# Rapport : laboratoire 5

Cours : RES

Emmanuel Schmid & Lemdjo Nzinke Marie Pascale

Professeur : Olivier Liechti

# Contents

# Rappel des 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. Pour ce faire, nous verrons que le serveur apache httpd peut agir en tant que serveur HTTP et en tant que proxy inverse. Nous verrons également que express.js est un framework JavaScript qui facilite l'écriture d'applications Web dynamiques.

Le deuxième objectif est de mettre en œuvre une application Web dynamique simple, mais complète. 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 (demandes AJAX) et récupérera le contenu généré dynamiquement.

Le troisième objectif est de pratiquer notre utilisation de Docker. Tous les composants de l'infrastructure Web seront emballés dans des images Docker personnalisées (nous créerons au moins 3 images différentes).

Source : Donnée laboratoire - Teaching-HEIGVD-RES-2017-Labo-HTTPInfra

# Environnement

OS : Windows 10 Pro

Docker : Docker toolbox (avec docker-machine)

Repo github : https://github.com/dbnsky/Teaching-HEIGVD-RES-2017-Labo-HTTPInfra

# Documentation générale

## Comment « build » une image Docker ?

Premièrement il faut créer un fichier « Dockerfile » à l'aide la commande « touch Dockerfle » dans le dossier désiré. Les containers utilisés dans ce laboratoire sont construits à l'aide d'image de base.

```
FROM php:7.0-apache
COPY src/ /var/www/html/
```

L'exemple ci-dessus représente un « dockerfile » utilisant une image de base, ainsi qu'une d'un site web dans la future image (dans le bon dossier). Commande pour build des images :

```
Docker build -t exemple .
```

L'option « -t » permet de spécifier un nom à l'image et le « . » spécifie que le container utilise le dossier courant comme racine du filesystème de celui-ci.

## Comment « run » une image Docker ?

Il existe plusieurs manières de lancer un container Docker. On a la possibilité de run le container en deamon, intéractif, avec ou sans manpping de port, etc.  A titre d'exemple voici un container avec un nom en mode deamon sur la base d'une image avec port mapping. Le serveur sera écouté à partir du port 8080 sur notre machine locale et via le port 80 sur notre machine docker.

```
Docker run -d -p 8080 :80 –name exempleRun exemple
```

## Obtention d'adresse IP

Pour obtenir l'adresse IP d'un container, il suffis de lancer la commande « docker ps » pour afficher les containers en cours d'exécution pour récupérer son nom et ensuite lancer la commande « docker inspect nameContainer | grep -i address » avec name : nom du container en question.

## Exécuter une commande sur un container en cours d'exécution

La commande exec de docker permet de lancer une commande directement dans le container. Voici un exemple d'utilisation : « docker exec -it containerName /bin/bash"

## Instalation Vim

Dans le terminal d'un container, il suffit de lancer :

- Apt-get update
- Apt-get install vim

Attention, si le container est redémarré l'installation sera perdu. Il est préférable de lancer l'installation à chaque démarrage du container via le Dockerfile. Ajouter dans le dockerfile : « RUN apt-get update && \ apt-get install -y vim »

# Etape 1

## Webcasts

- Labo HTTP (1): Serveur apache httpd "dockerisé" servant du contenu statique

Dans cette partie, il s'agit de configurer un serveur apache pour afficher du contenu static html. Nous ne l'installons pas directement sur notre machine, mais nous utilisons une image docker.

## Documentation

Github branch : fb-apache-static

Pour que le site web mise en place soit accessible en dehors de la docker-machine, il faut faire du port mapping (80:8080). Ainsi le contenue sera disponible via « 192.168.0.94 :80 ».

### Création & configuration de l'image Docker

L'image Docker créée à cette étape se base sur l'image officiel PHP dans sa version 7.0. L'image générée se nomme « res/apache_php » et charge le contenu du dossier src/ dans /var/www/html/ afin d'être visibile via son adresse IP.

### Apache Configuration

Les fichiers de configurations du service apache se trouve dans le dossier « /etc/apache2/ ». On y trouve le fichier de configuration principal « apche2.conf », ainsi que les dossiers « site-available » & « site-enable ». Ces dossiers contiennent des sous-fichier de configuration permettant la gestion des Virtual Hosts par exemple. Par défaut, il y a un seul host qui écoute sur le port 80. A travers le fichier « /etc/apache2/site-availaible/000-default.cong », on peut setter le dossier root, actuellement « /var/www/html/ ».

