

26/10/2018

Léo Guilpain – ESIR 3 - IoT

Docker – TD2

*« 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](#_Toc528232867)

[Partie 1 : Interconnexion de conteneurs 2](#_Toc528232868)

[Question 01 2](#_Toc528232869)

[Question 02 2](#_Toc528232870)

[Question 03 3](#_Toc528232871)

[Question 04 4](#_Toc528232872)

[Question 05 4](#_Toc528232873)

[Question 06 5](#_Toc528232874)

[Docker Link : 5](#_Toc528232875)

[Question 01 5](#_Toc528232876)

[Question 02 6](#_Toc528232877)

[Question 03 7](#_Toc528232878)

[Question 4 7](#_Toc528232879)

[Partie II. Docker-compose 8](#_Toc528232880)

[Question 01 8](#_Toc528232881)

[Question 02 8](#_Toc528232882)

[Question 03 9](#_Toc528232883)

[Question 04 9](#_Toc528232884)

[Conclusion 10](#_Toc528232885)

Introduction

Le but de ce TP est de d’héberger chaque site dans son propre conteneur docker.

Partie 1 : Interconnexion de conteneurs

Question 01

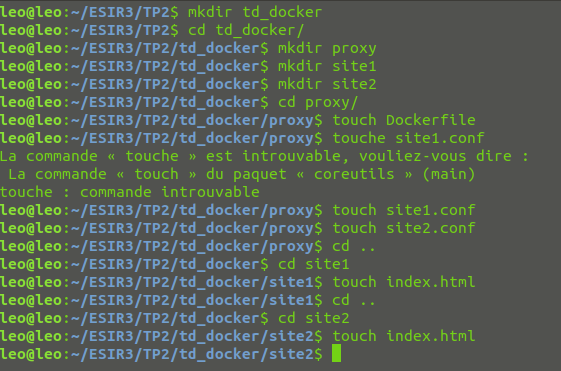


Figure 1 : Création des dossiers et fichiers

Question 02

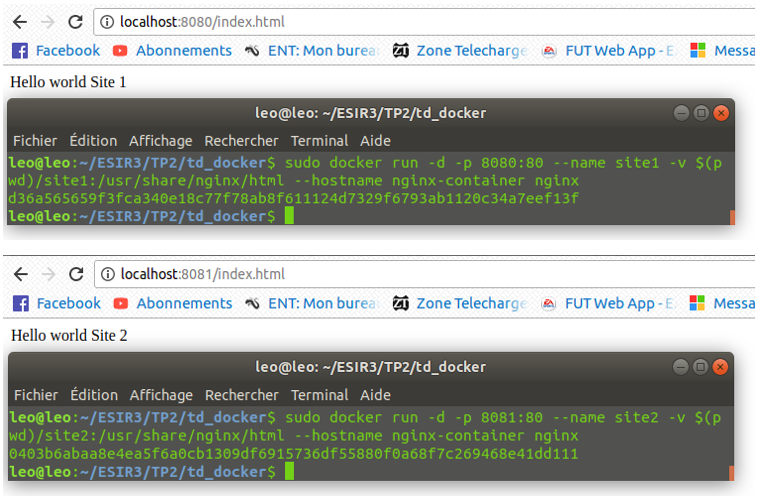


Figure 2 : Visualisation des sites

Question 03

Dans un premier temps on récupère les adresses IP à l’aide de la commande ci-dessous. On obtient pour le site 1 : **172.17.0.2** et pour le site 2 : **172.17.0.3**

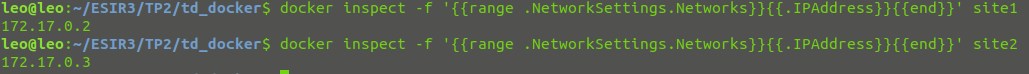


Figure 3 : Récupération adresses IP

On configure le fichier « site1.conf » comme ceci :

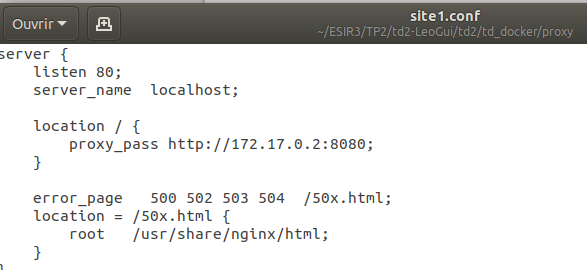


Figure 4 : Site1.conf

À noter que le fichier « site2.conf » est configuré de façon similaire en modifiant l’adresse IP et le numéro de port.

