days ago          109MB
```

Figure 18 : image

L'image a bien été créée.

Question 11

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ sudo docker run -d -p 80:80 --name nginx2 mynginximage
6849745c71dc4d20fda7f7a7cb86a59e0e5600aca5b29c088e7a711da006e85d
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker run -i -t --volumes-from nginx2 --name nginxfiles debian /bin/bash
root@5e0a254f5a0e:/#
```

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

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker run -i -t --volumes-from nginx2 --name nginxfiles debian /bin/bash
root@61a8ba93ef17:/# ls
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
root@61a8ba93ef17:/# cd etc/nginx/conf.d/
root@61a8ba93ef17:/etc/nginx/conf.d# ls
default.conf
root@61a8ba93ef17:/etc/nginx/conf.d# cat default.conf
```

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 ».

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;
}

```

Figure 22 : nginx.conf

Question 15

default.conf

```
server {
    listen 8080;
    root /docker-nginx/html;
    server_name  localhost;

    #charset koi8-r;
    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }


    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    #location ~ \.php$ {
    #    root           html;
    #    fastcgi_pass   127.0.0.1:9000;
    #    fastcgi_index  index.php;
    #    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
    #    include        fastcgi_params;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny  all;
    #}
}
```

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 :

default.conf

```
server {
    listen 80;
    server_name  localhost;

    location / {
            proxy_pass http://localhost:8080;
    }

    location ~ \.(gif|jpg|png)$  {
            root /usr/share/nginx/image;
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
}


server {
    listen 8080;
    server_name  localhost;

    location / {
          root /usr/share/nginx/html;
    }

}
```

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.