### Template Bootstrap

La mise en place d'un template bootstrap est facile, il suffit de :

- Télécharger les sources
- Déplacer les sources dans le bon dossier
- Build de l'image docker

Le Template gratuit « Landing Page » a été déployé.

# Etape 2 - Dynamic HTTP server with express.js

## Webcasts

- Labo HTTP (2a): Application node "dockerisée"
- Labo HTTP (2b): Application express "dockerisée"

Dans cette partie, nous présentons comment faire une application web dynamique à partir de node.js et docker, qui va retourner des données.

## Documentation

Github branch : fb-express-dynamic

Pour que récupérer les données au format Json en dehors de la docker-machine, il faut faire du port mapping (9090 :3000). Ainsi le contenue sera disponible via « 172.168.0.94:9090 ».

### Création & configuration de l'image Docker

L'image Docker créée à cette étape se base sur l'image officiel « node » dans sa version 4.8. L'image générée se nomme « res/express_students », charge le contenu du dossier « src/ » dans « opt/app » et exécute la commande «node /opt/app/index.js» à chaque fois qu'un container est lancé. La commande exécute un script en JavaScript.

### JavaScript

Lorsqu'on démarre une nouvelle application node.js, on utilise l'utilitaire « npm ». « npm  init » permet de gêner le package.json contenant les dépendances.

Le module utilisé lors du développement du script est le module « Chance ». Ce module permet de générer des données aléatoires. L'installation du module s'effectue en ligne de commande via l'utilitaire « npm » :

- Accéder au dossier contenant le package.json
- Lancer la commande : npm install -- save chance

La dépendance a été ajouté dans le package.json.

### Framework Express.js

Le script développé implémente un serveur et exécute des fonctions différentes selon l'url de la requête. Il est possible d'utiliser « node.js », mais le framework « express.js » facilite grandement le travail et simplifie le code. Il faut ajouter la dépendance, pour se faire se référer à l'étape précédente.

### Script

Le Script développé se base sur la solution proposée dans les webcast à la différence que les données renvoyées sont de types différents.

# Etape 3 - Reverse proxy with apache (static configuration)

## Webcasts

- Labo HTTP (3a): reverse proxy apache httpd dans Docker
- Labo HTTP (3b): reverse proxy apache httpd dans Docker
- Labo HTTP (3c): reverse proxy apache httpd dans Docker

Dans cette étape, nous installerons un proxy inverse pour communiquer avec nos deux serveurs depuis l'extérieur. Afin d'améliorer la sécurité et respecter la `same origin policy` (qui stipule qu'une requête ajax peut toujours communiquer avec un serveur dans le même domaine). Pour cette étape, nous devons configurer le conteneur pour cette tâche.

## Documentation

Github branch: fb-apache-reverse-proxy

Les attributions d'adresse IP est gérer par Docker et ne garantit pas qu'un container obtiendra toujours la même adresse. Le démarrage des 2 containers configurer préalablement ne spécifie pas de port mapping cette fois. Ils ne sont donc pas accessibles directement par l'extérieur. L'obtention de leur adresse IP est décrite dans la documentation générale.

Le contenu des 2 containers sont accessibles via 2 url différents à savoir « demo.res.ch » pour le contenu static et « demo.res.ch/api/students » pour le contenu dynamique.

La faiblesse de cette implémentation est l'« hardcodage » des adresses IP dans le reverser Proxy.

### Création & configuration de l'image Docker

L'image Docker créée à cette étape se base sur l'image officiel « php » dans sa version 7.0. L'image générée se nomme « res/apache_php », charge le contenu du dossier « conf/ » dans « /etc/apache2» et exécute les deux commandes suivantes :

| Commande | Description |
|---|---|
| A2enmod proxy proxy_http | Activer module supplémentaire d'apache |
| A2ensite 000-* 001-* | Activer les deux sites logiques |

Un serveur apache peut servir plusieurs sites logiques. Les commandes sont exécutées à chaque démarrage d'un container se servant de l'image. Les fichiers de configurations copié et les commandes exécuté lors du démarrage du container permet de mettre en place et configurer le reverse proxy.

