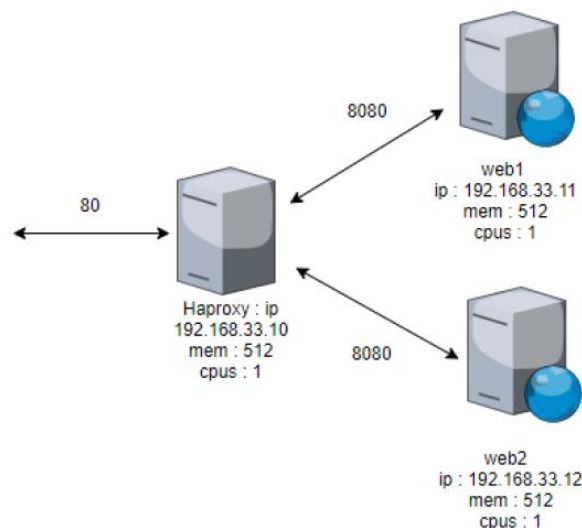


## Sommaire

1. Initiation au déploiement Vagrant .....	1
2. Initiation au provisionning avec Ansible.....	10
2.1) L'arborescence des fichiers ansible.....	12
2.2) Explication des playbooks .....	13
2.3) Explication des tâches effectuées dans chaque dossier roles .....	14
2.4) Explication du contenu des dossiers vars et defaults .....	16
2.5) Explication du dossier handlers.....	17
2.6) Explication des templates .....	17
2.7) Test du bon fonctionnement du script ansible .....	18

## 1. Initiation au déploiement Vagrant

Avant de commencer cette partie nous avons le schéma suivant qui va nous aider pour la suite :



1) Le script suivant permet de déployer 3 VMs haproxy, web1 et web2 avec la box centos/7 qui respecte les critères du schéma ci-dessus :

```
NODES = [
    { :hostname => "haproxy", :ip => "192.168.33.10", :cpus => 1, :mem => 512
    },
    { :hostname => "web1", :ip => "192.168.33.11", :cpus => 1, :mem => 512 },
    { :hostname => "web2", :ip => "192.168.33.12", :cpus => 1, :mem => 512 },
]
```

```

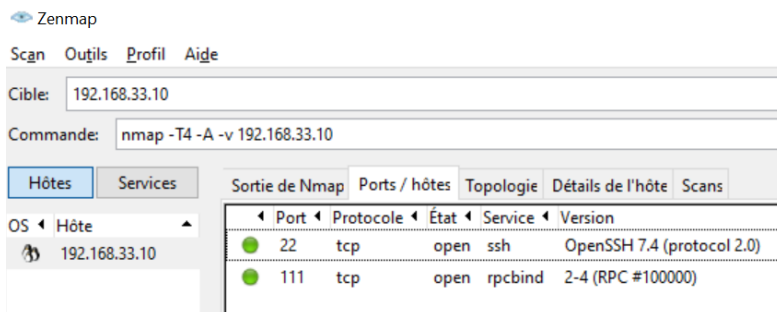
{:hostname => "web2", :ip => "192.168.33.12", :cpus => 1, :mem => 512}
]

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  NODES.each do |node|
    config.vm.define node[:hostname] do |cfg|
      cfg.vm.hostname = node[:hostname]
      cfg.vm.network "private_network", ip: node[:ip]
      cfg.vm.provider "virtualbox" do |v|
        v.customize ["modifyvm", :id, "--cpus", node[:cpus] ]
        v.customize ["modifyvm", :id, "--memory", node[:mem] ]
        v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
        v.customize ["modifyvm", :id, "--natdnsproxy1", "on"]
        v.customize ["modifyvm", :id, "--name", node[:hostname] ]
      end
    end
  end
end
end
end

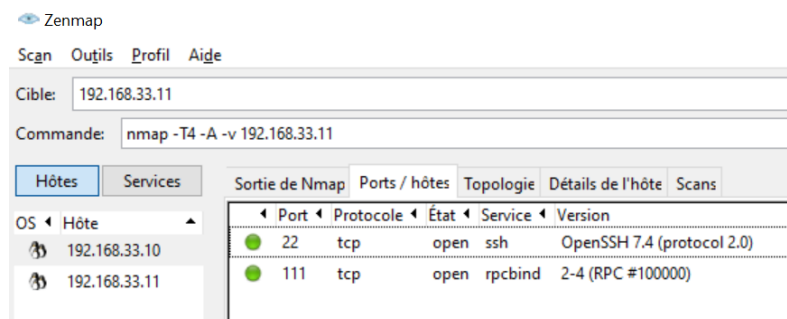
```

2) Maintenant je dois installer la commande nmap sur ma machine physique Windows et je scan les ports de mes différentes VMs :

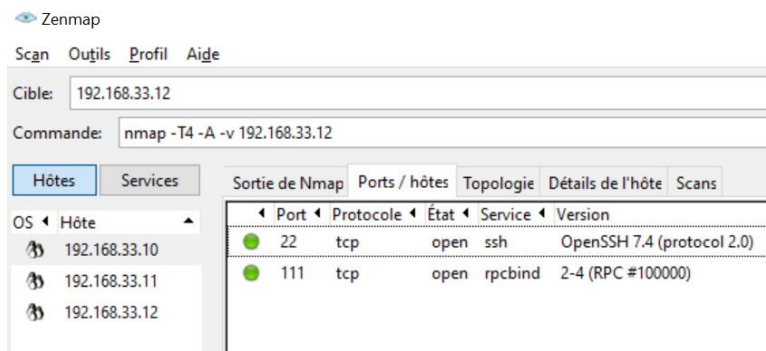
Machine haproxy :



