Automatiser l'instanciation et le provisionnement de machines virtuelles avec Virtualbox et Vagrant

Olivier Martin

Janvier 2024

Premier TP de virtualisation pour le module R4.08, BUT Informatique deuxième année.

Travail individuel en salle de machines.

Compétence travaillée : Installer, configurer, mettre à disposition, maintenir en conditions opérationnelles des infrastructures, des services et des réseaux et optimiser le système informatique d'une organisation
Niveau 2 : Déployer des services dans une architecture réseau.

Pré-requis :
- Notions d'utilisation du shell Bash
- Installation de logiciels sous GNU/Linux et Windows

Matériel ou logiciels utilisés :
- Ordinateur de type desktop, disposant d'un système d'exploitation Debian 11 et Windows 10
- Les étudiants doivent avoir les droits d'administration sur cet ordinateur

Table des matières

1 Motivation

1.1 Automatisation

Vous avez vu en cours qu'un des intérêts de la virtualisation était l'automatisation de toutes les tâches de mise en place des infrastructures informatiques.

Dans le premier TD vous avez déployé manuellement un service sur un ensemble de machines virtuelles. Dans ce TP nous aborderons la question de l'automatisation. Pour cela nous utiliserons l'hyperviseur Virtual Box que vous connaissez déjà depuis la première année de BUT.

Lorsque vous avez utilisé Virtual Box, vous êtes passé par son interface graphique, ou GUI. On peut aussi utiliser Virtual Box en ligne de commande (CLI), notamment avec le programme VBoxManage. La CLI permet de faire tout ce que permet la GUI, à condition de connaître toute les options et le fonctionnement interne du logiciel.

Si l'utilisation de la CLI peut sembler plus complexe que celle de la GUI, elle offre l'avantage de pouvoir être appelée dans des scripts shell, et donc d'automatiser les traitement là où la GUI nécessite la présence d'une personne.

Dans ce TP vous n'utiliserez pas directement la CLI de Virtual Box car un utilitaire va vous permettre de créer des VM sans connaître une seule option de VBoxManage, avec des fichiers de configuration que vous pourrez ensuite réutiliser avec d'autres hyperviseurs.

1.2 Vagrant

Vagrant est cet utilitaire, qui permet de créer des VM à partir de fichiers de configuration en exécutant une simple commande shell.

On pourrait sans doute faire la même chose avec des scripts shell, mais Vagrant est compatible avec plusieurs hyperviseur et permet de passer de l'un à l'autre en changeant quelques lignes du fichier de configuration, ce qui ne serait pas le cas avec un script shell.

Vagrant offre aussi plusieurs façons de provisionner[1] les machines qu'il crée. Une première possibilité est l'appel de commandes ou de scripts sur les machines mais, à moins de faire des scripts complexes et peu maintenables[2], elle est assez limitée. Une autre possibilité est l'utilisation de Ansible.

1.3 Ansible

Ansible est un outil de provisionnement. Il permet de configurer des systèmes à distance, et d'y installer des programmes et des données. Tout ceci se fait sans intervention humaine.

En combinant Vagrant et Ansible on peut donc instancier de manière automatique des VM et les paramétrer pour une utilisation particulière. La description du résultat attendu se fait par des fichiers de configuration, c'est ce qu'on appelle "Infrastructure as Code" (IaC) et c'est grâce à ces technologies que la virtualisation a permis de réduire les coûts des serveurs informatiques.

1.4 But du TP

Dans ce TP vous allez créer un fichier de configuration pour Vagrant pour créer deux VM connectées par un réseau privé. Sur ces VM nous installerons un service par l'intermédiaire de Ansible. Le service choisi est Wordpress, que vous connaissez bien maintenant puisque le TD 1 de ce module sur la virtualisation porte aussi sur lui, vous serez donc en terrain connu.

2 Travail sous système Debian

2.1 Installation de Vagrant

Vérifiez si Vagrant est installé avec la commande
`vagrant --version`

Installez-le si nécessaire avec les commandes habituelles (`apt update` et `apt install`.)

2.2 Instanciation d'une VM simple

Créez un répertoire, nommé `vagrant_ex1`, ouvrez un shell dans ce répertoire et exécutez la commande

1. provisionner : "Affecter des ressources à un utilisateur ou à un système, en effectuer la configuration depuis un guichet central" (Wiktionary)
2. Essayez de créer un script qui modifie un fichier .ini si vous en voulez la preuve.

