A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

11/06/2017

Laboratoire RES

HTTP INFRASTRUCTURE

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right.

Zacharie Nguetack & Silver Kameni

Table des matières

INTRODUCTION	2
Step 1: Static HTTP server with apache httpd	2
Step 2: Dynamic HTTP server with express.js	8
Step 3: Reverse proxy with apache (static configuration)	13
Step 4: AJAX requests with JQuery	17
Step 5: Dynamic reverse proxy configuration.....	21
CONCLUSION.....	25

INTRODUCTION

Dans ce laboratoire, nous allons implémenter une application client-serveur en utilisant les images docker qui encapsule un serveur apache qui nous livre les contenus statique ou dynamique selon la configuration. Ce laboratoire a trois principaux objectifs.

Le premier objectif de ce laboratoire est de se familiariser avec les outils logiciels qui nous permettront de créer une infrastructure Web complète. Par cela, nous entendons que nous allons construire un environnement qui nous permettra de servir des contenus statiques et dynamiques aux navigateurs Web.

Le deuxième objectif est de mettre en œuvre une application web dynamique. Nous créerons des ressources HTML, CSS et JavaScript qui seront diffusées aux navigateurs et présentées aux utilisateurs. Le code JavaScript exécuté dans le navigateur émettra des requêtes HTTP asynchrones à notre infrastructure Web et récupérera le contenu généré dynamiquement.

Le troisième objectif est de pratiquer notre utilisation de Docker. Tous les composants de l'infrastructure Web seront encapsulés dans des images Docker personnalisées.

Step 1: Static HTTP server with apache httpd

A. Critères d'acceptation

❖ **Vous avez un compte GitHub avec tout le nécessaire pour construire l'image Docker.**

Pour réunir tous les résultats de nos manipulations et selon la spécification du labo, nous avons créé un compte github dont le nom est le suivant : « **Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions** »

The screenshot shows the GitHub interface for the repository 'Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions' owned by 'silverkameni'. The repository description is 'the repo for the HTTP infra lab solution'. It shows 1 commit, 6 branches, 0 releases, and 1 contributor. The 'Code' tab is selected, displaying the repository's file structure with 'LICENSE' and 'README.md' files, both marked as 'Initial commit' from 'a day ago'. A 'Clone or download' button is visible. An 'Active Windows' notification is present in the bottom right corner.

❖ Vous faites une démo, où vous construisez l'image, exécutez un conteneur et accédez au contenu d'un navigateur.

Préparation :

Nous allons commencer au préalable par cloner le repo github du labo dans un répertoire spécifique en local via la commande:

Git clone [git@github.com:SoftEng-HEIGVD/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions.git](https://github.com/SoftEng-HEIGVD/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions.git)

Puis dans le dossier du repo cloné, nous devons créer l'arborescence suivante afin de mieux structurer notre travail:

C:\Users\SILVERCORP\Teaching-HEIGVD-RES-2016-LabBox\RES\Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions\docker-images\apache-php-image, et dans ce dernier répertoire, nous allons créer le fichier « **Dockerfile** » qui nous servira plus tard à build notre image docker.

❖ You have used a nice looking web template, different from the one shown in the webcast.

Afin de choisir une image contenant déjà les bons paquetages pour le fonctionnement du serveur httpd, nous nous rendons sur le site : <https://hub.docker.com/> et recherchons « httpd », nous allons dans notre cas choisir **l'image officielle de httpd**.

En sélectionnant donc l'image httpd officiel, on constate que nous avons une page qui nous donne toutes les directives d'utilisation et toutes les informations sur l'image. Ainsi en lisant sous l'onglet, '**how to use this image**' sous le lien '**php image**', on constate que nous avons la possibilité d'utiliser une image officielle de apache qui est déjà configurée pour servir des pages php ; ce qui sera important pour nous dans la suite de ce laboratoire.

Nous choisirons donc pour la suite de notre travail, la version de l'image officielle PHP:

- 7.0.20-apache, 7.0-apache ([7.0/apache/Dockerfile](#))

❖ You are able to explain what you do in the Dockerfile.

Nous avons toutes les spécifications et informations devant servir à build notre image. Donc nous allons dès lors remplir notre Dockerfile avec les informations suivantes :

```
FROM php:7.0-apache
```

```
COPY src/ /var/www/html/
```

Where `src/` is the directory containing all your PHP code. Then, run the commands to build and run the Docker image:

La première ligne permet de dire que l'image sera build en se basant sur la version « php :7.0-apache » et que à chaque fois qu'on démarrera un container, alors tous les fichiers se trouvant dans le dossier **src/** sera copier dans le répertoire **/var/www/html/** de notre container.

❖ You do a demo, where you build the image, run a container and access content from a browser.

Exploration et exécution de l'image :

Afin de voir ce qu'il y'a à l'intérieur de notre apache dont nous venons de créer le Dockerfile, nous ferions les manipulations suivantes afin de créer, d'inspecter des containers et aussi tester le bon fonctionnement de notre serveur.

Nous commençons d'abord par faire un « **docker ps** » afin de visualiser tous les containers en exécution et on constate bien évidemment qu'aucun container n'a été lancé.

Puis nous exécuterons donc la commande : **docker run -d -p 9090 :80 php:7.0-apache** qui aura donc pour rôle de lancer notre image (**run**) en arrière-plan (**-d**) auquel nous associerons un port mapping (**-p 9090 :80**) ce qui signifiera que si le serveur se met à l'écoute sur le port 80 alors nous allons nous mapper sur le port 9090, puis nous indiquerons également le nom du serveur apache (**php:7.0-apache**). A ce niveau, notre container doit s'être

normalement lancé et nous vérifierons cela en faisant à nouveau un « **docker ps** » et cette fois ci on constate bien que le container est bel et bien en cours d'exécution.

En faisant donc un « **docker logs nom_du_container** », nous devons également avoir la preuve que notre container s'est parfaitement lancé.

```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS
642a038ca22b       php:7.0-apache     "docker-php-entryp..." 14 seconds ago     Up 13 seconds
0.0.0.0->9090->80/tcp    quirky_bardeen

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ docker logs quirky_bardeen
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Sat May 27 10:20:58.121294 2017] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.10 (Debian) PHP/7.0.19 configured -- resuming normal operations
[Sat May 27 10:20:58.131876 2017] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
```

On peut d'ores et déjà faire un **test** afin de s'assurer du bon fonctionnement de notre serveur ; pour ce faire, on saisira la commande : **telnet 192.168.99.100 9090**, où **192.168.99.100** est l'adresse de notre docker et **9090** est le port sur lequel on s'est mappé lors du lancement du container.

Puis nous pourrions envoyer une commande à notre serveur (**GET / HTTP/1.0**) et observerons la réponse du serveur.

```
SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ telnet 192.168.99.100 9090
Trying 192.168.99.100...
Connected to 192.168.99.100.
Escape character is '^]'.
GET / HTTP/1.0
HTTP/1.1 400 Bad Request
Date: Sat, 27 May 2017 10:25:53 GMT
Server: Apache/2.4.10 (Debian)
Content-Length: 302
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.10 (Debian) Server at 172.17.0.2 Port 80</address>
</body></html>
Connection closed by foreign host.
```

