**Rapport de laboratoire**

WEB Infrastructure Lab

Amine Tayaa – Valentin Schaad – Sébastien Henneberger – (Benoît Zuckschwerdt)

1er juin 2015

**Table des matières :**

1 Introduction 3

2 Mise en place du front-end 4

3 Mise en place du back-end 7

4 Mise en place du serveur (reverse proxy + load-balancer) 9

5 Tests 11

6 Conclusion 12

# Introduction

Le but de ce laboratoire est de créer une infrastructure web et d’implémenter notamment un reverse proxy et un répartiteur de charge (load-balancer).

En tant qu’utilisateur, on devra pouvoir ouvrir une page web dont le contenu statique est délivré par le container implémentant le front-end. Un script s’exécutera dans la page web affichée par le client qui effectuera des requêtes AJAX pour récupérer un contenu dynamique délivré par le back-end.

Le front-end doit implémenter les sticky-sessions, c’est-à-dire qu’une fois qu’un client s’est connecté à un front-end, il doit se souvenir de lui et garder la même session pour leur communication. Autrement-dit, le container utilisé sera le même pour toute la session.

A l’inverse, le back-end coupera la connexion après toute communication. C’est-à-dire, que ce ne sera pas forcément le même container implémentant un back-end qui sera attribué pour le même utilisateur au cours du temps.

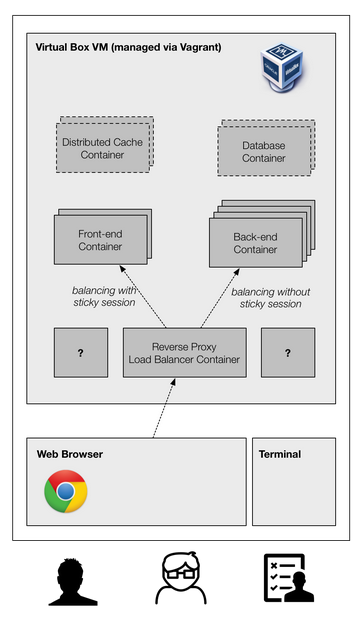


Schéma 1 Infrastructure Web

# Mise en place du front-end

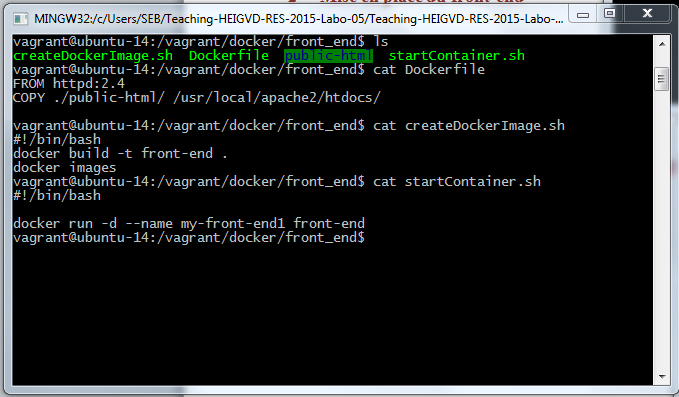
Nous avons utilisé l’image httpd disponible dont la documentation est disponible sous :

<https://registry.hub.docker.com/_/httpd/>

Il s’agit d’une image officielle d’un serveur Apache http.

Du coup, on se créé un fichier nommé DockerFile qui sera utilisé pour la création de l’image, voir la capture ci-dessous.

Pour plus de simplicité, on s’est créé un script permettant de créer l’image docker et de démarrer un container avec cette image, voir la capture ci-dessous :



Ensuite, il faut faire la page web avec un fichier index.html.

Il s’agit d’une simple page web, affichant un titre. Mais, elle possède un script javascript qui toutes les secondes interroge le back-end pour obtenir un contenu dynamique (représentation JSON de /api/message) :