Machine web1 :



Machine web2 :



Les 3 machines ont les mêmes ports ouverts par défaut lors de la création de celles-ci

Je dois maintenant protéger mes 3 VMs en installant firewalld dessus. Ce paquet étant déjà installé par défaut je n'ai besoin que d'activer le service et appliquer les règles de protection :

Sur les machines web1 et 2 :

```
sudo systemctl start firewalld
sudo firewall-cmd --permanent --zone=internal --add-source=192.168.33.10
sudo firewall-cmd --permanent --zone=internal --add-port=8080/tcp
sudo firewall-cmd --permanent --zone=public --add-port=22/tcp
sudo firewall-cmd --reload
```

Sur la machine haproxy :

```
sudo systemctl start firewalld
sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
sudo firewall-cmd --permanent --zone=public --add-port=22/tcp
sudo firewall-cmd --reload
```

Je ne laisse que les ports 80 et 22 ouverts sur ma VM haproxy

Sur les VMs web je ne laisse que la machine haproxy accéder à la machine via le port 8080 ainsi que le port 22 ouvert pour toutes les IPs dans le même réseau

3) Je peux maintenant configurer le SSH sur mes 3 VMs afin qu'elles soient accessibles via la commande ssh [nom\_VM].

Tout d'abord je dois me générer une paire de clé SSH :

```
ssh-keygen -t rsa -b 4096
```

Maintenant je peux mettre la paire de clé dans un répertoire spécifique au TP :

```
C:/Users/samy2/OneDrive/Bureau/MAJEURE VIRTUALISATION/SSH haproxy/public.pub
C:/Users/samy2/OneDrive/Bureau/MAJEURE VIRTUALISATION/SSH haproxy/id_rsa
```

Pour finir il faut mettre le code suivant dans le VagrantFile :

```
config.vm.define "haproxy" do |haproxy|
  haproxy.vm.provision "shell" do |s|
    #      ssh_pub_key = File.readlines("C:/Users/samy2/OneDrive/Bureau/MAJEURE
    VIRTUALISATION/SSH haproxy/public.pub").first.strip
    #      s.inline = <<-SHELL
    #      echo #{ssh_pub_key} >> /home/vagrant/.ssh/authorized_keys
    #      SHELL
    #      end
    # end
```

Ce code permet d'aller lire le contenu de la clé publique et de l'écrire dans le répertoire des clés autorisées lors de la création de la machine virtuelle

Maintenant je peux me connecter à la machine en mettant la commande suivante :

```
ssh -i "C:\Users\samy2\OneDrive\Bureau\MAJEURE VIRTUALISATION\SSH
haproxy\id_rsa" USERt@IP_Machine
```

Ici le -i sert à spécifier le chemin où se situe la clé privée

Si je veux configurer le SSH sur mes VMs pour qu'elles puissent se connecter entre elles via leur nom il faut faire les étapes suivantes :

Les VMs doivent d'abord savoir quelle IP porte quel nom. Pour cela j'édite le fichier /etc/hosts et dedans je défini les IPs et les noms auxquelles elles sont liées :

```
192.168.33.10 haproxy
192.168.33.11 web1
192.168.33.12 web2
```

Je dois aussi éditer le fichier /etc/ssh/sshd\_config pour autoriser chaque VMs à copier sa clé publique sur chaque serveur :

```
ChallengeResponseAuthentication yes
```

Puis redémarrer le service sshd :

```
sudo service sshd reload
```

Maintenant sur chaque serveur je peux faire les commandes suivantes pour générer une paire de clé SSH et copier la clé publique vers une machine distante :

```
ssh-keygen -t rsa -b 4096
sudo ssh-copy-id -i ~/.ssh/id_rsa.pub [Nom_Machine]
```

Je peux maintenant accéder à chaque VM grâce à la commande suivante :

```
ssh [Nom_Machine]
```

4) J'installe maintenant le paquet haproxy sur ma VM haproxy et le paquet nginx sur mes VMs web1 et web2 :

Haproxy :

```
sudo yum update -y
sudo yum install haproxy -y
```

Sur les serveurs web1 et web2 :

```
sudo yum update -y
sudo yum install epel-release -y
sudo yum install nginx -y
sudo systemctl start nginx
```

Nous avons besoin de récupérer le repository epel qui contient le paquet nginx pour centos

5) Pour que les serveurs web répondent sur le port 8080 il faut modifier le fichier **/etc/nginx/nginx.conf** :

```
server {
    listen      8080;
    listen      [::]:8080;
    server_name _;
    root        /usr/share/nginx/html;
```

Ensuite pour vérifier que le load-balancing fonctionne sur le serveur haproxy bien on peut modifier le fichier **/usr/share/nginx/html/index.html** et faire 2 pages web différentes sur les serveurs web puis relancer le serveur nginx pour prendre en compte les modifications