Ensuite on modifie le fichier Dockerfile de la même façon que dans le TP précédent.

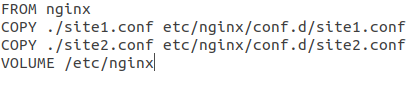


Figure 5 : Dockerfle

Question 04

Une fois le fichier Dockerfile configuré, il suffit de lancer la commande « docker build -t myproxy . » :

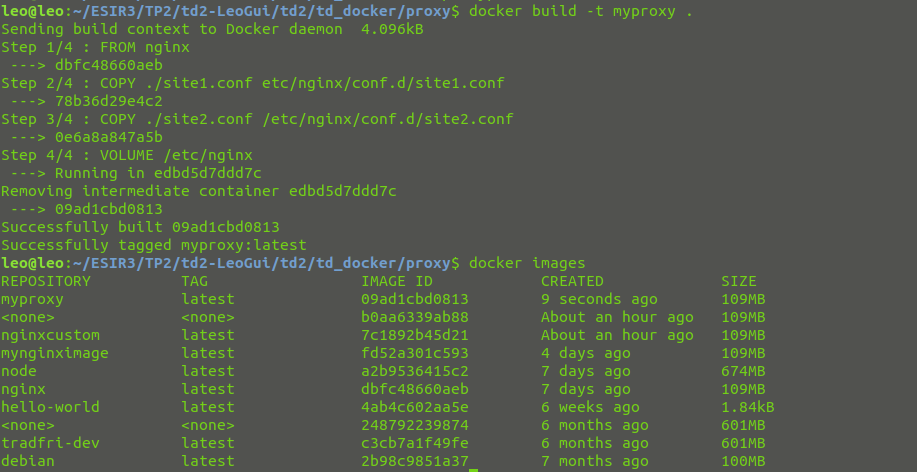


Figure 6 : Build de l'image

On voit bien grâce à “docker images” que l’image a été correctement créée.

Pour créer l’image et lancer le conteneur, il suffit de faire comme ceci :

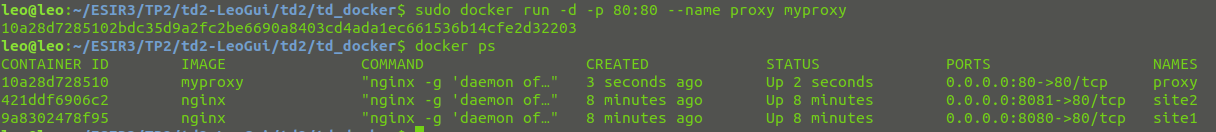


Figure 7 : Lancement du conteneur

On a lancé le conteneur « proxy » à partir de l’image « myproxy »

Question 05

On modifie le fichier hosts qui se trouve sur ordinateur à l’aide de la commande « vim »

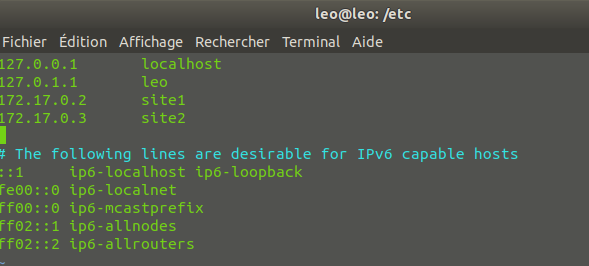


Figure 8 : etc/host

Ensuite lorsque l’on fait la commande “curl <http://site1>” et “curl <http://site2>” , on obtient ceci :



Figure 9 : curl site1

Les fichiers « index.html » des deux sites ont bien été liés correctement car on peut maintenant y avoir accès.

Question 06

Le problème de cette solution c’est qu’il faut connaître l’adresse IP des deux sites. Il faut également configurer sur l’ordinateur les adresses IP. Donc si ces dernières changent, la connexion ne se fait plus. De plus, les adresses IP et les numéros de ports sont stockés dans les fichiers de configuration, cela peut donc engendrer des problèmes de confidentialité mais surtout des problèmes de sécurisation.