Nous pourrions aussi grâce à la commande « **docker inspect nom_du_container** » pour visualiser toutes les informations (adresse IP...etc.) à propos du container que nous avons lancé.

```
SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
bde9bda39155      php:7.0-apache     "docker-php-entryp..." 5 seconds ago       Up 2 seconds       0.0.0.0:9090->80/tcp
eloquent_cori

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ docker inspect eloquent_cori
[
  {
    "Id": "bde9bda39155876880544879b73d8f2107ba00279a52d91a87ad1fb4188fb02f",
    "Created": "2017-05-28T14:41:11.668088117Z",
    "Path": "docker-php-entrypoint",
    "Args": [
      "apache2-foreground"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 1990,
      "ExitCode": 0,
      "Error": ""
    },
    "Image": "php:7.0-apache",
    "NetworkSettings": {
      "Bridge": "br-172.17.0.1",
      "Gateway": "172.17.0.1",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "IPAddress": "172.17.0.2",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "MacAddress": "02:42:ac:11:00:02",
      "Networks": {
        "bridge": {
          "IPAMConfig": null,
          "Links": null,
          "Aliases": null,
          "NetworkID": "2909c3c8f185fddcb9fc6faf28e881e1e73c1187057ef3b96daf3924109d8632",
          "EndpointID": "a448683046ddf590a1c802fafcd55292919e7a12ec3c15c68a8bf154518f012b",
          "Gateway": "172.17.0.1",
          "IPAddress": "172.17.0.2",
          "IPPrefixLen": 16,
          "IPv6Gateway": "",
          "GlobalIPv6Address": "",
          "GlobalIPv6PrefixLen": 0,
          "MacAddress": "02:42:ac:11:00:02"
        }
      }
    }
  }
]
```

```

    "NetworkSettings": {
      "Bridge": "br-172.17.0.1",
      "Gateway": "172.17.0.1",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "IPAddress": "172.17.0.2",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "MacAddress": "02:42:ac:11:00:02",
      "Networks": {
        "bridge": {
          "IPAMConfig": null,
          "Links": null,
          "Aliases": null,
          "NetworkID": "2909c3c8f185fddcb9fc6faf28e881e1e73c1187057ef3b96daf3924109d8632",
          "EndpointID": "a448683046ddf590a1c802fafcd55292919e7a12ec3c15c68a8bf154518f012b",
          "Gateway": "172.17.0.1",
          "IPAddress": "172.17.0.2",
          "IPPrefixLen": 16,
          "IPv6Gateway": "",
          "GlobalIPv6Address": "",
          "GlobalIPv6PrefixLen": 0,
          "MacAddress": "02:42:ac:11:00:02"
        }
      }
    }
  }
]
```

❖ You are able to show where the apache config files are located (in a running container).

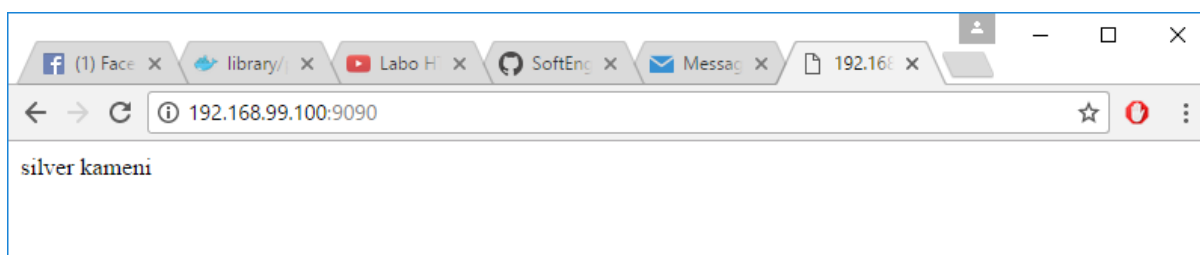
Explorer une image :

Pour explorer une image, on peut se connecter sur un container en exécution, pour ce faire, nous utiliserons la commande suivante : « **docker exec -it nom_du_conteneur commande_a_executer** »

Dans notre cas, la commande à exécuter sera : **/bin/bash** qui aura pour rôle de nous ouvrir un shell sur le container spécifié.

```
MINGW64:/c/Users/SILVERCORP/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInf

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-
images/apache-php-image (master)
$ docker exec -it eloquent_cor1 /bin/bash
root@bde9bda39155:/var/www/html# ls
root@bde9bda39155:/var/www/html# echo "silver kameni" > index.html
root@bde9bda39155:/var/www/html# echo "silver kameni" > index.html
root@bde9bda39155:/var/www/html# ls
index.html index.html
root@bde9bda39155:/var/www/html# rm index.html
root@bde9bda39155:/var/www/html# ls
index.html
root@bde9bda39155:/var/www/html#
```



A cette étape, nous avons déjà testé le fonctionnement de chacune des fonctionnalités et mis en place les éléments nécessaires à la construction de notre image et au démarrage d'un container.

Builder notre image httpd :

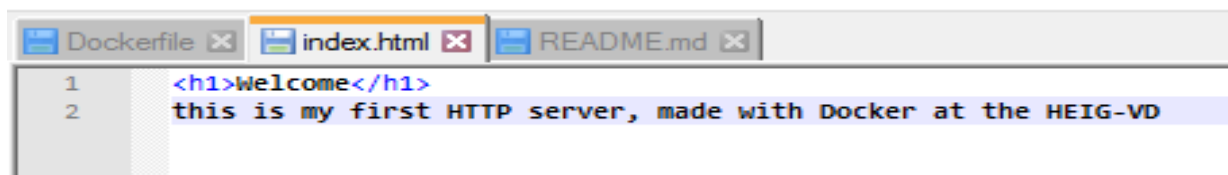
Tout d'abord, nous allons créer un nouveau dossier « content » au lieu de 'src' comme spécifié dans le Dockerfile, et dans ce dossier, nous allons créer un fichier **index.html** (page qui sera affiché dans le navigateur).

Nous devons donc modifier le contenu de notre Dockerfile comme suit :

```
Dockerfile  index.html  README.md
1 FROM php:7.0-apache
2 COPY content/ /var/www/html/
```

Afin de dire que les pages html contenu dans le dossier « content » seront copiée dans le dossier « /var/www/html » lors du lancement d'un container et c'est grâce à ce mécanisme, que nous pourrions afficher les pages html sur le navigateur.

Dans notre fichier **index.html**, nous allons y mettre du code html qui sera affiché dans le navigateur.



```
Dockerfile x index.html x README.md x
1 <h1>Welcome</h1>
2 this is my first HTTP server, made with Docker at the HEIG-VD
```

nous allons saisir la commande : **Docker build -t res/apache_php .**

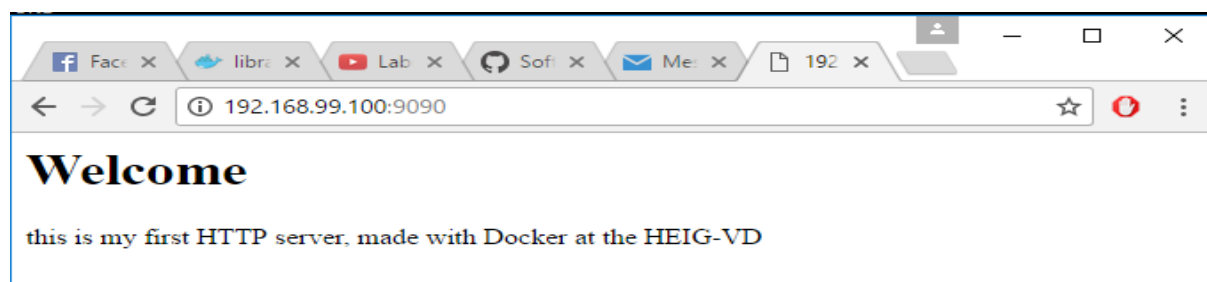
Ce qui signifie que le nom de notre image sera désormais « **res/apache_php** », et « . » signifie que le build doit être lancé en utilisant les éléments du répertoire courant.

```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ ls
content/ Dockerfile

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ docker build -t res/apache_php .
Sending build context to Docker daemon 3.584kB
Step 1/2 : FROM php:7.0-apache
--> 23f9c84560a6
Step 2/2 : COPY content/ /var/www/html/
--> e5c8e9bde9d6
Removing intermediate container 0c581912e97a
Successfully built e5c8e9bde9d6
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host.
All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recom
mended to double check and reset permissions for sensitive files and directories.
```

Ensuite, nous pouvons ainsi tester le bon fonctionnement de notre image en lançant un container.

```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-php-image (master)
$ docker run -p 9090:80 res/apache_php
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Sun May 28 16:40:29.859767 2017] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.10 (Debian) PHP/7.0.19 configured -- resuming normal operations
[Sun May 28 16:40:29.866778 2017] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
192.168.99.1 - - [28/May/2017:16:40:47 +0000] "GET / HTTP/1.1" 200 432 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36"
```



On remarque donc que notre image fonctionne bien et que le contenu de notre fichier index.html est bel et bien affiché dans le navigateur.