`vagrant init ubuntu/bionic64`

Cette commande crée dans le répertoire courant un fichier nommé `Vagrantfile`. Vous pouvez ouvrir ce fichier dans un éditeur, nous en étudierons le contenu plus tard.

Exécutez la commande
`vagrant up`

Vagrant télécharge les données nécessaires pour créer une VM avec le système Ubuntu 18.04 (nom de code "Bionic Beaver") en version 64 bits, puis il instancie la VM.

Ces données pour instancier la VM se nomment "box", elles sont stockées dans un répertoire propre à Vagrant. Vous pouvez utiliser la commande `vagrant box` pour gérer les boxes présentes sur votre poste de travail. Tapez `vagrant box --help` pour en savoir plus ou consultez la documentation de Vagrant sur `https://developer.hashicorp.com/vagrant/docs/cli/box`.

Dans ce TP vous laisserez Vagrant télécharger et mettre à jour les boxes. Vous n'aurez besoin que de `vagrant box remove` pour faire du ménage en fin de TP.

L'éditeur de Vagrant propose une large gamme de boxes prêtes à l'emploi et modifiables, la liste se trouve sur `https://app.vagrantup.com/boxes/search`.

Vous pouvez maintenant vous connecter à la machine avec la commande
`vagrant ssh`

Confirmez grâce à l'invite de commande que le shell n'est plus sur la machine hôte, testez la connexion réseau en mettant à jour les packages avec `apt update` (sans `sudo` car vous êtes root sur cette machine) puis déconnectez-vous avec `exit`.

On peut se connecter à la machine sans passer par Vagrant, avec un client SSH quelconque, par exemple la commande Unix `ssh`. Pour cela il faut un fichier de clé privée, et l'adresse et le port de la machine sur le réseau :

- La clé privée se trouve dans un sous répertoire de celui qui contient le vagrant file, à l'adresse relative `.vagrant/machines/default/virtualbox`
- L'adresse de connexion est 127.0.0.1, c'est à dire l'adresse locale de votre poste de travail. En effet, la VM n'est pas directement accessible par le réseau, il faut passer par un mécanisme de redirection de port : les données qui arrivent sur un port de votre poste de travail sont redirigées vers le port SSH de la VM.
- Pour trouver vers quel port de la machine hôte le port SSH de la VM a été redirigé il suffit de regarder le infos donnés lors du démarrage de la VM. Dans l'exemple de la figure 1 il s'agit du port 2222

Adaptez la commande suivante à votre configuration pour vous connecter à la VM sans passer par Vagrant :

```
ssh -i ./.vagrant/machines/default/virtualbox/private_key \
  vagrant@127.0.0.1 -p 2222
```

```
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/bionic64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/bionic64' version '20221207.0.0' is up to date...
==> default: Setting the name of the VM: vagrant_default_1673554661577_99641
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
```

FIGURE 1 – Paramètres de connexion en SSH de la VM

Si vous n'avez rien de particulier à faire sur la VM, déconnectez-vous.

2.3 Etude succincte du fichier Vagrantfile

Ouvrez le fichier Vagrantfile dans un éditeur. Il contient principalement des commentaires et se résume à :