6) Pour que le haproxy puisse servir à tour de rôle l'un des serveurs web il faut modifier le fichier **/etc/haproxy/haproxy.cfg** :

```
global
    log /dev/log local0
    log /dev/log local1 notice

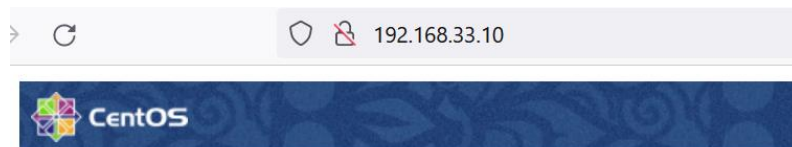
defaults
    log global
    mode http
    option httplog
    option dontlognull
    timeout connect 5000
```

```
timeout client 50000
timeout server 50000

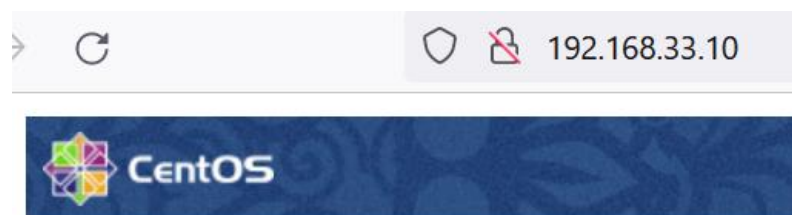
frontend http_front
  bind *:80
  stats uri /haproxy?stats
  default_backend http_back

backend http_back
  balance roundrobin
  server web1 192.168.33.11:8080 check
  server web2 192.168.33.12:8080 check
```

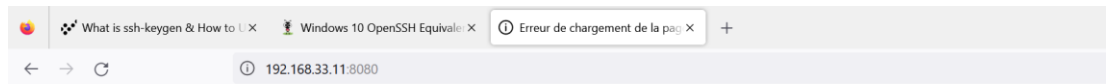
Je peux maintenant tester si le haproxy fait bien le load-balancing :



Maintenant si j'actualise la page je devrais arriver sur la page du serveur web2 :



Le load-balancing se fait bien grâce au serveur haproxy. Je peux aussi tester d'accéder à la page web d'un des serveurs via son IP directement. Je ne devrais pas pouvoir y arriver car les règles du firewall n'autorisent que le haproxy à se connecter :



Le délai d'attente est dépassé

Le serveur à l'adresse 192.168.33.11 met trop de temps à répondre.

- Le site est peut-être temporairement indisponible ou surchargé. Réessayez plus tard ;
- Si vous n'arrivez à naviguer sur aucun site, vérifiez la connexion au réseau de votre ordinateur ;
- Si votre ordinateur ou votre réseau est protégé par un pare-feu ou un proxy, assurez-vous que Firefox est autorisé à accéder au Web.