La première partie terminée, nous allons pusher notre image dans notre repo sur la branche :

- Git add docker-images/
- Git commit -m « Docker image 'static HTML with apache+php' working and validated”
- Git push origin **fb-apache-static**

Branch: fb-apache-stat... New pull request Create new file Upload files Find file Clone or download

This branch is 1 commit ahead of master. Pull request Compare

silverkameni Docker image static HTML with apache+php working and validated Latest commit 128e290 a day ago

docker-images/apache-php-image	Docker image static HTML with apache+php working and validated	a day ago
LICENSE	Initial commit	a day ago
README.md	Initial commit	a day ago

Step 2: Dynamic HTTP server with express.js

B. Critères d'acceptation

- ❖ Vous avez un compte GitHub avec tout le nécessaire pour construire l'image Docker.

Pour la réalisation de cette partie, nous travaillerons toujours sur notre même repo, et comme dans la partie précédente, nous allons créer une nouvelle branche : **git checkout -b fb-express-dynamic**

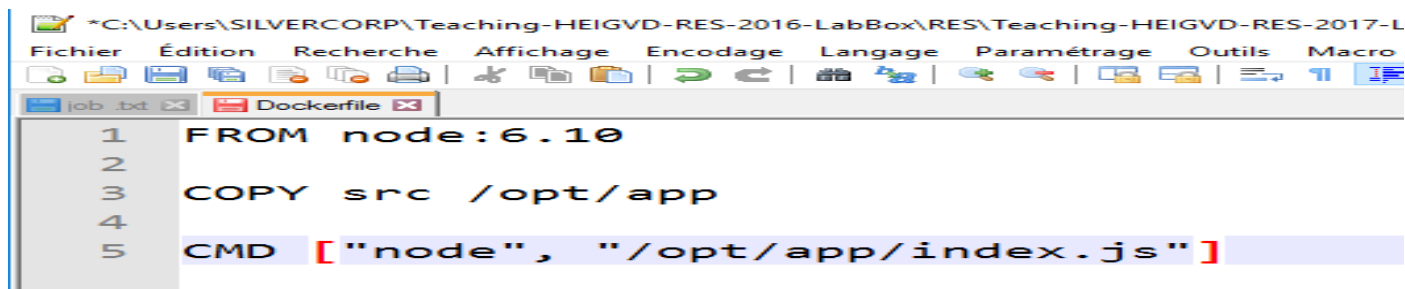
- ❖ Vous faites une démo, où vous construisez l'image, exécutez un conteneur et accédez au contenu d'un navigateur.

Préparation :

Une fois notre branche créée, nous allons comme dans la manipulation précédente, créer un dossier (**express-image**) qui contiendra tous les éléments nécessaires pour la mise en place de notre image.

Dans docker hub, nous allons choisir notre image officielle de node et nous choisirons une version stable qui dans notre cas sont la version **V6.10.3 LTS**.

Dans le fichier Dockerfile que nous allons créer dans notre dossier express-image, nous allons y entrer les instructions suivantes :

A screenshot of a text editor window showing a Dockerfile. The window title is "*C:\Users\SILVERCORP\Teaching-HEIGVD-RES-2016-LabBox\RES\Teaching-HEIGVD-RES-2017-L". The menu bar includes "Fichier", "Édition", "Recherche", "Affichage", "Encodage", "Langage", "Paramétrage", "Outils", and "Macro". The toolbar shows various icons for file operations. The Dockerfile content is as follows:

```
1 FROM node:6.10
2
3 COPY src /opt/app
4
5 CMD ["node", "/opt/app/index.js"]
```

Ce qui signifie que nous allons nous baser sur la version 6.10 de node pour construire notre image, puis que nous copierons les fichiers du répertoire **src** dans le dossier **/opt/app** de notre container. Ainsi, à chaque fois que nous lancerons un container, il exécutera le fichier « **index.js** ».

Puis dans notre dossier express-image, nous allons également créer le dossier « **src** » et à l'intérieur de ce dossier, nous exécuterons la commande : **npm init**

```
MINGW64:/c:/Users/SILVERCORP/Teaching-HEIGVD-RES-2016-LabBox/REs/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/express-image (fb-express-dynamic)
$ cd src/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/REs/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/express-image/src (fb-express-dynamic)
$ ls

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/REs/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/express-image/src (fb-express-dynamic)
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (src) students
version: (1.0.0) 0.1.0
description: just for a demo
entry point: (index.js)
test command:
git repository:
keywords:
author: silver kameni
license: (ISC)
About to write to C:\Users\SILVERCORP\Teaching-HEIGVD-RES-2016-LabBox\REs\Teaching-HEIGVD-RES-2017-Labo-HTTPInfra\docker-images\express-image\src\package.json:
{
  "name": "students",
  "version": "0.1.0",
  "description": "just for a demo ",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "silver kameni",
  "license": "ISC"
}

Is this ok? (yes) y
```

Pour la mise en place de notre image sous node, nous allons utiliser un module appelé « **chance** » qui nous permettra de générer toute sorte de donnée aléatoire.

Nous allons donc construire notre fichier **index.js** (pour qu'il nous génère un mot et une phrase aléatoire à chaque fois qu'on démarrera un container).

Pour ce faire, nous allons modifier notre fichier index.js comme ceci :

```

1  var Chance = require('chance');
2  var chance = new Chance();
3
4  var express = require('express');
5  var app = express();
6
7  app.get('/', function (req, res) {
8    res.send(generateText());
9  });
10
11 app.listen(3000, function () {
12   console.log('Accepting HTTP requests on port 3000!');
13 });
14
15 function generateText(){
16   var numberText = chance.integer({ min: 0, max: 10});
17   console.log(numberText);
18
19   var Textarea = [];
20
21   for(var i = 0; i < numberText; i++){
22     var word = chance.word();
23     var phrase = chance.sentence();
24
25     Textarea.push({
26       word: word,
27       phrase: phrase
28     });
29
30   };
31   console.log(Textarea);
32   return Textarea;
33 }

```

Puis nous allons builder notre image docker ensuite nous écouterons sur le **port 3000** et dans un autre terminal, nous allons établir une connexion sur le **port 3000** en local et enverrons par la suite une requête **GET**.

