Configuration d’un environnement de développement professionnel.

Créé par : HENRY Alexis,

Le *01/06/2022*.

Modifié par : HENRY Alexis,

Le *01/06/2022*.

Version du document : v1.

Sommaire

Introduction

Cette documentation concerne la mise en place d'un environnement de développement professionnel. Plusieurs options existent pour permettre une mise en place d'un environnement de ce type… Comme Vagrant, Multipass, ou bien même l'utilisation de simples machines virtuelles créées à l'aide de VMWare ou VirtualBox.

Pour ce qui est de l'environnement utilisé tout au long de cette documentation, il a été mis en place à l'aide de Multipass. Le système d'exploitation utilisé est Ubuntu 20.04.

Qu'est-ce que Multipass ?

Multipass est une CLI (Interface en ligne de commande) qui permet de lancer et de gérer assez simplement des machines virtuelles, que ce soit sur Windows, Linux ou bien Mac.

Pourquoi Multipass ?

Multipass permet de générer très rapidement des machines virtuelles. Notamment grâce à sa prise en charge de cloud-init. Un outil qui permet de personnaliser les machines virtuelles au démarrage de celle-ci. Permettant aussi la configuration du stockage, ainsi que de la ram, dédiés à la machine virtuelle. La documentation ainsi que les différents forums autours de Multipass sont assez fournies. La plupart des problèmes pouvant survenir peuvent être résolus à l'aide d'une simple recherche sur internet. De ce fait nous pouvons facilement générer des machines avec les outils que nous avons besoins en cas de problème.

Environnements

Un environnement de développement professionnel est lui-même composé de plusieurs environnements. Cela permet de faire en sorte que les développeurs puissent vérifier que le code qu'ils ont fourni soit fonctionnel. Pour cela, un environnement dit professionnel, se devra de posséder les trois (au minimum) sous-environnements indiqués ci-dessous :

Environnement de développement

L'environnement de développement, comme son nom l'indique, celui à partir duquel le code sera produit. Dans ce cas de figure, nous allons cloner le repository sur la machine de développement, puis, à l'aide de Samba, ouvrir le code dans un éditeur, comme PHP Storm, Visual Studio Code, etc…

L'installation du package avahi-daemon va nous permettre d'accéder à la machine plus facilement, via le nom de celle-ci. Et ce sans avoir à modifier régulièrement le fichier host de Windows.

Dans le cas d'un environnement de développement, les installations suivantes devront être réalisées :

- Installation d'Apache,
- Installation de PHP,
- Installation de MariaDB,
- Installation de PhpMyAdmin,
- Installation de Composer,
- Installation de xDebug,
- Installation d'Avahi,
- Installation de Samba.

Environnement de tests/préproduction

L'environnement de tests, va permettre de tester le code qui a été réalisé lors du développement. Ainsi que de réaliser des tests afin de s'assurer qu'il n'y ait pas d'erreur dans le code. Cette machine va récupérer le code envoyé sur GitHub depuis la machine de développement, cela à l'aide de ce script. Après s'être assuré qu'aucune erreur ne soit présente, cette machine, s'occupera de mettre tout le code en production, à l'aide de ce script.

Dans le cas d'un environnement de tests, les installations suivantes devront être réalisées :

- Installation d'Apache,
- Installation de PHP,
- Installation de MariaDB,
- Installation de Composer,
- Installation de xDebug.

Environnement de production

Pour ce qui est de l'environnement de production, il y a plusieurs possibilités. Vous pouvez vous procurez un vps (Serveur dédié virtuel) auprès d'un hébergeur, type OVHcloud. Une seconde possibilité est d'héberger soi-même en ouvrant les ports de sa box. À titre d'exemple, lors de nos études, il a été mis à notre disposition des machines virtuelles servant de vps. Lesquelles permettront l'hébergement de nos portfolios. J'ai personnellement, pris un vps chez OVHcloud, ainsi qu'un nom de domaine que j'ai redirigé vers celui-ci. De ce fait le site que je suis actuellement entrain de développer, est disponible à l'adresse suivante :

- http://alexishenry.eu

Dans le cas d'un environnement de production, uniquement les installations suivantes devront être réalisées :

- Installation d'Apache,
- Installation de PHP,
- Installation de MariaDB.