Réessayer

7) Pour terminer je peux mettre en place un script qui va déployer automatiquement toute l'architecture que nous venons de voir. Le script est donc le suivant :

```
NODES = [
  {:hostname => "haproxy", :ip => "192.168.33.10", :cpus => 1, :mem => 512
},
  {:hostname => "web1", :ip => "192.168.33.11", :cpus => 1, :mem => 512},
  {:hostname => "web2", :ip => "192.168.33.12", :cpus => 1, :mem => 512}
]

Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  NODES.each do |node|
    config.vm.define node[:hostname] do |cfg|
      cfg.vm.hostname = node[:hostname]
      cfg.vm.network "private_network", ip: node[:ip]
      cfg.vm.provider "virtualbox" do |v|
        v.customize ["modifyvm", :id, "--cpus", node[:cpus] ]
        v.customize ["modifyvm", :id, "--memory", node[:mem] ]
        v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
        v.customize ["modifyvm", :id, "--natdnsproxy1", "on"]
        v.customize ["modifyvm", :id, "--name", node[:hostname] ]
      end
    end
  end
end

## CONFIG MACHINE HAPROXY ##
config.vm.define "haproxy" do |haproxy|
  haproxy.vm.provision "shell", inline: <<-SHELL
    sudo yum update -y
    sudo yum install haproxy sshpass -y
```

```
SHELL

haproxy.vm.provision "file", source: "haproxy.cfg", destination:
"/tmp/haproxy.cfg"

haproxy.vm.provision "shell", inline: <<-SHELL
    sudo rm /etc/haproxy/haproxy.cfg
    sudo mv /tmp/haproxy.cfg /etc/haproxy/haproxy.cfg
    sudo systemctl start haproxy

    sudo bash -c 'echo "192.168.33.11 web1" >> /etc/hosts'
    sudo bash -c 'echo "192.168.33.12 web2" >> /etc/hosts'

    sudo sed -i 's/ChallengeResponseAuthentication
no/ChallengeResponseAuthentication yes/g' /etc/ssh/sshd_config
    sudo systemctl restart sshd

    ssh-keygen -t rsa -b 4096 -f /home/vagrant/.ssh/haproxy -q -N ""

    sudo systemctl start firewalld
    sudo firewall-cmd --permanent --zone=public --add-port=80/tcp
    sudo firewall-cmd --permanent --zone=public --add-port=22/tcp
    sudo firewall-cmd --reload
SHELL
end

## CONFIG MACHINE WEB 1 ##
config.vm.define "web1" do |web|
    web.vm.provision "shell", inline: <<-SHELL
        sudo yum update -y
        sudo yum install epel-release -y
        sudo yum install nginx sshpass -y
        sudo sed -i "s/80/8080/g" /etc/nginx/nginx.conf
        sudo bash -c 'echo "<html><h1>Web 1</h1></html>" >
/usr/share/nginx/html/index.html'
        sudo systemctl start nginx

        sudo bash -c 'echo "192.168.33.10 haproxy" >> /etc/hosts'
        sudo bash -c 'echo "192.168.33.12 web2" >> /etc/hosts'

        sudo sed -i 's/ChallengeResponseAuthentication
no/ChallengeResponseAuthentication yes/g' /etc/ssh/sshd_config
        sudo systemctl restart sshd

        ssh-keygen -t rsa -b 4096 -f /home/vagrant/.ssh/web1 -q -N ""

        sudo systemctl start firewalld
        sudo firewall-cmd --permanent --zone=internal --add-
source=192.168.33.10
```



```
sudo firewall-cmd --permanent --zone=internal --add-port=8080/tcp
sudo firewall-cmd --permanent --zone=public --add-port=22/tcp
sudo firewall-cmd --reload
SHELL
end

## CONFIG MACHINE WEB 2 ##
config.vm.define "web2" do |web|
  web.vm.provision "shell", inline: <<-SHELL
    sudo yum update -y
    sudo yum install epel-release -y
    sudo yum install nginx sshpass -y
    sudo sed -i "s/80/8080/g" /etc/nginx/nginx.conf
    sudo bash -c 'echo "<html><h1>Web 2</h1></html>" >
/usr/share/nginx/html/index.html'
    sudo systemctl start nginx

    sudo bash -c 'echo "192.168.33.10 haproxy" >> /etc/hosts'
    sudo bash -c 'echo "192.168.33.11 web1" >> /etc/hosts'

    sudo sed -i 's/ChallengeResponseAuthentication
no/ChallengeResponseAuthentication yes/g' /etc/ssh/sshd_config
    sudo systemctl restart sshd

    ssh-keygen -t rsa -b 4096 -f /home/vagrant/.ssh/web2 -q -N ""

## CLE WEB2 vers WEB1 & HA ##
    sudo sshpass -p "vagrant" ssh-copy-id -i
/home/vagrant/.ssh/web2.pub -o StrictHostKeyChecking=no vagrant@web1
    sudo sshpass -p "vagrant" ssh-copy-id -i
/home/vagrant/.ssh/web2.pub -o StrictHostKeyChecking=no vagrant@haproxy

## CLE WEB1 vers WEB2 & HA ##
    sudo ssh -i /home/vagrant/.ssh/web2 -o StrictHostKeyChecking=no
vagrant@web1 "sudo sshpass -p 'vagrant' ssh-copy-id -i
/home/vagrant/.ssh/web1.pub -o StrictHostKeyChecking=no vagrant@web2 \
    && sudo sshpass -p 'vagrant' ssh-copy-id -i
/home/vagrant/.ssh/web1.pub -o StrictHostKeyChecking=no vagrant@haproxy"

## CLE HA vers WEB1 & WEB2 ##
    sudo ssh -i /home/vagrant/.ssh/web2 -o StrictHostKeyChecking=no
vagrant@haproxy "sudo sshpass -p 'vagrant' ssh-copy-id -i
/home/vagrant/.ssh/haproxy.pub -o StrictHostKeyChecking=no vagrant@web1 \
    && sudo sshpass -p 'vagrant' ssh-copy-id -i
/home/vagrant/.ssh/haproxy.pub -o StrictHostKeyChecking=no vagrant@web2"
```

```

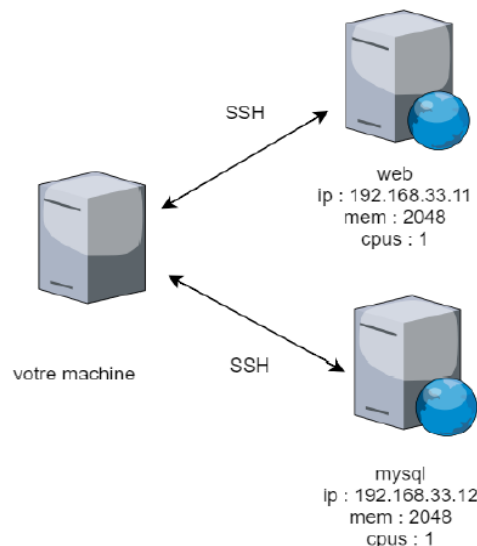
        sudo systemctl start firewalld
        sudo firewall-cmd --permanent --zone=internal --add-
source=192.168.33.10
        sudo firewall-cmd --permanent --zone=internal --add-port=8080/tcp
        sudo firewall-cmd --permanent --zone=public --add-port=22/tcp
        sudo firewall-cmd --reload
    SHELL
end
end

```

Il nous faut créer un fichier haproxy.cfg dans le dossier où se situe le VagrantFile. Ce fichier contient les modifications du haproxy. Le VagrantFile va récupérer ce fichier et l'envoyer vers la VM haproxy afin de remplacer l'ancien fichier haproxy.cfg