Build Image :

```

STILVERCORP@DESKTOP-SS97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/express-image (fb-dynamic-configuration)
$ docker build -t res/express_students .
Sending build context to Docker daemon  2.239MB
Step 1/4 : FROM node:6.10
--> 3f3928767182
Step 2/4 : RUN apt-get update && apt-get install -y vim
--> Using cache
--> 601ccecbe3a1
Step 3/4 : COPY src /opt/app
--> Using cache
--> b30b82e341ba
Step 4/4 : CMD node /opt/app/index.js
--> Using cache
--> f29223cb71c9
Successfully built f29223cb71c9
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.

STILVERCORP@DESKTOP-SS97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/express-image (fb-dynamic-configuration)
$

```

Run Image :

```
Sélection MINGW64/c/Users/SILVERCORP/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/expre...
phrase: 'Lecmozih vinnel emazu zeboez tuvigo osezevis ta hefji dak nejelop pudvu tised.' } ]
SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-
images/express-image/src (fb-express-dynamic)
$
SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-
images/express-image/src (fb-express-dynamic)
$ docker run -p 9090:3000 res/express students
Accepting HTTP requests on port 3000!
[ { word: 'los',
  phrase: 'Weeco tadfu sizuc gafob jevfihki afutaek taj utiuspel nahbu jajore enlus cocudi.' },
  { word: 'lepa',
  phrase: 'Tipoti silino enubibe fov hejnikwov niforfuz futzigom jap efkodo unuwomvu hobcefud ugoucwi zeg nic nuadu ko
kupjec uksuz sulur.' } ]

GET / HTTP/1.0
Host: 192.168.100.99
docker@default:~$ telnet 192.168.100.99 3000
telnet: can't connect to remote host (192.168.99.100): Connection refused
docker@default:~$ telnet 192.168.99.100 3000
telnet: can't connect to remote host (192.168.99.100): Connection refused
docker@default:~$ telnet 192.168.99.100 9090
GET / HTTP/1.0
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 262
ETag: W/"106-kkHQ7pRfSxb2qUeZIKvYrYZw/jg"
Date: Mon, 05 Jun 2017 19:25:53 GMT
Connection: close

[{"word":"los","phrase":"Weeco tadfu sizuc gafob jevfihki afutaek taj utiuspel nahbu jajore enlus cocudi."}, {"word":"lep
a","phrase":"Tipoti silino enubibe fov hejnikwov niforfuz futzigom jap efkodo unuwomvu hobcefud ugoucwi zeg nic nuadu ko
kupjec uksuz sulur."}]Connection closed by foreign host
docker@default:~$
```

```
(2) V x Softi x libra x Labo x Char x EX Exen x 192. x
192.168.99.100:9090
[{"word":"omopzej","phrase":"Diu su muwbiad sobti urgornem few rekbiv zacacce wawpizdi me lomo vejsabse
zoopri faf liben posafe hanafza."}, {"word":"gos","phrase":"Regcah wozooona cuvegok bukse law giskarsu gecjut
farfu ij loze umtihoku fe zu ho jikdig sizviufo teoz."}, {"word":"jovis","phrase":"Zu pu gu amu niadi lezufif
ze zase milhu ziv ekisawi can cabip oslu."}, {"word":"sukbiwul","phrase":"Turotrul le ge ru gooh bulem zasuj
geba edo pu etebutmij iv bebiwupe fah."}, {"word":"ejluh","phrase":"Curritej siwzola buh heche jevewul lut
ewgon mubvas aj cota wuize igo nefe."}, {"word":"na","phrase":"Wifipe odraktes lep badzo wo kiki jidve mel cu
ziw nugrihiy ki izuh nec."}, {"word":"alruk","phrase":"Nerapu vijagzot zolsoepi wab in utimal fade kezdoc
gilde ureribol si basjihar tute zedubsej odjifzuh."}]
```

En exécutant une requête GET vers notre serveur en utilisant postman, nous obtenons :

```
Postman
File Edit View Collection History Help
Runner Import Builder Team Library SYNC OFF
http://192.168.99.100 x + No Environment
GET http://192.168.99.100:9090 Params Send
Authorization Headers Body Pre-request Script Tests
Type No Auth
Body Cookies Headers (6) Tests Status: 200 OK Time: 125 ms
Pretty Raw Preview JSON
1 [
2 {
3   "word": "ihkadi",
4   "phrase": "Laros hehonbin ok walwugpah cur pacrig omowuabu fotenom judmek anali pujtino ri."
5 },
6 {
7   "word": "gumwu",
8   "phrase": "Curosit jacluki piti maci ot uniji nu cof du bog dud melofade sidu pupwi mog."
9 },
10 {
11   "word": "cufu",
12   "phrase": "Uglal menruk solifa awemibrib hun tez jalut vivakju bijimep vamefipoj botevuug hevpes mugisel lae
13 }
14 ]
```

Step 3: Reverse proxy with apache (static configuration)

Pour la réalisation de cette partie, nous travaillerons toujours sur notre même repo, et comme dans la partie précédente, nous allons créer une nouvelle branche : **git checkout -b fb-apache-reverse-proxy**

Préparation :

Après avoir tué tous les containers en cours d'exécution, nous allons dans notre dossier docker-images, afin de créer le répertoire dans lequel sera stocké tous les éléments nécessaire au bon fonctionnement de notre reverse proxy.

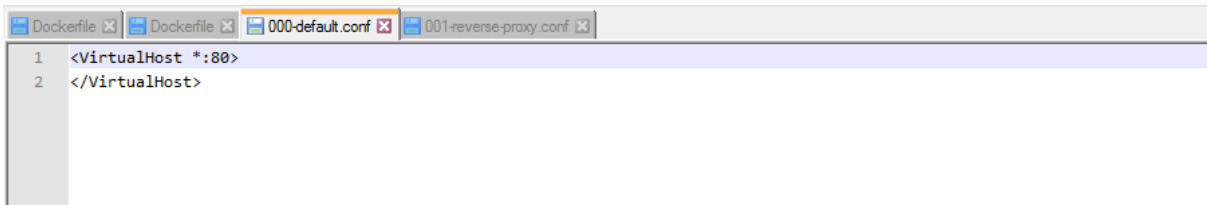
```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images (fb-express-dynamic)
$ ls
apache-php-image/  apache-reverse-proxy/  express-image/
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images (fb-express-dynamic)
$ cd apache-reverse-proxy/
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-express-dynamic)
$ ls
Dockerfile
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-express-dynamic)
$ mkdir conf
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-express-dynamic)
$ mkdir sites-available
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-express-dynamic)
$ touch sites-available/000-default.conf
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-express-dynamic)
$ touch sites-available/001-reverse-proxy.conf
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-express-dynamic)
$
```

Nous allons ainsi modifier le Dockerfile de notre future image :

```
Dockerfile x Dockerfile x 000-default.conf x 001-reverse-proxy.conf x
1 FROM php:7.0-apache
2
3 COPY conf/ /etc/apache2
4
5 RUN a2enmod proxy proxy_http
6 RUN a2ensite 000-* 001-*
```

Puis nous allons remplir nos fichiers de configuration afin que notre reverse proxy agisse comme nous le souhaitons ; Dans le fichier 001-reverse-proxy.conf, nous utiliserons la spécification « ProxyPass » et « ProxyPassReverse » afin de demander à notre reverse proxy d'accéder au container crée à partir d'express_students si il reçoit une requête de la forme « **GET /api/students/ HTTP/1.0** » et d'accéder au container crée à partir de apache_php si il reçoit une requête de la forme : « **GET / HTTP/1.0** »

```
Dockerfile x Dockerfile x 000-default.conf x 001-reverse-proxy.conf x
1 <VirtualHost *:80>
2     ServerName demo.res.ch
3
4     #ErrorLog ${APACHE_LOG_DIR}/error.log
5     #CustomLog ${APACHE_LOG_DIR}/access.log combined
6
7     ProxyPass "/api/students/" "http://172.17.0.4:3000/"
8     ProxyPassReverse "/api/students/" "http://172.17.0.4:3000/"
9
10    ProxyPass "/" "http://172.17.0.2:80/"
11    ProxyPassReverse "/" "http://172.17.0.2:80/"
12 </VirtualHost>
```



Puisque nous avons tous les fichiers nécessaires pour construire notre image apache-reverse-proxy, nous allons le builder, et démarrer 03 conteneurs donc un sur **apache_rp**, un autre sur **express_students**, puis un autre sur **apache_php**,