### Configuration Reverse Proxy

L'image de base PHP utilisé permet d'activer le service proxy via la commande « a2enmod ». Ensuite, il faut activer les sites logiques (virtualhost) via la commande « a2ensite ».

| VirtualHost | Description |
|---|---|
| 000-default.conf | Host par défaut permet de blinder l'infrastructure. Aucun site n'est accessible si aucun « host » et spécifier dans l'en-tête de la requête (message d'erreur). |
| 001-reverse-proxy.conf | Règle de routage des requêtes : Choix du noeds à accéder selon l'url de la requête. |

Les règles sont écrites de la plus restrictive à la moins restrictive !

# Etape 4 - AJAX requests with JQuery

## Webcasts

- Labo HTTP (4): AJAX avec JQuery

L'objectif de cette partie est d'implémenter une requête ajax. On veut envoyer des requêtes pour récupérer le contenu de dynamique (express_students) et l'afficher dans notre page static.

## Documentation

Github branch : fb-ajax-jquery

Cette partie du laboratoire a nécessité l'installation de vim, la marche à suivre est disponible dans la documentation générale. Les manipulations sont disponibles dans la documentation générale.

Aucune image docker ne sera créé à cette étape, mais plusieurs modifications vont être effectué.

Modification res/apache_php

Le script récupérant les données Json se lance du côté de la page static (index.html). Il suffit d'ajouter la ligne suivante : `<script src="js/students.js"></script>.`

Il faut à présente créer le script dans le dossier « js », nommé students.js.

```
$(function() {
	console.log("Loading students");

	function loadStudents() {
		$.getJSON( "/api/students/", function( students ) {
			console.log(students);
			var message = "Nobody is here";
			if( students.length > 0 ) {
				message = students[0].name;
				}
				$(".skills").text(message);
			});
		};

		loadStudents();
		setInterval(loadStudents, 2000);
});
```

On remarque qu'il faut spécifier l'url à laquelle les données Json sont disponible, setter le texte de la balise implémentant la class « skills » et setter l'intervalle de temps entre le chargement d'une valeur différente.

# Etape 5 - Dynamic reverse proxy configuration

## Webcasts

- Labo HTTP (5a): configuration dynamique du reverse proxy
- Labo HTTP (5b): configuration dynamique du reverse proxy
- Labo HTTP (5c): configuration dynamique du reverse proxy
- Labo HTTP (5d): configuration dynamique du reverse proxy
- Labo HTTP (5e): configuration dynamique du reverse proxy

## Documentation

Github branch : fb-dynamic-configuraiton

On a relevé que la faiblesse de notre infrastructure était que les adresses IP des différents noeds était hardcodées. Le but de cette étape est de rendre la configuration dynamique.

### Passage d'une variable d'environnement lors du démarrage d'un container

Lorsqu'on démarre un container, on peut lui fournir en paramètre des variables d'environnement à l'aide du flag « -e » de la commande « docker run », exemple : « docker run -e HELLO=world -it res/apache_rp /bin/bash ». Ensuite, on peut les consulter à l'aide la commande « export ».

### Utilisation de PHP pour créer un template de configuration du reverse proxy

Après avoir analyser le dockerfile de l'image de base « PHP », on remarque l'appel du script « apache2-foreground ». On reprend le script initial au quel on ajoute :

```
echo "Setup for the RES lab..."
echo "Static app URL: $STATIC_APP"
echo "Dynamic app URL: $DYNAMIC_APP"
php /var/apache2/templates/config-template.php > '/etc/apache2/sites-
available/001-reverse-proxy.conf'
```

La dernière ligne permet d'exécuter le script php et de concaténer le résultat au fichier de configuration « 001-reverse-proxy.conf ».

Le template récupère les valeurs des variable d'environnement et géner les règles adéquates

```
<?php
        $dynamic_app = getenv('DYNAMIC_APP');
        $static_app  = getenv('STATIC_APP');
?>
<VirtualHost *:80>
    ServerName demo.res.ch

       ProxyPass         '/api/students/' 'http://<?php print "$dynamic_app";?>'
    ProxyPassReverse '/api/students/' 'http://<?php print "$dynamic_app";?>'

       ProxyPass         '/' 'http://<?php print "$static_app";?>/'
    ProxyPassReverse '/' 'http://<?php print "$static_app";?>/'
</VirtualHost>
```

# Conclusion

Nous avons beaucoup apprécié ce laboratoire qui nous a permis d'assimiler la théorie et de mettre en pratique nos connaissances. Nous avons trouvé très intéressante la mise en place d'une infrastructure à travers docker.