Déploiement des instances

Installations

Les commandes cat sont à ignorer, elles servent juste à afficher toutes les commandes qui sont, elles, à exécuter. Toutes les commandes sont disponibles sur GitHub à ce lien.

Apache

Installation d'Apache,

```
ubuntu@dev:~$ sudo apt install apache2 -y
```

PHP

Installation de PHP, la version de PHP installé lors de ce projet est la version 8.1.

Sachant que ce n'est pas la version présente dans les dépôts d'Ubuntu 20.04, cela nécessite donc d'installer le package software-properties-common, comme ci-dessous :

```
ubuntu@dev:~$ cat php-install
sudo apt-get install software-properties-common -y;
sudo add-apt-repository ppa:ondrej/php -y;
sudo apt-get install php8.1 -y;
sudo apt install php8.1-common php8.1-mysql php8.1-xml php8.1-xmlrpc php8.1-curl php8.1-gd php8.1-imagick php8.1-cli php8.1-dev php8.1-imap php8.1-mbstring php8.1-opcache
 php8.1-soap php8.1-zip php8.1-redis php8.1-intl -y
```

MariaDB

Installation de MariaDB,

```
ubuntu@dev:~$ cat mariadb-install
sudo apt install mariadb-server -y
sudo apt install mariadb-client-core-10.3 -y
```

Samba

Installation de Samba,

Samba va servir à partager les fichiers de l'instance vers la machine de développement.

```
ubuntu@dev:~$ sudo apt install samba -y
```

Composer

Installation de Composer,

Composer un outil qui permet de gérer les dépendances en PHP.

Pour l'installer, il faut exécuter la commande suivante :

```
ubuntu@dev:~$ sudo curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/local/bin --filename=composer
```

XDebug

Installation de xDebug,

XDebug est une extension pour PHP contenant beaucoup de fonctionnalités.

```
ubuntu@dev:~$ sudo apt-get install php-xdebug -y
```

Avahi

Installation d'Avahi,

Le package avahi-deamon permet entre autres, à une machine de publier son nom de domaine sur le réseau, ce qui rend donc accessible la machine via une url de ce type :

- http://nomdelamachine.local

La commande servant à installer de package est la suivante :

```
ubuntu@dev:~$ sudo apt-get install avahi-daemon -y
```

PhpMyAdmin

Installation de PhpMyAdmin,

L'environnement comprenant la version 8.1 de PHP, l'installation de PhpMyAdmin n'est pas celle habituellement effectuée. Cela est dû au fait que les dépôts d'Ubuntu ne possèdent pas la dernière version de PhpMyAdmin. La version contenue n'est pas compatible avec la version 8.1 de PHP. De ce fait, nous devons téléchargement et mettre en place la dernière version de PhpMyAdmin à la main.

Les commandes ci-dessous, permettent d'installer la version 5.1.3 de PhpMyAdmin.

```
ubuntu@dev:~$ cat pma-install
sudo wget https://files.phpmyadmin.net/phpMyAdmin/5.1.3/phpMyAdmin-5.1.3-english.tar.gz
sudo tar -xf phpMyAdmin-5.1.3-english.tar.gz
rm -rf phpMyAdmin-5.1.3-english.tar.gz
sudo mkdir /usr/share/phpmyadmin
sudo mv phpMyAdmin-5.1.3-english/* /usr/share/phpmyadmin/
sudo mkdir -p /var/lib/phpmyadmin/tmp
sudo chown -R www-data:www-data /var/lib/phpmyadmin
sudo cp /usr/share/phpmyadmin/config.sample.inc.php /usr/share/phpmyadmin/config.inc.php
sudo rm -rf phpMyAdmin-5.1.3-english
```

Configurations

Accès à PhpMyAdmin

Pour que l'accès à PhpMyAdmin soit opérationnel, il faudra ajouter un fichier de configuration à Apache. Le fichier est disponible sur GitHub à [ce lien].