Docker Link :

Question 01

On effectue la commande ci-dessous pour lier les conteneurs :

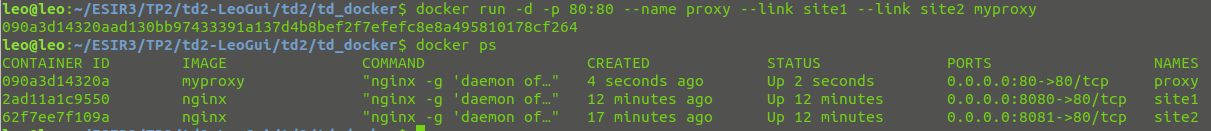


Figure 10 : lien entre les conteneurs

Ensuite on regarde les variables d’environnement :

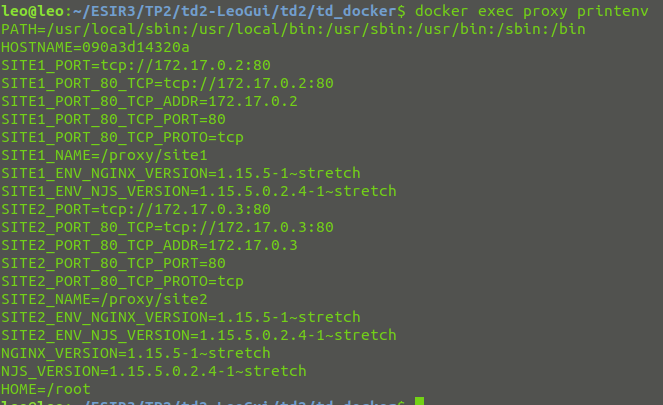


Figure 11 : Variables d'environnement

Comme prévu, nous avons des liens entre les différents conteneurs. Par exemple, nous avons accès aux différentes adresses IP. Nous allons donc pouvoir faire des références à ces variables.

Question 02

Pour avoir accès au shell du conteneur “proxy”, on effectue la commande suivante :

“sudo docker exec -it proxy bash”.

Ensuite après avoir mis à jour, on télécharge et installe “vim” pour pouvoir modifier le fichier en cas de besoin.

En naviguant dans les répertoires du conteneur « proxy », et surtout dans le répertoire « etc/host », on peut voir que les adresses IP ont bien été récupérées correctement.

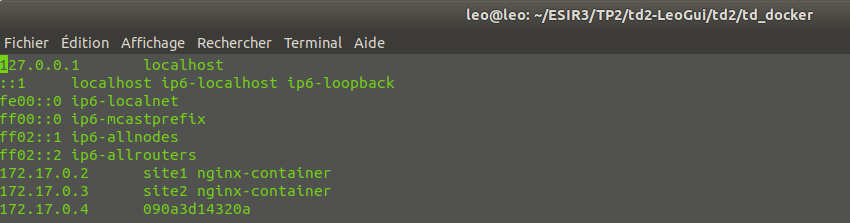


Figure 12 : etc/host

Question 03

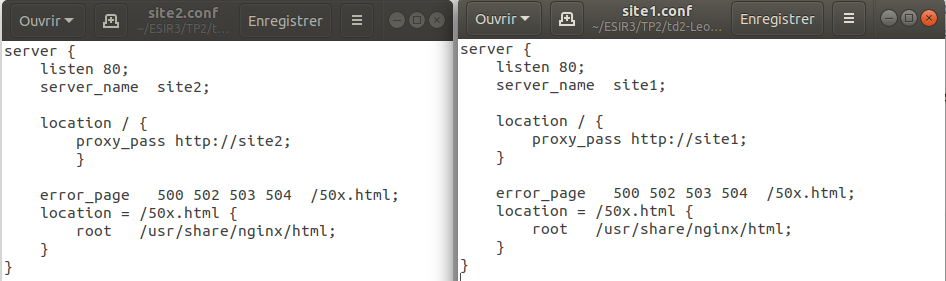


Figure 13 : Fichiers de configuration

On a ici, dans les deux fichiers de configuration, remplacé l’adresse IP ainsi que le numéro de port par le nom des conteneurs. Comme le lien a été fait auparavant, le conteneur « proxy » a accès à ces variables. Cela permet également une meilleure sécurisation des données.