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
end
```

La syntaxe du fichier de configuration est propre au langage Ruby dans lequel est écrit Vagrant. Il n'est pas nécessaire de connaître Ruby pour utiliser Vagrant et pour éditer un Vagrantfile.

2.4 Test des changements d'état

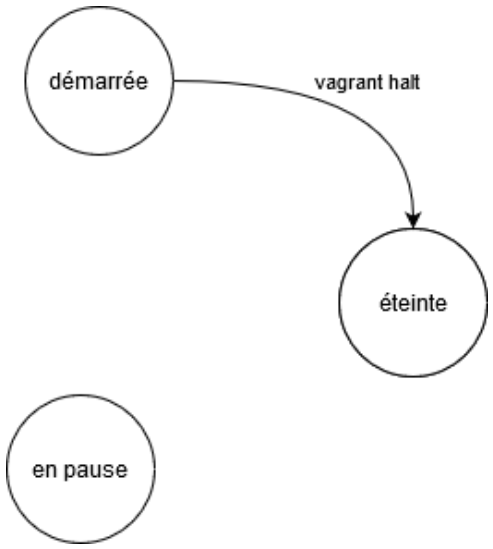

FIGURE 2 – Etats d'une VM de Vagrant, et commandes qui permettent de passer d'un état à un autre (à compléter)

On a vu plusieurs actions passées à la commande vagrant :

— `vagrant init` crée le fichier Vagrantfile

— `vagrant up` démarre la machine virtuelle à partir du fichier Vagrantfile et télécharge le fichier "box" si nécessaire.
— `vagrant ssh` permet de s'y connecter

On peut aussi utiliser `vagrant status` pour connaître l'état de la machine. Celle-ci peut être démarrée, éteinte ou en pause.

Il existe d'autres actions comme : `vagrant halt`, `vagrant suspend`, `vagrant resume` et `vagrant reload`.

La figure 2 indique les différents états dans lesquels la machine peut se trouver. Complétez ce diagramme en indiquant les transitions entre les états et les appels à la commande `vagrant` qui déclenchent ces transitions.

2.5 Provisionnement avec le shell et redirection de ports

Vers la fin du fichier VagrantFile se trouvent les lignes suivantes :

```
# Enable provisioning with a shell script. Additional provisioners such as
# Ansible, Chef, Docker, Puppet and Salt are also available. Please see the
# documentation for more information about their specific syntax and use.
# config.vm.provision "shell", inline: <<-SHELL
# apt-get update
# apt-get install -y apache2
# SHELL
```

Décommentez les quatre dernières lignes de ce bloc. La suite dépend de l'état de la machine virtuelle :
— Si la machine est démarrée, exécutez la commande `vagrant provision`
— Sinon, exécutez la commande `vagrant up`
— Si la machine est en pause, il faudra peut-être faire `vagrant up` puis `vagrant provision`

La machine virtuelle exécute ensuite les commandes shell `apt-get update` et `apt-get install -y apache2` qui sont données dans le fichier Vagrantfile. Vous pouvez suivre l'avancement de ces commandes sur votre terminal.

Ces commandes installent le package "apache2" sur la VM, mais vous ne pouvez pas vous y connecter car le port 80 de la VM n'est pas redirigé vers un port de votre poste de travail. Pour celà il faut décommenter la ligne suivante du Vagrantfile :

```
# config.vm.network "forwarded_port", guest: 80, host: 8080
```

Relancez ensuite la VM avec `vagrant reload`, vérifiez dans les informations affichées par Vagrant que le port 80 est redirigé, puis connectez-vous au port 8080 de votre machine locale avec un navigateur web. La page par défaut du serveur web Apache doit s'afficher.

Remarque : Si le port 8080 de la machine hôte n'est pas disponible alors Vagrant vous en informe et en propose un autre.

Travail personnel : Vous allez modifier la manière dont la VM se connecte au réseau. Dans l'état actuel elle se connecte en mode "NAT" (*Network Address Translation*), sur un réseau local privé auquel seuls Virtualbox et la VM ont accès. On doit donc utiliser la redirection de ports pour se connecter à un service hébergé sur la VM, par exemple SSH ou HTTP. Il existe aussi le mode "Bridged" qui ajoute la VM sur le même réseau local que l'hôte. Le fichier Vagrantfile contient dans ses commentaires toutes les informations pour passer en mode "Bridged".
Faites les modifications nécessaires pour le mode "Bridged" ; désactivez la redirection de ports ; redémarrez la VM pour prendre en compte les modifications ; récupérez l'adresse IP de la VM sur le réseau local (par exemple en appelant `ip a` dans la partie `config.vm.provision`) ; puis connectez votre navigateur web au port 80 de cette adresse.

Vous pourrez ensuite détruire cette VM avec `vagrant destroy`.

2.6 Instancier plusieurs VM

Vous allez maintenant travailler dans un nouveau répertoire que vous nommerez `vagrant_ex2`.

On peut instancier plusieurs VM avec le même fichier Vagrantfile comme ceci :