<script language=**"JavaScript"**>

$**(**document**).**ready**(*function*** **()** **{**

refreshNodes**();**

**});**

***function*** refreshNodes**()** **{**

$**.**getJSON**(**'/api/message'**,**

***function*(**data**)** **{**

$**(**'#monitor'**).**html**(** data.author**);**

**});**

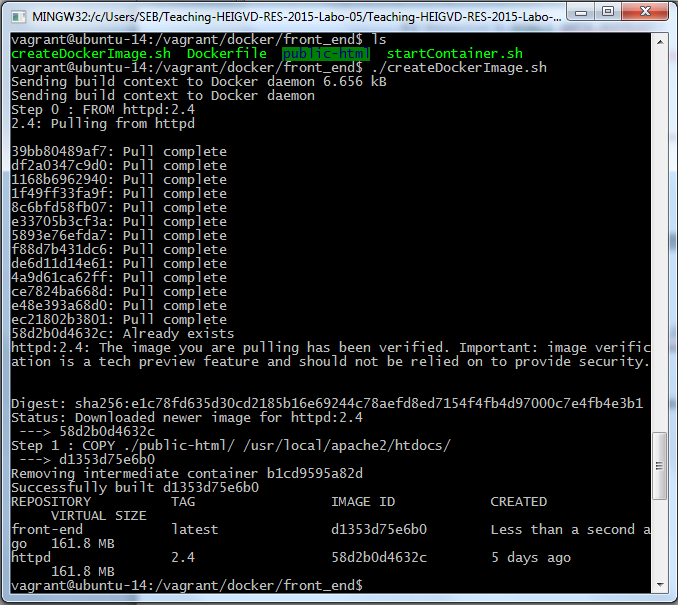
***var*** t**=**setTimeout**(**"refreshNodes()"**,** 1000**);**

**}**

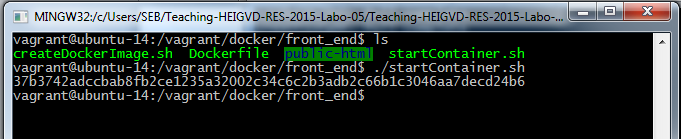
</script>

On peut ensuite démarrer un container grâce au deux scripts créés précédemment :

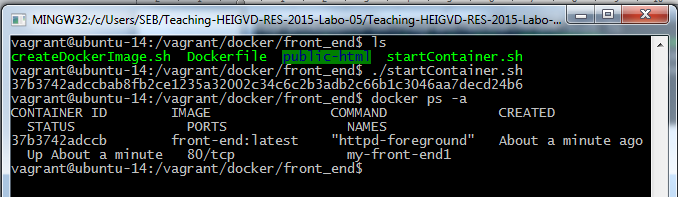
La création de l’image prend quelques minutes.



Puis, on démarre un container avec cette image créée :

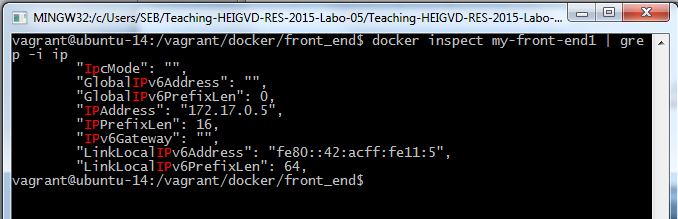


On vérifie que tout s’est bien passé :



Tout semble ok, le container existe.

On va finalement récupérer l’adresse ip du container pour pouvoir paramétrer le serveur à la fin :

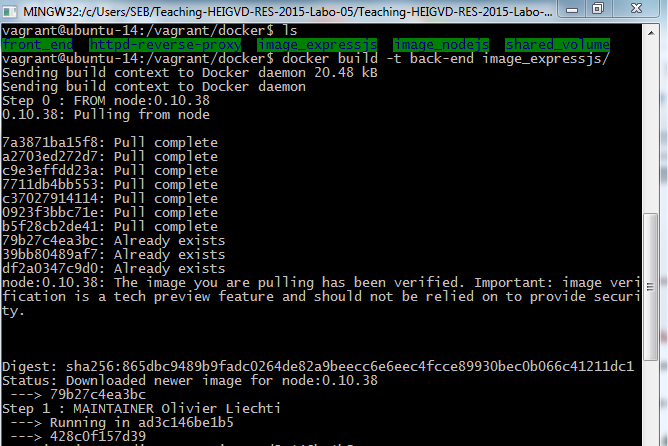


Le container implémentant le front-end possède l’adresse ip 172.17.0.5.

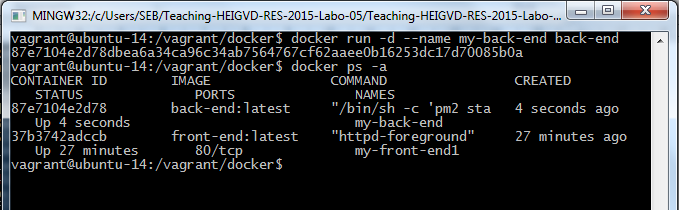
# Mise en place du back-end

Nous avons utilisé l’image nommée « image\_express.js » du labo3.