Question 4

Il faut donc refaire toutes les manipulations. Après ceci, on se rend compte du bon fonctionnement en réalisant la commande “curl <http://site1>”. Cette dernière renvoie la même chose que précédemment alors que l’on a modifié les fichiers de configuration. Cela signifie donc que le lien a bien été fait correctement. Désormais le proxy connait les noms de domaines “site1” et “site2”

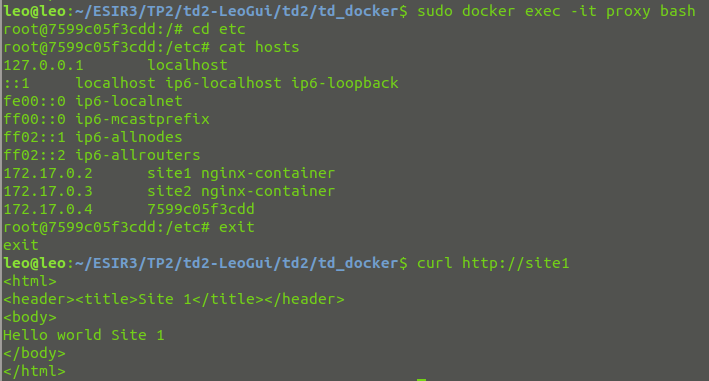


Figure 14 : curl site1

Partie II. Docker-compose

Question 01

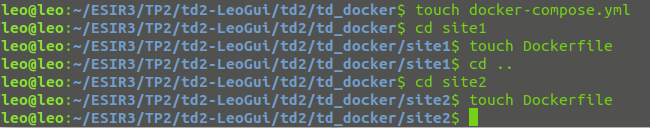


Figure 15 : Création des fichiers

Question 02

On modifie le fichier “docker-compose.yml” comme ceci :



Figure : docker-compose.yml

Le premier bloc correspond au conteneur “proxy”. On le build à partir du fichier “Dockerfile” se trouvant dans le dossier proxy. Ce conteneur sera lié au conteneur “site1” et “site2” et sera lancé sur le port “80”.

Le conteneur “site1” sera lancé à partir de l’image nginx et sera monté à partir du dossier “site1”. Il sera lancé sur le port 8080.

Le conteneur “site2” sera lancé à partir de l’image nginx et sera monté à partir du dossier “site2”. Il sera lancé sur le port 8081.

Question 03



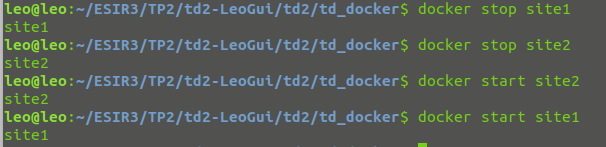
Figure : docker -compose up

Lorsqu’on lance la commande “docker-compose up --build”, le build se fait correctement.

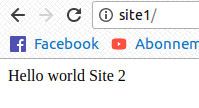
Ensuite on a la ligne “attaching to site2,site1,proxy”. Il suffit ensuite sur le navigateur de taper “<http://site1>” ou “<http://site2>” et nous avons bien accès à l’index.html du site. Comme on peut le voir dans l’invite de commande, les opérations se font correctement.

Question 04

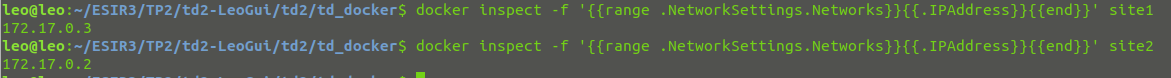
On effectue les commandes suivantes :



Ensuite lorsque l’on tape sur internet “<http://site1>”, on obtient l’index.html du site 2



Pour comprendre pourquoi ce fonctionnement, j’ai regardé les deux adresses IP des conteneurs :



Comme on peut le voir, elles ont été inversées. C’est donc pour cela que les index.html sont inversés.

Conclusion

Dans ce TP sur Docker, nous avons donc réussi à héberger chaque site dans un conteneur docker différents.