## 2. Initiation au provisionning avec Ansible

Durant cette partie je vais devoir reproduire l'architecture suivante :



1) A l'aide de Vagrant je reproduis l'architecture ci-dessus :

```

NODES = [
  {:hostname => "web", :ip => "192.168.56.2", :cpus => 1, :mem => 2048 },
  {:hostname => "mysql", :ip => "192.168.56.3", :cpus => 1, :mem => 2048}
]

Vagrant.configure("2") do |config|
  config.vm.box = "generic/debian11"
  NODES.each do |node|
    config.vm.define node[:hostname] do |cfg|
      cfg.vm.hostname = node[:hostname]
      cfg.vm.network "private_network", ip: node[:ip]
      cfg.vm.provider "virtualbox" do |v|
        v.customize ["modifyvm", :id, "--cpus", node[:cpus] ]
        v.customize ["modifyvm", :id, "--memory", node[:mem] ]
      end
    end
  end
end

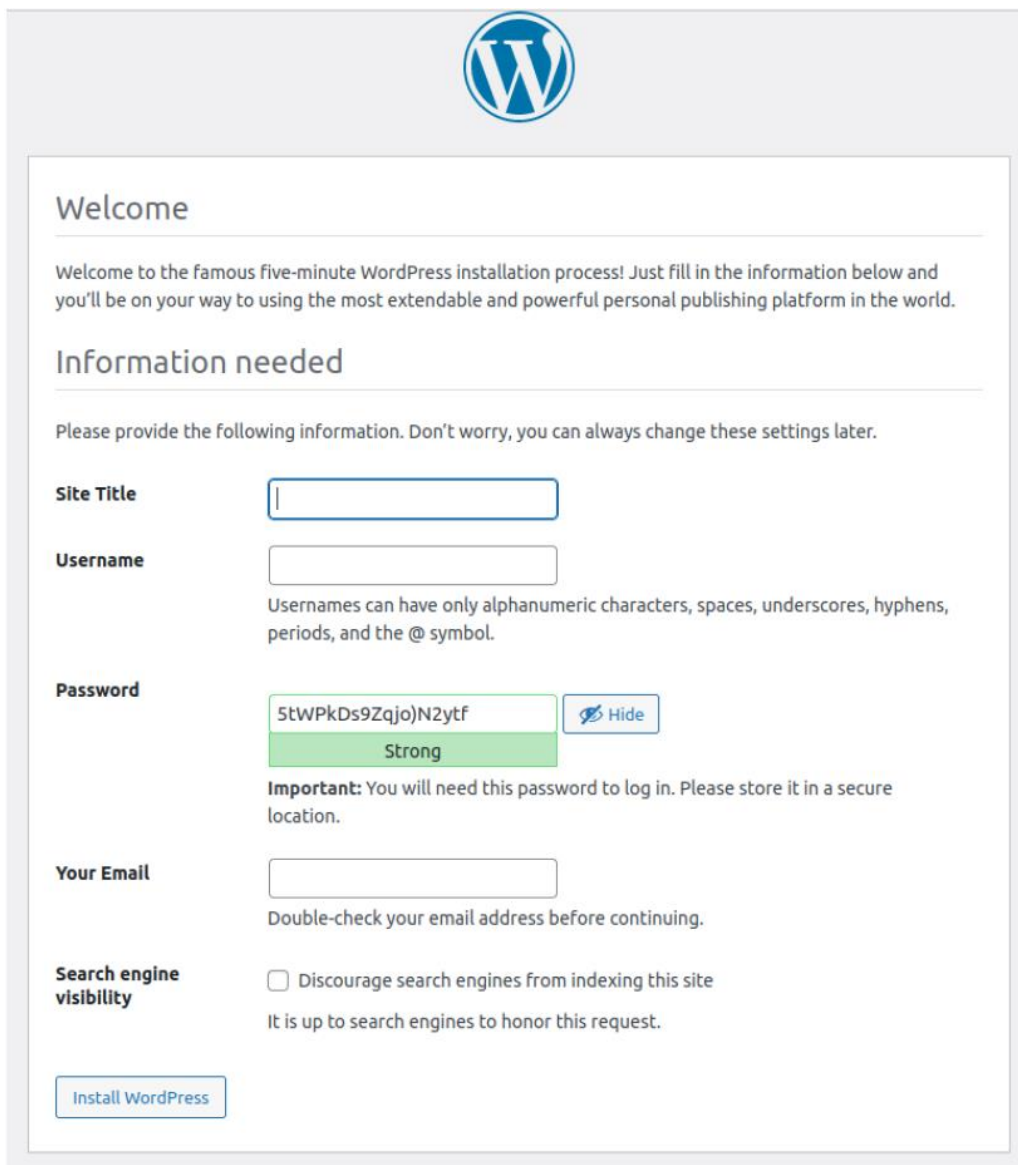
```

```
v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
v.customize ["modifyvm", :id, "--natdnsproxy1", "on"]
v.customize ["modifyvm", :id, "--name", node[:hostname] ]

end
end
end
```