```
Vagrant.configure("2") do |config|
  config.vm.define "vm-web" do |web|
    web.vm.box = "ubuntu/bionic64"
  end

  config.vm.define "vm-sql" do |sql|
    sql.vm.box = "ubuntu/bionic64"
  end
end
```

Il faut alors indiquer le nom de la machine aux commandes `vagrant`, par exemple
`vagrant ssh vm-web`

Les fichiers propres aux différentes machines se trouvent désormais dans `.vagrant/machines/vm-web/` et dans `.vagrant/machines/vm-sql/`

Vérifiez que les deux machines se lancent avec le nouveau fichier Vagrantfile, et connectez-vous à l'une d'entre elles par la commande `ssh`.

2.7 Création d'un réseau privé

Vous allez maintenant créer un réseau privé virtuel entre les deux machines, comme vous l'aviez fait [3] lors de l'installation de Wordpress sur Proxmox.

Sur ce réseau vous associerez l'adresse 192.168.56.10 à vm-sql et 192.168.56.11 à vm-web. Ajoutez pour cela les lignes suivantes au Vagrantfile :

3. Ou comme vous l'*auriez* fait si le proxy avait été moins capricieux !

```
web.vm.network "private_network", ip: "192.168.56.11"
sql.vm.network "private_network", ip: "192.168.56.10"
```

Chacune des deux lignes va dans le bloc de définition de la VM correspondante. Vous pouvez relancer les deux machines et prendre en compte les changements avec un seul appel à `vagrant reload vm-web vm-sql`

Connectez-vous par SSH et vérifiez que les machines peuvent se "pinger" par leurs adresses privées : sur vm-web la commande `ping 192.168.56.10` retourne des résultats, et pareil pour `ping 192.168.56.11` sur vm-sql.

2.8 Modification du nom interne des VM

En vous connectant aux VM en SSH vous pouvez voir que les deux s'appellent "vagrant". Pour leur donner des noms plus explicites insérez les lignes suivantes au Vagrantfile, encore une fois chaque ligne va dans le bloc de définition de la VM correspondante :

```
web.vm.hostname = "vm-web"
sql.vm.hostname = "vm-sql"
```

Redémarrez les VM et vérifiez que le nouveau nom est pris en compte en vous connectant par SSH : le nouveau nom doit apparaître dan l'invite de commande.

2.9 Provisionnement avec Ansible

L'outil de provisionnement Ansible est dit "sans agent", c'est à dire qu'il n'y a pas d'application ni de service à installer sur les machines à configurer. Ansible se connecte à la machine à provisionner par SSH, il lui suffit donc d'y avoir accès.

L'utilisation Ansible est intégrée dans Vagrant : il n'est pas nécessaire de lui donner les paramètres de SSH pour le faire fonctionner, il suffit d'indiquer un fichier "playbook" dans un bloc "config.vm.provision".

Les fichiers "playbook" [4] sont les fichiers de données de Ansible au format YAML. Ils contiennent des listes de tâches qui indiquent l'état dans lequel la machine à provisionner doit se trouver à la fin du provisionnement.

Vous n'étudierez pas dans ce TP la syntaxe du format YAML. Retenez juste que la structure logique du fichier est donnée par l'indentation, comme en langage Python. Veillez donc à respecter les colonnes auxquelles commencent les lignes de texte et à ne pas utiliser de tabulations dans les fichiers playbook.

2.9.1 Playbook pour vm-web

Le playbook pour vm-web est donné à l'annexe B. Enregistrez-le dans le répertoire du Vagrantfile avec le nom `playbook_web.yaml`.

4. Le nom "playbook" vient du champ linguistique du sport, pensez à une liste d'actions à mener pour atteindre un but.

Les différentes étapes sont indiquées en commentaires dans le fichier. La syntaxe des fichiers playbook se trouve sur le site de l'éditeur à l'adresse `https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_intro.html#playbook-syntax`

Travail personnel : Parcourez la liste des tâches de `playbook_web.yaml` et mettez-les en relation avec les différentes étapes d'installation de Wordpress sur le site `https://ubuntu.com/tutorials/install-and-configure-wordpress`

2.9.2 Playbook pour vm-sql

Le playbook pour vm-sql est donné à l'annexe C. Enregistrez le fichier dans le répertoire du Vagrantfile avec le nom `playbook_sql.yaml`.

Travail personnel : Parcourez la liste des tâches de `playbook_sql.yaml` et mettez-les en relation avec les différentes étapes d'installation de Wordpress sur le site `https://ubuntu.com/tutorials/install-and-configure-wordpress`

Pour créer le fichier `wordpress.conf` (étape 4 sur le site ubuntu.com), vous allez copier un fichier depuis l'hôte vers vm-web. Créez un répertoire `data` et ajoutez-y un fichier nommé `wordpress.conf` dont le contenu est :