Il vous suffira de le télécharger et/ou de placer le contenu dans un fichier du même nom. Puis de déplacer ce fichier dans le dossier : « /etc/apache2/mods-available/ ».

La commande ci-dessous vous permettra de réaliser cela :

```
ubuntu@dev:~$ sudo mv path-to-file/phpmyadmin.conf /etc/apache2/mods-available//phpmyadmin.conf
```

Il faut bien entendu, remplacer 'path-to-file' par le chemin allant jusqu'au fichier créer.

Accès au site web via un nom de domaine

Utilisation d'Avahi

Si vous utilisez le package avahi-deamon, pour accéder à la machine via une page web, il vous suffira de vous rendre sur : http://nomDeLaMachine.local

Il vous faudra remplacer « nomDeLaMachine » par le nom de votre instance.

Via les fichiers de configuration de Windows

Si vous ne souhaitez pas utiliser Avahi, et vous contentez du fichier hosts de Windows. Il vous faudra alors réaliser les étapes ci-dessous :

- Récupérer l'adresse IP de votre machine :

```
ubuntu@dev:~$ echo $(/sbin/ip -o -4 addr list eth0 | awk '{print $4}' | cut -d/ -f1)
172.20.172.3
```

- Editer le fichier hosts de Windows :

```
PS C:\Users\Alexis> notepad 'C:\Windows\System32\drivers\etc\hosts'
```

- Ajouter l'IP de la machine associée au nom de domaine souhaitée :

```
172.20.172.3 nomdelamachine.com
```

Déploiement automatisé

Cloud-init

Le cloud-init fait partit des options que permet Multipass. C'est un fichier qui permet de préciser au lancement d'une machine virtuelle tout ce qu'on veut qu'elle effectue.

Le script qui est disponible sur GitHub à ce lien, génère une machine virtuelle avec tout ce qui est précisé dans cette documentation, déjà installé, ainsi que les droits déjà configurés.

Scripts

Pour optimiser le déploiement des instances ainsi que le temps prit à les lancer et à les configurer, j'ai opté pour un ensemble de scripts qui vont générer automatiquement une machine en fonction du type de machine que l'on désire (dev, tests, …).

J'ai créé un script en batch, qui à l'exécution, va simplement demander si c'est une machine de développement ou de test. En fonction de ça, il va attribuer un nom à la machine, puis la créer. Script qui est disponible ici :

- https://github.com/AlxisHenry/script-multipass-launch

Il va ensuite cloner à l'intérieur de cette machine un repository, contenant, tous les fichiers de configurations nécessaire au déploiement d'une machine. Ce repository contient un script en Bash, qui va à son tour, installer tout ce dont nous avons parlé dans cette documentation, et ce, en fonction du type de machine demandée. Repository disponible ici :

- https://github.com/AlxisHenry/linux-professional-environment

Le déploiement d'une machine ne prend donc plus que quelques minutes à réaliser.

Mise à jour du code entre les différents environnements

Vers l'environnement de tests/préproduction

Explication du fonctionnement

Le script qui va s'occuper de mettre à jour le code sur la machine de tests/préproduction, va fonctionner à l'aide de git. Il va réaliser cela à l'aide d'un git pull, une commande qui permet de récupérer le code mis à jour sur GitHub.

Script

Le script dans sa globalité est disponible ici :

- https://github.com/AlxisHenry/CCI-2021-PORTFOLIO/blob/main/tests/Scripts/up.sh

Vers l'environnement de production

Explication du fonctionnement

Contrairement à la machine de tests, le script d'envoi en production, ne fonctionnera pas à l'aide de git, pour la simple et bonne raison, que git n'est pas quelque chose à posséder en production. Nous privilégierons un système de transfert, comme rsync.

Script

Le script dans sa globalité est disponible ici :

- https://github.com/AlxisHenry/CCI-2021-PORTFOLIO/blob/main/tests/Scripts/prod.sh