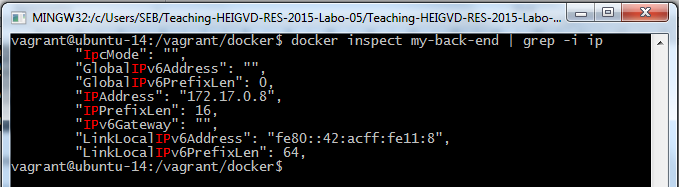
Tout d’abord, on crée l’image (capture coupée) :



Ensuite, on lance un container avec cette image, et on vérifie que tout s’est déroulé correctement :



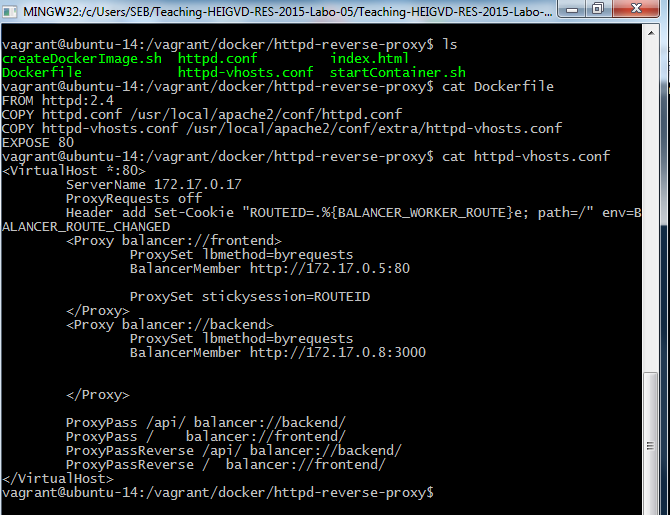
On récupère l’adresse IP :



L’adresse ip du container implémentant le back-end est 172.17.0.8

# Mise en place du serveur (reverse proxy + load-balancer)

On utilise la même image httpd que précédemment pour le front-end :

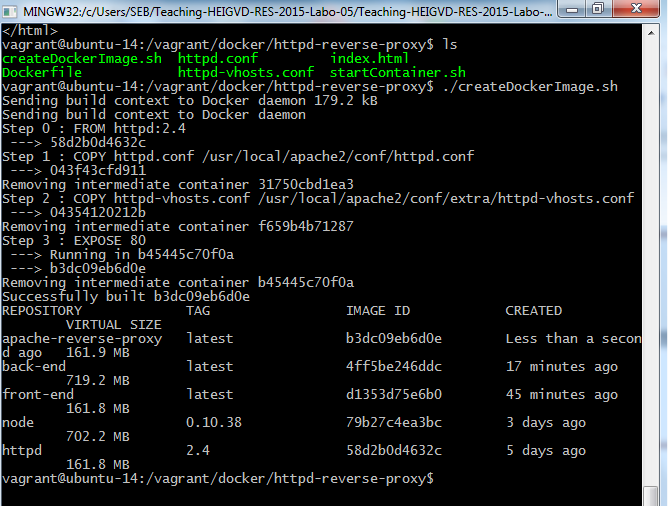


Grâce aux adresses ip récupérée, on peut configurer correctement le load-balancing.

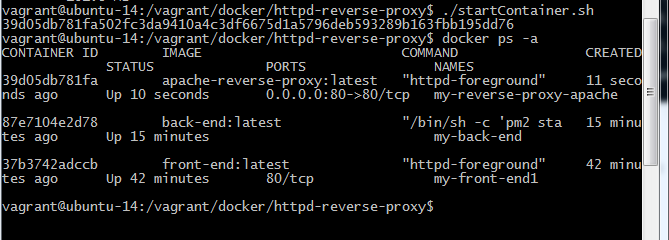
Quand le serveur verra passer une requête pour /api, il saura qu’il doit la transmettre au back-end, et si c’est plutôt / alors il sait que cela concerne le front-end.

Et le front-end utilise les sticky-sessions.

On peut donc créer l’image :



On crée le container associé et on vérifie que tout s’est bien passé :