```
<VirtualHost *:80>
    DocumentRoot /srv/www/wordpress
    <Directory /srv/www/wordpress>
        Options FollowSymLinks
        AllowOverride Limit Options FileInfo
        DirectoryIndex index.php
        Require all granted
    </Directory>
    <Directory /srv/www/wordpress/wp-content>
        Options FollowSymLinks
        Require all granted
    </Directory>
</VirtualHost>
```

Travail personnel : Retrouvez la tâche Ansible qui correspond à la copie de ce fichier depuis la machine locale vers la VM.

2.9.3 Variables communes aux deux playbooks

Certaines informations (notamment le nom de la base de données, le nom de l'utilisateur et son not de passe) sont communes aux deux machines. Pour éviter de les enregistrer en double et de risquer des incohérences, ces informations sont dans un fichiers de variables qui est inclus par les deux playbooks.

Créez un fichier envvars.yaml dont le contenu est :

```
---
wordpress_mysql_user: "wordpress"
```

```
wordpress_mysql_password: "wordpress"
wordpress_mysql_database: "wordpress"
wordpress_mysql_host: "192.168.56.10"
proxy_env:
  http_proxy: http://c3-login:p4ssw0rd@192.168.0.2:3128
  https_proxy: http://c3-login:p4ssw0rd@192.168.0.2:3128
```

Adaptez la valeur des variables `http_proxy` et `https_proxy` à vos paramètres de configuration.

Travail personnel : Recherchez quelle est la tâche qui utilise la variable proxy_env. À votre avis pourquoi cette tâche a-t-elle besoin de ces informations.

2.9.4 Paramétrage de Vagrant pour Ansible

Vous devez indiquer à Vagrant que le provisionnement de chaque VM se fait par Ansible avec un fichier playbook dédié.

Ajoutez le bloc suivantes au fichier Vagranfile, dans la section de configuration de vm-web :

```
web.vm.provision "ansible_local" do |ansible|
   ansible.playbook = "playbook_web.yaml"
end
```

Puis ajoutez le bloc suivant à la section de configuration de vm-sql :

```
sql.vm.provision "ansible_local" do |ansible|
   ansible.playbook = "playbook_sql.yaml"
end
```

2.9.5 Instanciation et provisionnement des VM

À cette étape votre répertoire de travail doit être organisé comme à la figure 3.