Docker run -d --name apache_static res/apache_php

Docker run -d res/apache_rp

Docker run -d --name express_dynamic res/express_students

Puis nous allons grâce à la commande **Docker inspect**, trouver les adresses IP de nos conteneurs.

```
TEACHING-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-apache-reverse-proxy)
$ docker build -t res/apache_rp .
Sending build context to Docker daemon 5.12kB
Step 1/4 : FROM php:7.0-apache
--> 23f9e86506a6
Step 2/4 : COPY conf/ /etc/apache2
--> 44a168a2750b
Removing intermediate container 1e1f8e69611b
Step 3/4 : RUN a2enmod proxy proxy_http
--> Running in 5f69e6daa6cd
Enabling module proxy.
Considering dependency proxy for proxy_http:
Module proxy already enabled
Enabling module proxy_http.
To activate the new configuration, you need to run:
    service apache2 restart
--> 11063bc3118d
Removing intermediate container 5f69e6daa6cd
Step 4/4 : RUN a2ensite 000.* 001.*
--> Running in 83ada4ef786
Site 000-default already enabled
Enabling site 001-reverse-proxy.
To activate the new configuration, you need to run:
    service apache2 reload
--> 6163e60cf250
Removing intermediate container 83ada4ef786
Successfully built 6163e60cf250
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-apache-reverse-proxy)
$ docker run -p 8080:80 res/apache_rp
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.3. Set the 'ServerName' directive globally to suppress this message
[Thu Jun 08 23:30:40.219579 2017] [pm:prefork:notice] [pid 1] AH00662: Apache/2.4.18 (Debian) PHP/7.0.19 configured -- resuming normal operations
[Thu Jun 08 23:30:40.219857 2017] [core:notice] [pid 1] AH00994: Command line: 'apache2 -D FOREGROUND'
demo.res.ch:80 192.168.99.1 - - [08/Jun/2017:23:30:46 +0000] "GET /api/students/ HTTP/1.0" 200 1108 "-" "-"
```

```
SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                               NAMES
554f323a2f8a        res/express_students "node /opt/app/ind..." About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp               express_dynamic
e3d92417ad4c        res/apache_php      "docker-php-entryp..." 3 minutes ago       Up 3 minutes        0.0.0.0:8080->80/tcp               apache_static
9dcf2f5d1e4e        res/apache_rp        "docker-php-entryp..." 10 minutes ago      Up 10 minutes        0.0.0.0:8080->80/tcp               stupefied_stonebraker

SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$ docker inspect express_dynamic | grep -i "ipaddress"
unknown shorthand flag: 'i' in -i
See 'docker inspect --help'.

SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$ docker inspect express_dynamic | grep -i "ipaddress"
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.4",
    "IPAddress": "172.17.0.4",

SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$ docker inspect apache_static | grep -i "ipaddress"
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.2",
    "IPAddress": "172.17.0.2",

SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$ docker inspect apache_rp | grep -i "ipaddress"
Error: No such object: apache_rp

SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$ docker inspect stupefied_stonebraker | grep -i "ipaddress"
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.3",
    "IPAddress": "172.17.0.3",
```

Afin de tester le bon fonctionnement de notre reverse proxy, nous procéder comme suit :

```
SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$ telnet 192.168.99.100 8080
Trying 192.168.99.100...
Connected to 192.168.99.100.
Escape character is '^]'.

GET /api/students/ HTTP/1.0
Host: demo.res.ch

HTTP/1.1 200 OK
Date: Thu, 08 Jun 2017 23:30:46 GMT
Server: Apache/2.4.18 (Debian)
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 867
ETag: W/"363-N880ZX5onSMf1tIpaSrnVgQKThs"
Connection: close

[{"word": "cuojodas", "phrase": "Ru kanicfo pun ku no tuhupjug wegoveuvu mun obsit tama ukeozutug gatv jazvanab."}, {"word": "dicegu", "phrase": "Ima cumbuwut gauh vu vop roffuw zoniz ah helpitef vooca se han nalcavu acibega keotobij padvliw tear fo torvejlo."}, {"word": "ovu", "phrase": "Ti odapako tohibozo cu kom dovoun af reh pizlaun guco afretofo pirwanen gi."}, {"word": "vasif", "phrase": "Be ire ohto u tbn firke bdiplu iva kuramne cazub gagu cavi ombasfif cahpozan weohaho vovzec rasu beokke."}, {"word": "basa", "phrase": "Farfa vuru ijejud pur vujan zoape egret ku imeju raltemvom hehepted guojo zamep ijo lolejeh za getlofar latro ujrabjos."}, {"word": "garkunbo", "phrase": "Bil celutatu voke nifa bomus fewsajtu favco no ditjiewu tofzi puhzivze te uma ascetur guheewe."}, {"word": "ewe", "phrase": "Upuuu f rigaludu so uzivitlo tam gejerini imziawo re detikib nadeda gilupjom takmi."}]Connection closed by foreign host.

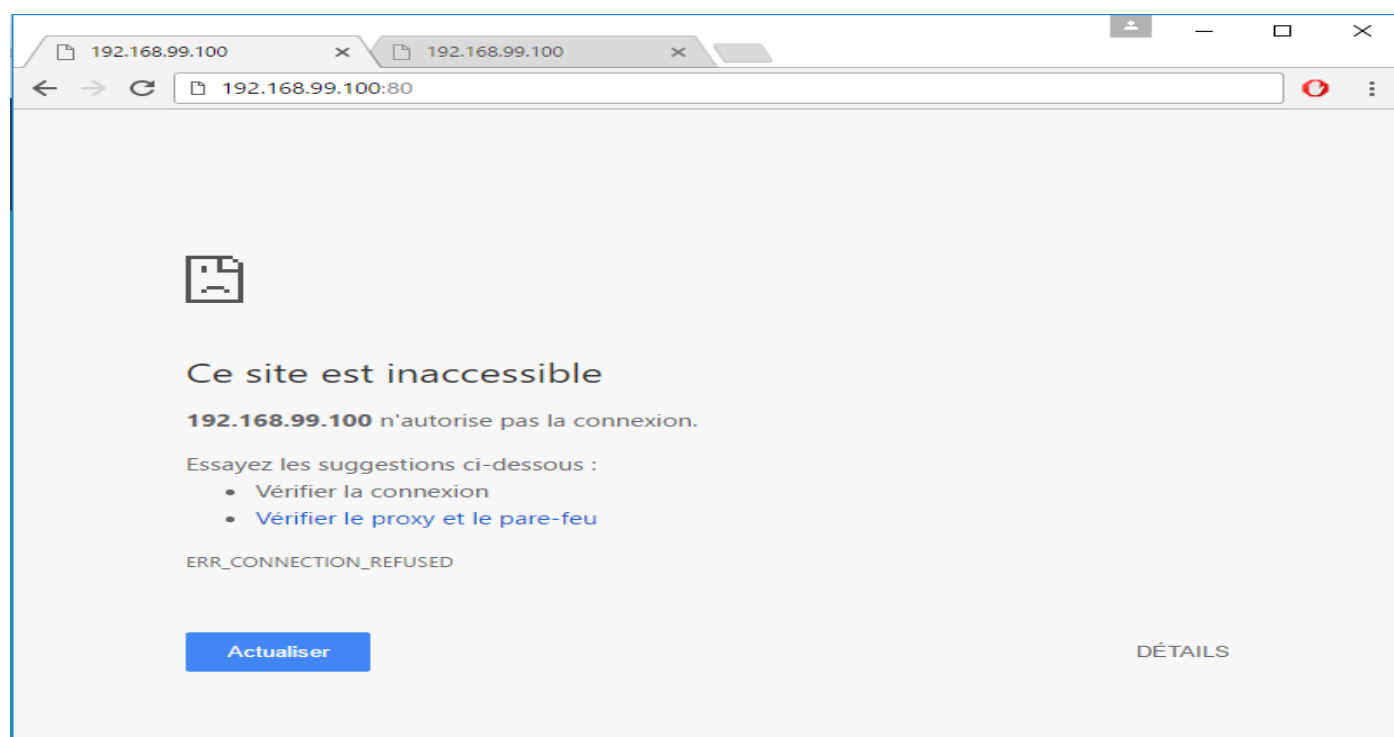
SILVERCORP@DESKTOP-5597LLH MINGW64 ~
$
```


Puis via notre navigateur, nous allons atteindre notre container basé sur express_students :



- ❖ You can explain and prove that the static and dynamic servers cannot be reached directly (reverse proxy is a single entry point in the infra).