Les IPs ne sont pas les bonnes car Virtualbox n'accepte pas la plage IP 192.168.33.X/24 sous Linux. Il faudrait pour cela que je crée une nouvelle carte réseau en tant que root sur Vbox pour pouvoir ensuite modifier les IPs dans le fichier Vagrantfile

2) A l'aide de l'outil ansible je dois provisionner mes machines afin d'y installer un serveur web, une base de données et un Wordpress. Le résultat de mon script devra m'emmener sur la page suivante lors de ma connexion à ma VM web dans un navigateur (firefox, google chrome...) :



The image shows the WordPress installation welcome screen. At the top is the WordPress logo. Below it is a 'Welcome' section with a message about the five-minute installation process. The main section is 'Information needed', which asks for site details. It includes input fields for 'Site Title', 'Username', and 'Your Email'. The 'Password' field is filled with '5tWPkDs9Zqjo)N2ytf' and is marked as 'Strong'. There is a 'Hide' button next to the password field. A note states that the password is important for login. At the bottom, there is a checkbox for 'Search engine visibility' which is unchecked, with a note that it is up to search engines to honor the request. An 'Install WordPress' button is at the bottom left.

**Welcome**

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

**Information needed**

Please provide the following information. Don't worry, you can always change these settings later.

**Site Title**

**Username**

Username can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

**Password**  [Hide](#)

**Strong**

**Important:** You will need this password to log in. Please store it in a secure location.

**Your Email**

Double-check your email address before continuing.

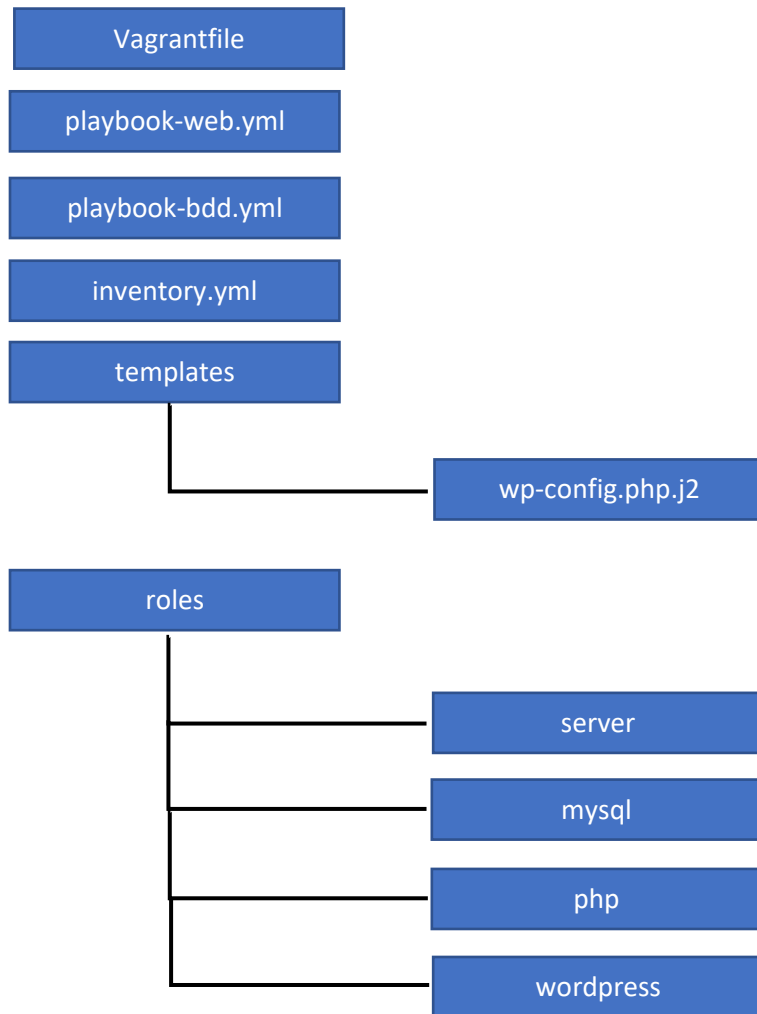
**Search engine visibility** ☐ Discourage search engines from indexing this site

It is up to search engines to honor this request.

[Install WordPress](#)

## 2.1) L'arborescence des fichiers ansible

Voici donc l'arborescence de mes scripts :



Chaque rôle contient différents dossiers avec les tâches qui sont réalisées ainsi que les variables qui sont utilisées dans les rôles

Je peux maintenant modifier mon Vagrantfile pour qu'il exécute chaque playbook sur la machine concernée :

```
config.vm.define "web" do |web|

  web.vm.provision "shell", inline: <<-SHELL
  sudo apt update -y
  sudo apt install ansible -y
  SHELL

  web.vm.provision "ansible" do |ansible|
    ansible.playbook = "playbook-web.yml"
  end
end
```

```
config.vm.define "mysql" do |mysql|

  mysql.vm.provision "shell", inline: <<-SHELL
  sudo apt update -y
  sudo apt install ansible -y
  SHELL

  mysql.vm.provision "ansible" do |ansible|
    ansible.playbook = "playbook-bdd.yml"
  end
end
```

Chaque playbook se base sur un fichier inventaire qui va contenir les IP/FQDN des machines sur lesquelles le playbook doit se lancer. Voici le contenu de mon inventaire :

```
localhost ansible_connection=local
```