```
.
├── envvars.yaml
├── playbook_sql.yaml
├── playbook_web.yaml
├── Vagrantfile
│
└── data
    └── wordpress.conf
```

FIGURE 3 – Organisation des différents répertoires et fichiers pour la création du site Wordpress avec Vagrant et Ansible

Selon leur état actuel, vous pouvez les démarrer avec :
`vagrant up vm-web vm-sql`
ou juste les provisionner avec :
`vagrant provision vm-web vm-sql`

Le provisionnement met un peu de temps. Quand il se termine, Ansible affiche un message qui résume le travail effectué : nombre de tâches exécutées, nombre de tâches ayant entrainé des modifications, nombre de tâches échouées, etc. Le nombre de tâches échouées doit être de 0.

Vous pouvez maintenant vous connecter au site avec un navigateur web. Avez-vous pensé à rediriger le port 80 de vm-web ? Il faudra peut-être modifier le fichier Vagrantfile et recharger vm-web.

2.10 Utilisation de virt-manager

Comme indiqué au début de ce document, Vagrant peut travailler avec plusieurs solutions de virtualisation. Sous Linux, les principales sont Virtual Box et libvirt (dont l'interface graphique s'appelle virt-manager). Les boxes ne sont pas compatibles avec tous les hyperviseurs, nous allons chercher sur le site `https://app.vagrantup.com/boxes/search` une box Ubuntu 18.04 compatible avec libvirt.

Une fois la box trouvée, modifiez le fichier Vagrantfile pour utiliser cette box et démarrez-la avec
`vagrant up --provider=libvirt`

Vous pouvez vérifier que la VM est lancée avec libvirt avec l'interface graphique de virt-manager ou avec la commande shell `virsh list`

Remarque : il sera peut-être judicieux d'essayer libvirt sur une VM simple avant de l'utiliser pour Wordpress. Dans ce cas travaillez dans un autre répertoire et ne supprimez pas les fichiers liés à Wordpress.

3 Travail sous système Windows

Le travail sous Linux est terminé, vous pouvez dès maintenant faire la phase de nettoyage de la section 4, ce qui vous épargnera un redémarrage de l'ordinateur en fin de séance.

Avant de redémarrer le poste sous Windows, sauvegardez une copie de vos fichiers de configuration. Vous devez avoir trois fichiers `.yaml`, un fichier `Vagrantfile` et un fichier `.conf`.

Vous pouvez ensuite supprimer les répertoire `vagrant_ex*` et redémarrer l'ordinateur. Choisissez l'OS Windows.

3.1 Installation de Vagrant et Ansible

Si Vagrant n'est pas installé sur votre poste Windows vous pouvez le télécharger sur `https://developer.hashicorp.com/vagrant/downloads`. Vérifiez après l'installation que Vagrant est configuré dans le PATH en appelant `vagrant --version`

Il n'est pas nécessaire d'installer Ansible car Vagrant télécharge les fichiers dont il a besoin quand cela est nécessaire.

3.2 Reprise des fichiers Vagrant et Ansible

Recréez l'arborescence indiquée à la section 2.9.5. Vérifiez que le fichier Vagrantfile utilise la box "ubuntu/bionic64" qui n'est compatible qu'avec Virtual Box puis exécutez la commande `vagrant up vm-web vm-sql`
depuis le répertoire du Vagrantfile. Les VM doivent se lancer, vérifiez que Wordpress fonctionne.

3.3 Essais avec VMWare Workstation

Éditez le Vagrantfile pour utiliser une box compatible VMWare.

Installez le plug-in VMWare workstation de Vagrant avec la commande
`vagrant plugin install vagrant-vmware-desktop`

Appelez ensuite `vagrant up --provider=vmware-desktop`. Vérifiez avec l'interface graphique de VMWare que les machines se sont lancées correctement.

4 Nettoyage

Avant de quitter la salle, supprimez les fichiers téléchargés et les programmes que vous aurez installés. Veillez à supprimer les machines virtuelles avec `vagrant destroy` et les images utilisées avec `vagrant box remvove`, sous Windows et sous Debian.

Supprimez aussi le répertoires `vagrant_ex1` et `vagrant_ex2`, ainsi que leur contenu.

5 Conclusion

Dans ce TP vous avez appris à instancier des machines virtuelles avec Vagrant, à exécuter des scripts lors du premier lancement de la machine, à rediriger des ports et à configurer la machine sur le réseau local.

Vous avez ensuite vu comment un seul fichier Vagrantfile permet d'instancier plusieurs machine et comment Ansible permet de provisionner automatiquement ces machines.

Enfin, vous avez vu comment choisir un fournisseur de virtualisation autre que Virtual Box, sous Linux et sous Windows.

Annexes

A Documentation de Vagrant

La documentation de Vagrant se trouve sur le site officiel `https://www.vagrantup.com/`.

Vous y trouverez en particulier des informations sur (cliquez pour ouvrir les pages concernés) :

- La syntaxe des principales actions sur les VM :
  - `vagrant up`
  - `vagrant halt`
  - `vagrant suspend`
  - `vagrant resume`
  - `vagrant reload`
  - `vagrant provision`
  - `vagrant destroy`
- Les commandes `vagrant box` pour charger et supprimer des "boxes"
- La syntaxe complète pour les Vagrantfile

B Contenu du fichier playbook_web.yaml

```
---
- hosts: all
  vars_files:
    - envvars.yaml
  become: true
  tasks:
  # Installation des packages pour Apache et PHP
    - name: Installation des packages Apache
      apt: name=apache2 cache_valid_time=3600 state=latest
    - name: Installation des packages PHP
      apt: name=php cache_valid_time=3600 state=latest
    - name: Installation des packages pour exécuter PHP avec Apache
      apt: name=libapache2-mod-php cache_valid_time=3600 state=latest
    - name: Installation de PHP pour MySQL
      apt: name=php-mysql cache_valid_time=3600 state=latest
    - name: Autres paquets pour Wordpress
      apt:
        name:
          - ghostscript
          - php-bcmath
          - php-curl
          - php-imagick
          - php-intl
          - php-json
          - php-mbstring
          - php-xml
          - php-zip
        state: latest
        cache_valid_time: 3600
    - name: Installation de Avahi
```