Depuis l'extérieur, le seul point d'entrée est le reverse proxy, car le port mapping a été réalisé uniquement sur le containers du reverse proxy et c'est dans les fichiers de configs du reverse proxy que nous avons spécifié que selon les requêtes GET, il devait redirigé cela soit vers le serveur apache statique ou dynamique.





❖ You are able to explain why the static configuration is fragile and needs to be improved.

La configuration statique est fragile car les adresses ip sont hardcoded et rien ne garantit que nos containers auront toujours la même adresse à chaque fois qu'on les démarre.

Step 4: AJAX requests with JQuery

Pour la réalisation de cette partie, nous travaillerons toujours sur notre même repo, et comme dans la partie précédente, nous allons créer une nouvelle branche : **git checkout -b fb-ajax-jquery**

Préparations :

- **Configuration, build et run de l'image apache-php-image**

```
Dockerfile x Dockerfile x 000-default.conf x 001-reverse-proxy.conf x
1 FROM php:7.0-apache
2
3 RUN apt-get update && \
4     apt-get install -y vim
5
6 COPY content/ /var/www/html/
```

```
-Labo-HTTPInfra/docker-images (fb-apache-reverse-proxy)
$ cd apache-php-image/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017
-Labo-HTTPInfra/docker-images/apache-php-image (fb-apache-reverse-proxy)
$ ls
content/ Dockerfile

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017
-Labo-HTTPInfra/docker-images/apache-php-image (fb-apache-reverse-proxy)
$ docker build -t res/apache_php .
Sending build context to Docker daemon 2.435MB
Step 1/3 : FROM php:7.0-apache
--> 23f9c84560a6
Step 2/3 : RUN apt-get update && apt-get install -y vim
--> Running in 31b495de6365
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]
Get:2 http://security.debian.org jessie/updates/main amd64 Packages [521 kB]
```

- Configuration, build et run de l'image reverse proxy

```
Dockerfile x Dockerfile x 000-default.conf x 001-reverse-proxy.conf x hosts x
1 FROM php:7.0-apache
2
3 RUN apt-get update && \
4     apt-get install -y vim
5
6 COPY conf/ /etc/apache2
7
8 RUN a2enmod proxy proxy_http
9 RUN a2ensite 000-* 001-*
```

```
-Labo-HTTPInfra/docker-images (fb-apache-reverse-proxy)
$ cd apache-reverse-proxy/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017
-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-apache-reverse-proxy)
$ ls
conf/ Dockerfile

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017
-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-apache-reverse-proxy)
$ docker build -t res/apache_rp .
Sending build context to Docker daemon 5.12kB
Step 1/5 : FROM php:7.0-apache
--> 23f9c84560a6
Step 2/5 : RUN apt-get update && apt-get install -y vim
--> Using cache
--> 01e28d327de6
Step 3/5 : COPY conf/ /etc/apache2
--> 410c5fa0bf12
Removing intermediate container 8e46fa1b19b0
Step 4/5 : RUN a2enmod proxy proxy_http
--> Running in a0fff42b3cb8
Enabling module proxy.
```

```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017
-Labo-HTTPInfra/docker-images/apache-reverse-proxy (fb-apache-reverse-proxy)
$ docker run -it res/apache_rp /bin/bash
root@0240d2ba0da5:/var/www/html# vi
root@0240d2ba0da5:/var/www/html#
```

- Configuration, build et run de l'image express image

```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images (fb-apache-reverse-proxy)
$ cd express-image/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/express-image (fb-apache-reverse-proxy)
$ ls
Dockerfile  src/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra/docker-images/express-image (fb-apache-reverse-proxy)
$ docker build -t res/express_students .
Sending build context to Docker daemon 2.239MB
Step 1/4 : FROM node:6.10
--> 3f3928767182
Step 2/4 : RUN apt-get update && apt-get install -y vim
--> Running in 783bfbf7e65f
```

```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker ps -a
CONTAINER ID        IMAGE                                     COMMAND                  CREATED             STATUS              PORTS              NAMES
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker run -d --name apache_static res/apache_php
103d57960332219e705c8ade9edb2b5bf9e7b9314d62e3296fd780183d01af4
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker run -d -p 8080:80 --name apache_rp res/apache_rp
e2b1c7e830ec2a08401e1cfbe18049645badd9a3e0bcd66c069088d7e72e923
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker run -d --name express_dynamic res/express_students
6233f35c0505000c47b6da30615e225d9c716eb409586910237146c3905abb14
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker inspect apache_static !grep -i "ipaddress"
bash: !grep: event not found
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker inspect apache_static | grep -i "ipaddress"
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.2",
    "IPAddress": "172.17.0.2",
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker inspect apache_rp | grep -i "ipaddress"
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.3",
    "IPAddress": "172.17.0.3",
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$ docker inspect express_dynamic | grep -i "ipaddress"
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.4",
    "IPAddress": "172.17.0.4",
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/res/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions (fb-ajax-jquery)
$
```

Dans notre dossier apache-php-image/content/ , nous allons modifier le fichier index.html afin qu'il puisse executer notre script « students.js ».

```

521
522     <!-- Bootstrap Core JavaScript -->
523     <script src="vendor/bootstrap/js/bootstrap.min.js"></script>
524
525     <!-- Plugin JavaScript -->
526     <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.3/jquery.easing.min.js"></script>
527
528     <!-- Contact Form JavaScript -->
529     <script src="js/jqBootstrapValidation.js"></script>
530     <script src="js/contact_me.js"></script>
531
532     <!-- Theme JavaScript -->
533     <script src="js/freelancer.min.js"></script>
534
535     <!-- Custom Theme JavaScript -->
536     <script src="js/students.js"></script>
537
538 </body>
539
540 </html>

```

- Création de notre script students.js :

```

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutio
ns (fb-ajax-jquery)
$ cd docker-images/apache-php-image/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutio
ns/docker-images/apache-php-image (fb-ajax-jquery)
$ cd content/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutio
ns/docker-images/apache-php-image/content (fb-ajax-jquery)
$ ls
css/      img/      js/       LICENSE  package.json  vendor/
gulpfile.js  index.html  less/    mail/    README.md

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutio
ns/docker-images/apache-php-image/content (fb-ajax-jquery)
$ cd js/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutio
ns/docker-images/apache-php-image/content/js (fb-ajax-jquery)
$ ls
contact_me.js  freelancer.js  freelancer.min.js  jqBootstrapValidation.js

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutio
ns/docker-images/apache-php-image/content/js (fb-ajax-jquery)
$ touch students.js

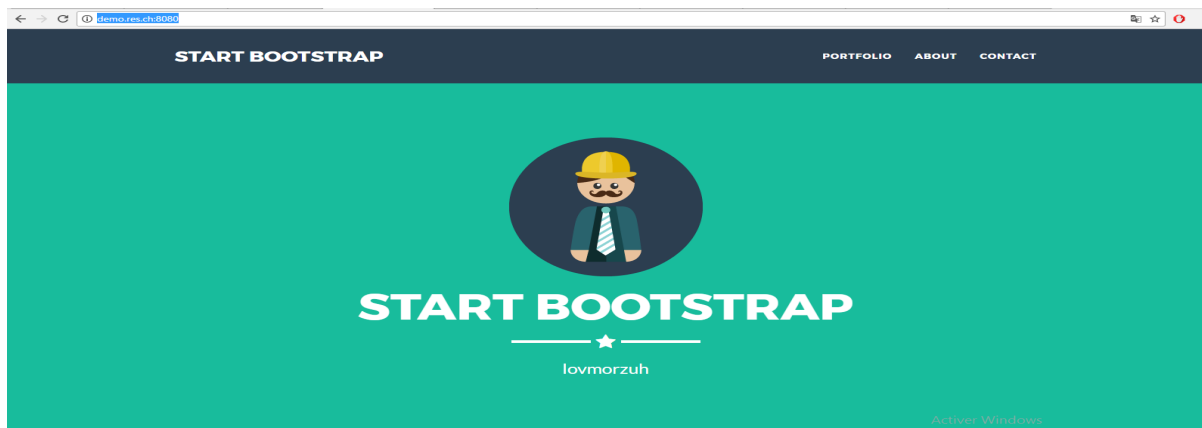
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutio
ns/docker-images/apache-php-image/content/js (fb-ajax-jquery)
$