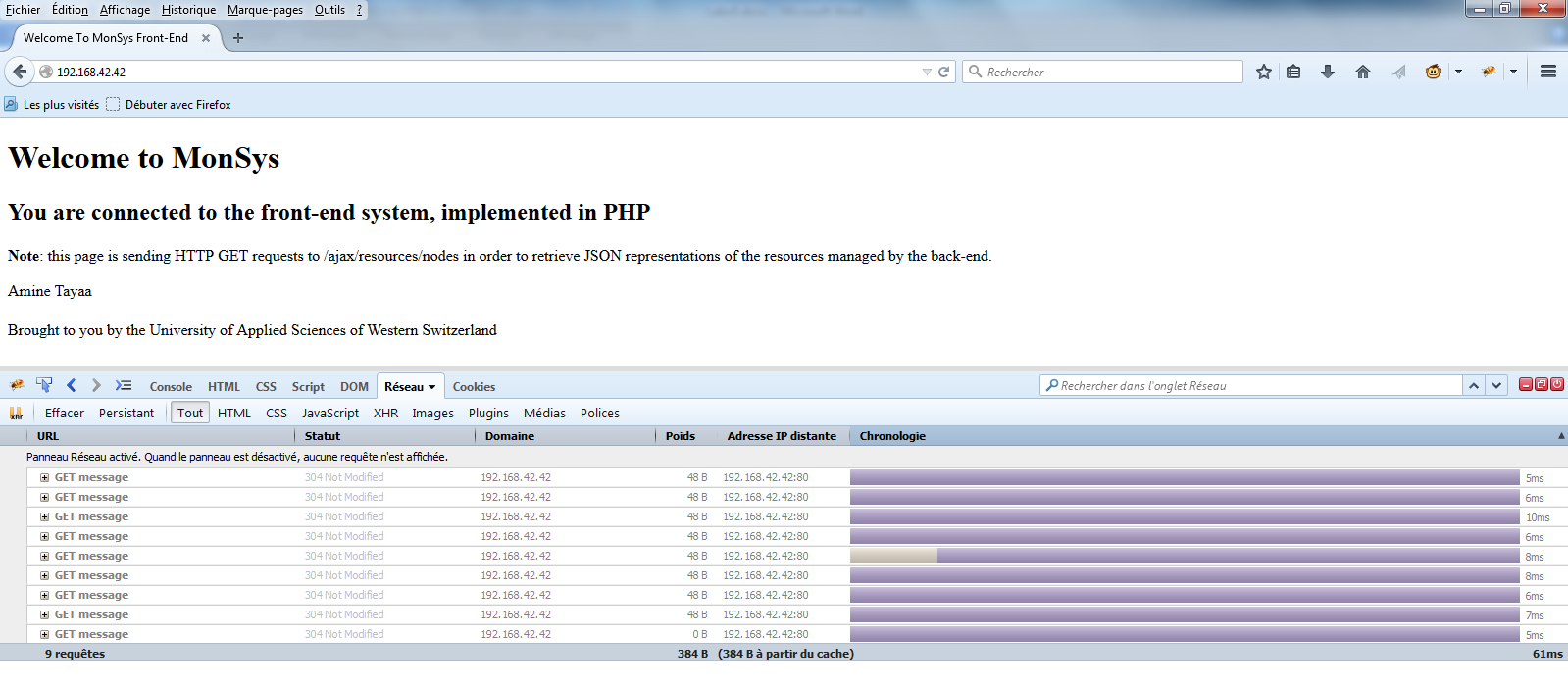
Donc on a bien nos trois containers : le front-end, le back-end et le serveur.

# Tests

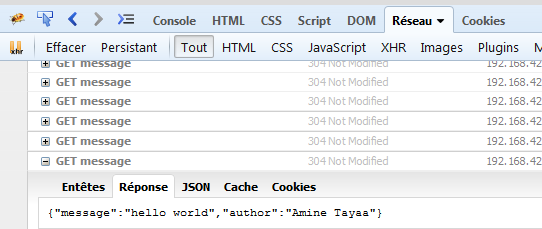
Testons déjà l’ouverture de la page web à l’adresse 192.168.42.42 (adresse configurée statiquement dans le fichier Vagranfile ) :



Et si on regarde avec fireBug, on devrait voir les requêtes GET toutes les secondes :



On voit que la réponse concerne bien la représentation JSON du message et de l’auteur :



Le contenu délivré par le back-end n’est pas dynamique pour le moment.

Pour ce faire, l’idée est d’afficher sur la page web, les adresses ip des containers répondant. Par-conséquent, on verra si les sticky-sessions sont fonctionnels pour le front-end et pas pour le back-end.

# Conclusion

Le dialogue s’effectue correctement entre le serveur, le front-end et le back-end, mais lorsqu’il existe seulement un exemplaire de chaque image.

Nous n’avons pas réussi à implémenter les sticky-sessions (en ajoutant route=1, route=2).

Nous nous sommes pris beaucoup trop tard pour ce laboratoire et nous nous somme laissé submergé par la charge de travail.

Nous avons mis la priorité sur un autre cours.

Yverdons-les-Bains, le 1 juin 2015

Amine Tayaa – Valentin Schaad

Sébastien Henneberger – (Benoît Zuckschwerdt)