Cette ligne veut dire que le playbook va devoir s'exécuter sur la machine locale et non pas sur une machine distante

## 2.2) Explication des playbooks

Voici maintenant le contenu de chaque playbook :

### Playbook-web.yml :

```
- hosts: all

roles:
- server
- php
- wordpress
```

Ce playbook s'exécute sur chaque machine définie dans la partie « all » (ici machine locale) du fichier inventaire et va exécuter les différentes tâches contenues dans les rôles qui sont appelés

### Playbook-bdd.yml :

```
- hosts: all

roles:
- server
- mysql
```

## 2.3) Explication des tâches effectuées dans chaque dossier roles

### Roles/server/tasks/mail.yml :

```
---
- name: Installation du package permettant d'installer le paquet mysql-server
  ansible.builtin.apt:
    deb: https://dev.mysql.com/get/mysql-apt-config_0.8.22-1_all.deb
    become: true

- name: Mise à jour du serveur (apt update)
  ansible.builtin.apt:
    update_cache: yes
    become: true

- name: Installation des paquets nécessaires à Wordpress
  ansible.builtin.apt:
    pkg:
      - apache2
      - mysql-server
      - php-mysql
      - php
      - libapache2-mod-php
      - pip
    become: true

- name: Installation d'un paquet python (pymysql)
  ansible.builtin.pip:
    name: pymysql
    become: true
```

Ce rôle est dédié aux 2 machines et est composé de 4 tâches. Ces 4 tâches vont permettre de télécharger et d'installer tous les paquets nécessaires pour avoir un LAMP fonctionnel. Ici pour avoir le paquet mysql-server disponible au téléchargement il a d'abord fallu que je télécharge et que j'installe le paquet .deb qui va contenir le paquet mysql

### Roles/mysql/tasks/mail.yml :

```
---
# tasks file for mysql

- name: Création de la base de données pour Wordpress
  mysql_db:
    name: "{{ wp_mysql_db }}"
    state: present
    login_unix_socket: /run/mysqld/mysqld.sock
    become: true
```

```
- name: Création utilisateur distant
mysql_user:
  name: "{{ wp_mysql_user }}"
  password: "{{ wp_mysql_password }}"
  priv: ' *.*:ALL '
  host: "{{ wp_mysql_host }}"
  login_unix_socket: /run/mysqld/mysqld.sock
  become: true
```

Ce rôle est dédié à la base de données et est composé de 2 tâches. La 1<sup>ère</sup> va permettre de créer une base de données mysql qui portera le nom stocké dans mon fichier de variables. La 2<sup>ème</sup> va permettre de créer un utilisateur distant qui pourra accéder à la base de donnée depuis une autre machine (ici la machine web)

#### Roles/php/tasks/mail.yml :

```
---
# tasks file for php
- name: Installation des modules php
  apt:
    name:{{ item }}
    state:present
  become: true
  with_items:
    - php-gd
    - php-ssh2
```

Ce rôle est dédié au serveur web et est composé d'une seule tâche. Cette tâche permet d'installer des extensions à php afin de faire tourner correctement le wordpress

#### Roles/wordpress/tasks/mail.yml :

```
---
- name: Téléchargement de Wordpress
  get_url:
    url: https://wordpress.org/latest.tar.gz
    dest: /tmp/wordpress.tar.gz
    validate_certs: no
  become: true

- name: Extraction du paquet Wordpress
  unarchive:
    src: /tmp/wordpress.tar.gz
    dest: /var/www/
    copy: no
  become: true
```

```
- name: Mise à jour du site par défaut de Apache
  become: true
  lineinfile:
    dest: /etc/apache2/sites-enabled/000-default.conf
    regexp: "(.)+DocumentRoot /var/www/html"
    line: "DocumentRoot /var/www/wordpress"
  notify:
    - restart apache

- name: Configuration du fichier wp-config.php
  template:
    src: templates/wp-config.php.j2
    dest: /var/www/wordpress/wp-config.php
  become: true
```

Ce rôle est dédié au serveur web et est composé de 4 tâches. La 1<sup>ère</sup> et la 2<sup>ème</sup> tâche vont permettre de télécharger et installer les paquets pour avoir Wordpress. La 3<sup>ème</sup> tâche va modifier le document root du serveur apache pour ne pas lancer la page contenue dans /var/www/html mais dans /var/www/wordpress et va relancer le service pour prendre en compte les modifications. Pour finir la dernière tâche va créer le fichier wp-config.php qui va contenir les informations de ma base de données afin d'y avoir accès directement au lancement du Wordpress.

## 2.4) Explication du contenu des dossiers vars et defaults

Les dossiers vars et defaults vont contenir les différentes variables qui seront appelées dans les tasks vu précédemment. Dans les tasks les variables sont appelées entre {{ }}. Lorsqu'une variable est appelée le script va chercher le contenu de cette variable dans le dossier defaults ou vars.