```

```
students.js | README.md | Dockerfile | Dockerfile | Dockerfile | 000-default.conf | 001-reverse-proxy.conf | index.js
1  $(function() {
2      console.log("Loading Text");
3
4      function loadText() {
5          $.getJSON( "/api/students/", function( Textarea ) {
6              console.log(Textarea);
7              var message = "No word is here";
8              if ( Textarea.length > 0 ) {
9                  message = Textarea[0].word;
10             }
11             $(".skills").text(message);
12         });
13     };
14
15     loadText();
16     setInterval( loadText, 2000);
17 });
18
```

Le but de notre script students.js est de remplacer la ligne html qui se trouve au niveau de la classe « skills » par un mot qui sera généré de manière aléatoire toutes les 2000 ms.

Resultat :



Step 5: Dynamic reverse proxy configuration

Pour la réalisation de cette partie, nous travaillerons toujours sur notre même repo, et comme dans la partie précédente, nous allons créer une nouvelle branche : **git checkout -b fb-dynamic-configuration**

Tout d'abord, on va tuer tous containers actifs :

```
SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
bfed23b12335       res/apache_php     "docker-php-entryp..." 2 hours ago         Up 2 hours          80/tcp             apache_static
6233f35c6585       res/express_students "node /opt/app/ind..." 9 hours ago         Up 9 hours          0.0.0.0:8080->80/tcp express_dynamic
e2b1c7e830ec       res/apache_rp      "docker-php-entryp..." 9 hours ago         Up 9 hours          0.0.0.0:8080->80/tcp apache_rp

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$ docker kill apache_rp
Error response from daemon: Cannot kill container express_students: No such container: express_students

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$ docker kill express_dynamic
Error response from daemon: Cannot kill container apache_php: No such container: apache_php

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
bfed23b12335       res/apache_php     "docker-php-entryp..." 2 hours ago         Up 2 hours          80/tcp             apache_static

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$ docker kill apache_php
Error response from daemon: Cannot kill container apache_php: No such container: apache_php

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$ docker kill apache_static
Error response from daemon: Cannot kill container apache_php: No such container: apache_php

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$ docker rm $(docker ps -a -q)
bfed23b12335
6233f35c6585
e2b1c7e830ec

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-php-image (fb-ajax-jquery)
$
```

Puis nous allons créer notre nouvelle branche, ainsi que tous les fichiers nécessaires dans le dossier apache-reverse-proxy et enfin nous allons rebuilder notre image apache_rp.

Contenu des fichiers :

```
1 #!/bin/bash
2 set -e
3
4 # Add setup for RES lab
5 echo "setup for the RES lab..."
6 echo "Static app URL: $STATIC_APP"
7 echo "Dynamic app URL: $DYNAMIC_APP"
8
9 # Note: we don't just use "apache2ctl" here because it itself is just a shell-script wrapper around apache2 which provides e
10 # (also, when run as "apache2ctl <apache args>", it does not use "exec", which leaves an undesirable resident shell process)
11
12 : "${APACHE_CONFDIR:=/etc/apache2}"
13 : "${APACHE_ENVVARS:=${APACHE_CONFDIR}/envvars}"
14 if test -f "$APACHE_ENVVARS"; then
15     . "$APACHE_ENVVARS"
16 fi
```

```
SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-ajax-jquery)
$ git checkout -b fb-dynamic-configuration
Switched to a new branch 'fb-dynamic-configuration'
M README.md
M Dockerfile

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ ls
conf/ Dockerfile

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ touch apache2-foreground
$ ls
apache2-foreground* conf/ Dockerfile

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ chmod 755 apache2-foreground

SILVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker build -t res/apache_rp .
Sending build context to Docker daemon 7.168kB
Step 1/6 : FROM php:7.0-apache
--> 23f9c84509a6
Step 2/6 : RUN apt-get update && apt-get install -y vim
--> Using cache
--> 01e28d327de6
Step 3/6 : COPY apache2-foreground /usr/local/bin/
```

Puis nous allons lancer un container reverse proxy en lui passant les adresses IP de nos containers apache statique et dynamique.

Mais avant cela, nous devons créer et modifier notre fichier de config php qui nous sera nécessaire afin de récupérer les adresses IP de nos containers depuis l'extérieur ce qui nous permet de ne plus avoir des adresses ip hardcodés dans notre fichier de configuration reverse proxy.

- Création de config-template.php et modification :

```
SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ ls
apache2-foreground*  conf/  Dockerfile

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ mkdir template

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ cd template/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy/template (fb-dynamic-configuration)
$ touch config-template.php

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy/template (fb-dynamic-configuration)
$
```



```
1 <?php
2 $dynamic_app = getenv('DYNAMIC_APP');
3 $static_app = getenv('STATIC_APP');
4 ?>
5 <VirtualHost *:80>
6     ServerName demo.res.ch
7
8     ProxyPass '/api/students/' 'http://<?php print "$dynamic_app"?>/'
9     ProxyPassReverse '/api/students/' 'http://<?php print "$dynamic_app"?>/'
10
11     ProxyPass '/' 'http://<?php print "$static_app"?>/'
12     ProxyPassReverse '/' 'http://<?php print "$static_app"?>/'
13 </VirtualHost>
```

- Modification du fichier apache2-foreground


```

1 #!/bin/bash
2 set -e
3
4 # Add setup for RES lab
5 echo "setup for the RES lab...."
6 echo "Static app URL: $STATIC_APP"
7 echo "Dynamic app URL: $DYNAMIC_APP"
8 php /var/apache2/templates/config-template.php > /etc/apache2/sites-available/001-reverse-proxy.conf
9
10 # Note: we don't just use "apache2ctl" here because it itself is just a shell-script wrapper around apache2 which provides e
11 # (also, when run as "apache2ctl <apache args>", it does not use "exec", which leaves an undesirable resident shell process)
12
13 : "${APACHE_CONFDIR:=/etc/apache2}"
14 : "${APACHE_ENVVARS:=${APACHE_CONFDIR}/envvars}"
15 if test -f "$APACHE_ENVVARS"; then
16     . "$APACHE_ENVVARS"
17 fi
18

```

- Build de notre image apache-reverse-proxy

```

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017
-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ ls
apache2-foreground*  conf/  Dockerfile  templates/

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017
-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker build -t res/apache_rp .
Sending build context to Docker daemon  9.216kB
Step 1/7 : FROM php:7.0-apache
--> 23f9c84560a6
Step 2/7 : RUN apt-get update && apt-get install -y vim
--> Using cache