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ sudo docker run -d -p 80:80 --name nginx2 -v $(pwd)/td1/html:/usr/share/nginx/html -v $(p
wd)/docker-nginx/conf:/etc/nginx/conf.d -v $(pwd)/td1/images:/usr/share/nginx/image mynginximage
[sudo] Mot de passe de leo :
05c3df7993522e7fa70e0a2c2a4cc16486dfd70755af2707e05f805b4650a332
```

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

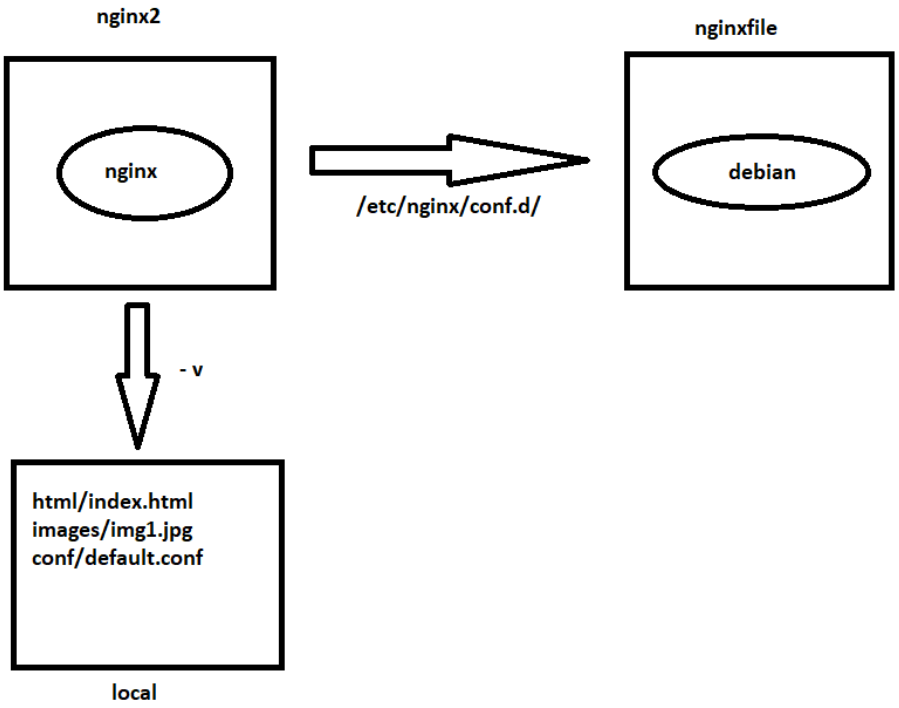


Figure 25 : Explication du montage

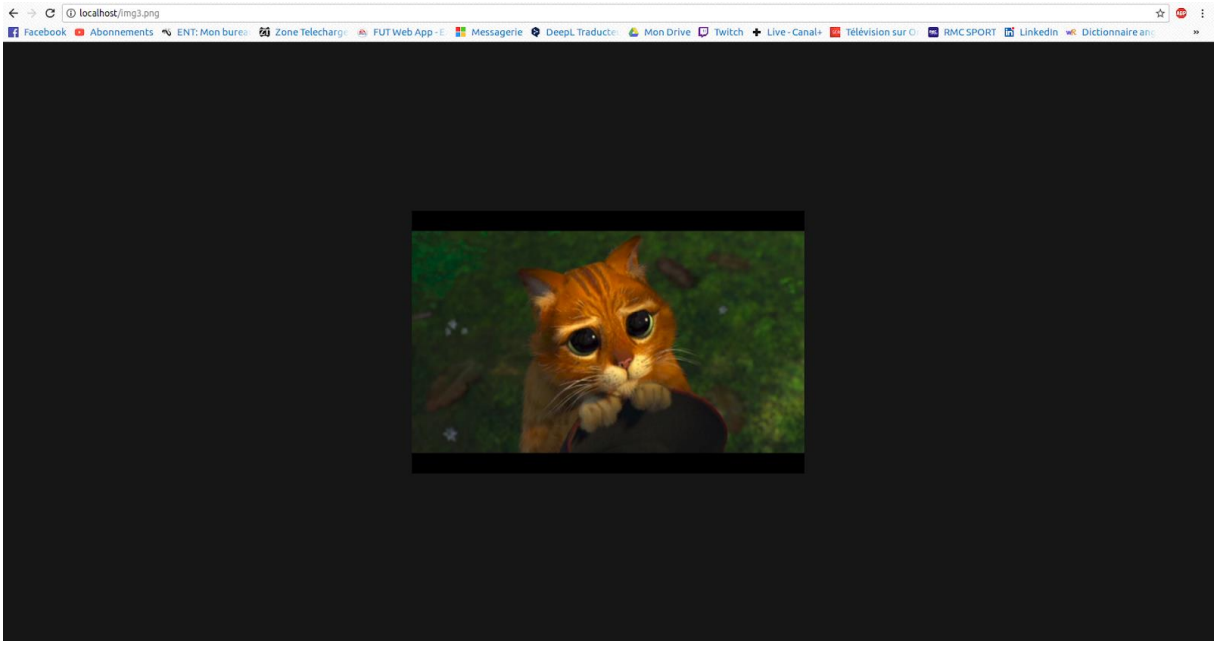


Figure 26 : img1.jpg

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

Dockerfile

```
FROM nginx
COPY ./docker-nginx/conf /etc/nginx/conf.d
COPY /td1/images /usr/share/nginx/image
COPY /td1/html /usr/share/nginx/html
VOLUME /etc/nginx
```

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.

```
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker build -t nginxcustom .
Sending build context to Docker daemon   3.04MB
Step 1/5 : FROM nginx
 ---> dbfc48660aeb
Step 2/5 : COPY ./docker-nginx/conf /etc/nginx/conf.d
 ---> 1606c269417c
Step 3/5 : COPY /td1/images /usr/share/nginx/image
 ---> 327ac143c006
Step 4/5 : COPY /td1/html /usr/share/nginx/html
 ---> 2d2aa16d2775
Step 5/5 : VOLUME /etc/nginx
 ---> Running in 274f56db0b4e
Removing intermediate container 274f56db0b4e
 ---> 7c1892b45d21
Successfully built 7c1892b45d21
Successfully tagged nginxcustom:latest
leo@leo:~/ESIR3/TP1/td1-LeoGui$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginxcustom         latest              7c1892b45d21        About a minute ago   109MB
mynginximage        latest              fd52a301c593        4 days ago          109MB
node                latest              a2b9536415c2        7 days ago          674MB
nginx               latest              dbfc48660aeb        7 days ago          109MB
hello-world         latest              4ab4c602aa5e        6 weeks ago         1.84kB
<none>              <none>              248792239874        6 months ago        601MB
tradfri-dev         latest              c3cb7a1f49fe        6 months ago        601MB
debian              latest              2b98c9851a37        7 months ago        100MB
```

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.