Voici donc le contenu de mes dossiers defaults :

### Roles/mysql/defaults/main.yml :

```
---
# defaults file for mysql
wp_mysql_db: wordpressdb
wp_mysql_user: wordpress
wp_mysql_password: wordpress
wp_mysql_host: 192.168.56.2
```

### Roles/wordpress/defaults/main.yml :

```
---
# defaults file for wordpress
wp_mysql_db: wordpressdb
wp_mysql_user: wordpress
wp_mysql_password: wordpress
```



```
wp_mysql_host: 192.168.56.3
```

## 2.5) Explication du dossier handlers

Le dossier handlers va contenir une ou plusieurs tâches permettant de redémarrer un service par exemple. Ici pour l'installation de Wordpress j'ai besoin d'apporter des modifications sur le serveur apache et donc de redémarrer le service une fois le serveur configuré. Pour cela j'utilise un handler.

Son contenu est le suivant : roles/wordpress/handlers/main.yml :

```
---
# handlers file for wordpress
- name: restart apache
  service:
    name: apache2
    state: restarted
  become: yes
```

## 2.6) Explication des templates

Les templates permettent de créer des documents qui contiennent des informations de configuration par exemple. Ici j'ai utilisé un template pour me créer un fichier wp-config.php permettant de mettre à jour la liaison de mon Wordpress à ma base de données.

Voici donc son contenu : templates/wp-config.php.j2 :

```
<?php

/** The name of the database for WordPress */
define( 'DB_NAME', '{{ wp_mysql_db }}' );

/** Database username */
define( 'DB_USER', '{{ wp_mysql_user }}' );

/** Database password */
define( 'DB_PASSWORD', '{{ wp_mysql_password }}' );

/** Database hostname */
define( 'DB_HOST', '{{ wp_mysql_host }}' );

/** Database charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );
```

```
/** The database collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

define( 'AUTH_KEY',          'put your unique phrase here' );
define( 'SECURE_AUTH_KEY',   'put your unique phrase here' );
define( 'LOGGED_IN_KEY',     'put your unique phrase here' );
define( 'NONCE_KEY',         'put your unique phrase here' );
define( 'AUTH_SALT',         'put your unique phrase here' );
define( 'SECURE_AUTH_SALT',  'put your unique phrase here' );
define( 'LOGGED_IN_SALT',    'put your unique phrase here' );
define( 'NONCE_SALT',        'put your unique phrase here' );

/**#@-*/

$table_prefix = 'wp_';

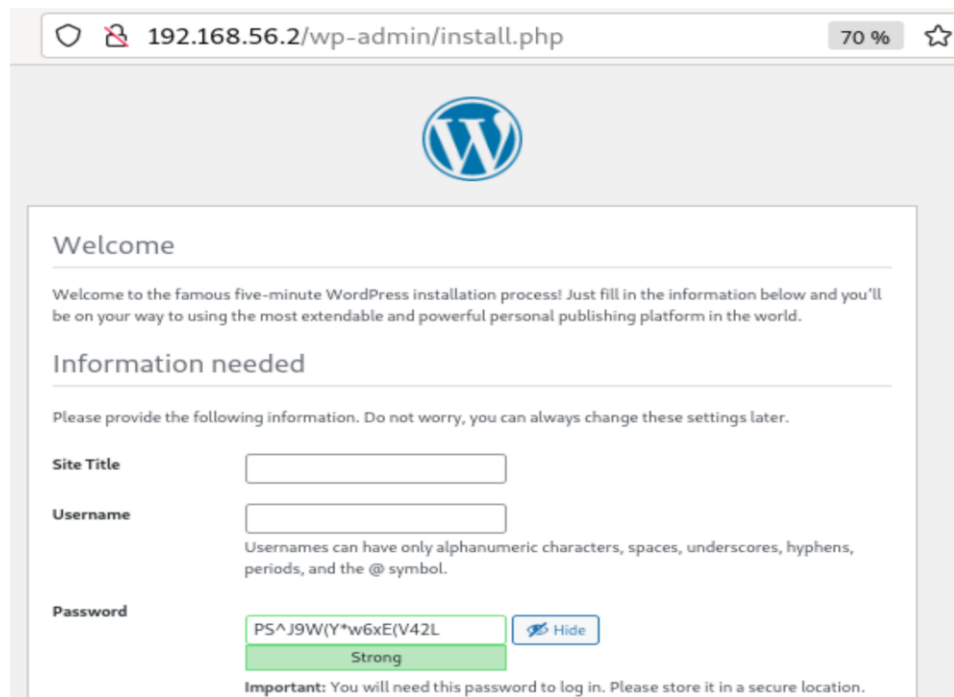
define( 'WP_DEBUG', false );

/** Absolute path to the WordPress directory. */
if ( ! defined( 'ABSPATH' ) ) {
    define( 'ABSPATH', __DIR__ . '/' );
}


/** Sets up WordPress vars and included files. */
require_once ABSPATH . 'wp-settings.php';
```

## 2.7) Test du bon fonctionnement du script ansible

Pour terminer je peux vérifier le bon fonctionnement de mon script en exécutant la commande « vagrant up ». Une fois les machines créées et configurées je peux lancer un navigateur et vérifier que je tombe bien sur la page de configuration de mon Wordpress :



192.168.56.2/wp-admin/install.php 70 %



## Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

### Information needed

Please provide the following information. Do not worry, you can always change these settings later.

**Site Title**

**Username**

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

**Password**  [Hide](#)

**Strong**

**Important:** You will need this password to log in. Please store it in a secure location.