```

A ce stade, nous a lancer notre container reverse proxy en lui passant donc les adresses IP de nos containers web statique et dynamique. Afin de prouver que cela ne se base pas sur des adresses ip hardcodés, nous allons lancer plusieurs serveur apache statique et dynamique et nous choisirons les adresses IP d'un serveur apache statique et un serveur apache dynamique que nous allons transmettre a notre container reverse proxy via la variable d'environnement.

```

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d res/apache_php
6289b9db33fe973b0fae66960c5130632368cf6da82fd7d98d7aa9f681a2034b4

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d res/apache_php
55a85624967a5110d3a30aa67b3613577a59a879403f0c1c9810c45dd47ef5f

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d res/apache_php
abcdd61a13bd63a45da69adf18f07c0c436e6f78e0c5b0740b5540d67554c67f

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d --name apache_static res/apache_php
900f9fc023db7f5984d0f9be554a5c2c2f3024a6eb150e396e4040bcb9c5a7c

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d res/express_students
1ec39ff59dd97e44da4b15863430b407e649c857b3bc8161e51a023f8f01b519

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d res/express_students
0d7a5b164ca0a733c09aaefee00ed12af40b1e3f876db4478a202a21b6a53f05

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d --name express_dynamic res/express_students
e6909b9b0041af2b520791d3a44e6e0e0db01089343d4d51298413901d5016

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker ps

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e6909b9b0041	res/express_students	"node /opt/app/ind..."	2 minutes ago	Up 2 minutes		express_dynamic
0d7a5b164ca8	res/express_students	"node /opt/app/ind..."	3 minutes ago	Up 3 minutes		sad_bardeen
1ec39ff59dd9	res/express_students	"node /opt/app/ind..."	3 minutes ago	Up 3 minutes		silly_benz
900f9fc023db	res/apache_php	"docker-php-entryp..."	4 minutes ago	Up 4 minutes	80/tcp	apache_static
abcdd61a13bd6	res/apache_php	"docker-php-entryp..."	4 minutes ago	Up 4 minutes	80/tcp	heuristic_swanson
55a85624967a	res/apache_php	"docker-php-entryp..."	4 minutes ago	Up 4 minutes	80/tcp	inspiring_bardeen
6289b9db33fe9	res/apache_php	"docker-php-entryp..."	4 minutes ago	Up 4 minutes	80/tcp	wizardly_austin

```

SILVERCORP@DESKTOP-5S97LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)

```

```
SLIVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker inspect express_dynamic | grep -i ipaddr
      "SecondaryIPAddresses": null,
      "IPAddress": "172.17.0.8",
      "IPAddress": "172.17.0.8",
    ]
  }
}

SLIVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker inspect apache_static | grep -i ipaddr
      "SecondaryIPAddresses": null,
      "IPAddress": "172.17.0.5",
      "IPAddress": "172.17.0.5",
    ]
  }
}

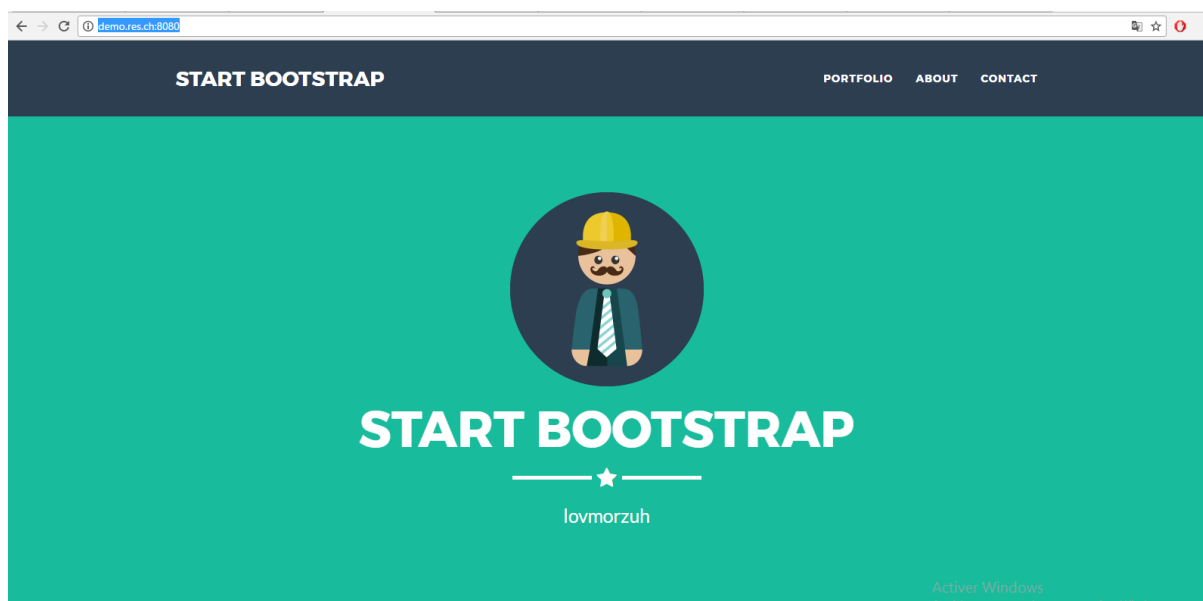
SLIVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker run -d -e STATIC_APP=172.17.0.5:80 -e DYNAMIC_APP=172.17.0.8:3000 --name apache_rp -p 8080:80 res/apache_rp
888:80 res/apache_rp
8c3e193d081a04230ed1cea4dc70a34e9b828938813854e2075519908a2dc5c

SLIVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
e3e193d081a04230ed1cea4dc70a34e9b828938813854e2075519908a2dc5c    res/apache_rp      "docker-php-entryp..." 11 seconds ago     Up 10 seconds      0.0.0.0:8080->80/tcp    apache_rp
e69098b90041       res/express_students "node /opt/app/ind..." 6 minutes ago      Up 6 minutes       80/tcp               express_dynamic
8d7a5b164ca0       res/express_students "node /opt/app/ind..." 7 minutes ago      Up 7 minutes       80/tcp               sad_bardeen
1ec39ff59d09       res/express_students "node /opt/app/ind..." 7 minutes ago      Up 7 minutes       80/tcp               silly_benz
9009f0c022db       res/apache_php      "docker-php-entryp..." 8 minutes ago      Up 8 minutes       80/tcp               apache_static
abcd01a13bd6       res/apache_php      "docker-php-entryp..." 9 minutes ago      Up 9 minutes       80/tcp               heuristic_swanson
55a85624967a       res/apache_php      "docker-php-entryp..." 9 minutes ago      Up 9 minutes       80/tcp               inspiring_bardeen
6289bdb33fe9       res/apache_php      "docker-php-entryp..." 9 minutes ago      Up 9 minutes       80/tcp               wizardly_austin

SLIVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$ docker logs apache_rp
setup for the RES lab...
Static app URL: 172.17.0.5:80
Dynamic app URL: 172.17.0.8:3000
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.9. Set the 'ServerName' directive globally to suppress this message
[Sun Jun 11 11:31:41.053525 2017] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.18 (Debian) PHP/7.0.19 configured -- resuming normal operations
[Sun Jun 11 11:31:41.058811 2017] [core:notice] [pid 1] AH00054: Command line: 'apache2 -D FOREGROUND'

SLIVERCORP@DESKTOP-5597LLH MINGW64 ~/Teaching-HEIGVD-RES-2016-LabBox/RES/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra-solutions/docker-images/apache-reverse-proxy (fb-dynamic-configuration)
$
```

Résultat :



CONCLUSION

Il était question pour nous de concevoir une application client-serveur en utilisant les images dockers encapsulant un serveur apache préconfiguré. Au terme de notre analyse, nous pouvons dire que ce laboratoire est particulièrement important pour nous. Car il permet de faire mettre en pratique les connaissances acquises et nous permet ainsi de voir de près tous les processus intervenant dans la communication entre un client et un serveur httpd.