```
  apt: name=avahi-daemon cache_valid_time=3600 state=latest

# Création du répertoire pour Wordpress
- name: Creation du répertoire /srv/www
  file:
    path: /srv/www
    state: directory
    owner: www-data
    group: www-data
    mode: '0755'
# Programme et données statiques de Wordpress
- name: Téléchargement et installation des fichiers de Wordpress
  unarchive:
    owner: www-data
    group: www-data
    mode: '0755'
    src: https://wordpress.org/latest.tar.gz
    dest: /srv/www
    remote_src: yes
  environment: "{{ proxy_env }}"

# Paramètre du site pour Apache
- name: Fichier de configuration pour Apache
  copy:
    src: /vagrant/data/wordpress.conf
    dest: /etc/apache2/sites-available

# Configuration de apache (modles et sites)
- name: enabled mod_rewrite
  apache2_module: name=rewrite state=present
- name: a2ensite wordpress
```

```
  command: a2ensite wordpress
- name: a2dissite 000-default
  command: a2dissite 000-default
- name: Redémarrage de Apache
  service: name=apache2 state=restarted

# Configuration de Wordpress pour la DB
- name: Création du fichier de configuration de Wordpress
  copy:
    src: /srv/www/wordpress/wp-config-sample.php
    dest: /srv/www/wordpress/wp-config.php
    owner: www-data
- name: Fichier de config WP - nom de la base
  replace:
    path: /srv/www/wordpress/wp-config.php
    regexp: "database_name_here"
    replace: "{{wordpress_mysql_database}}"
- name: Fichier de config WP - nom de l'utilsateur
  replace:
    path: /srv/www/wordpress/wp-config.php
    regexp: "username_here"
    replace: "{{wordpress_mysql_user}}"
- name: Fichier de config WP - mdp de l'utilsateur
  replace:
    path: /srv/www/wordpress/wp-config.php
    regexp: "password_here"
    replace: "{{wordpress_mysql_password}}"
- name: Fichier de config WP - serveur MySQL
  replace:
    path: /srv/www/wordpress/wp-config.php
    regexp: "localhost"
```

```
      replace: "{{wordpress_mysql_host}}"
  - name: Supprmer les lignes avec les clés et les sels
    lineinfile:
      path: /srv/www/wordpress/wp-config.php
      regexp: 'put your unique phrase here'
      state: absent
  - name: Ajouter les lignes key/salt au fichier de config
    shell: 'curl https://api.wordpress.org/secret-key/1.1/salt/ >> /srv/www/wordpress/wp-config.php'
```

C Contenu du fichier playbook_sql.yaml

```
---
- hosts: all
  vars_files:
    - envvars.yaml
  become: true
  tasks:
    - name: Installation de Avahi
      apt: name=avahi-daemon cache_valid_time=3600 state=latest
  # Installation de MySQL
    - name: Installation de MySQL
      apt: name=mysql-server cache_valid_time=3600 state=latest
    - name: Installation de Python 3
      apt: name=python3 cache_valid_time=3600 state=latest
    - name: Installation de Python pour MySQL, requis pour la suite
      apt: name=python-pymysql cache_valid_time=3600 state=latest
    - name: Création de la base pour Wordpress
      mysql_db:
        name: '{{wordpress_mysql_database}}'
        state: present
        collation: utf8_general_ci
```

```
    login_unix_socket: /var/run/mysqld/mysqld.sock
- name: Création de l'utilisateur pour Wordpress
  mysql_user:
    name: '{{wordpress_mysql_user}}'
    host: '%'
    password: '{{wordpress_mysql_password}}'
    priv: '{{wordpress_mysql_database}}.*:SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER'
    state: present
    login_unix_socket: /var/run/mysqld/mysqld.sock

# Configuration de MySQL
- name: Permettre la connexion depuis une machine distante
  ini_file:
    path: /etc/mysql/mysql.conf.d/mysqld.cnf
    section: mysqld
    option: 'bind-address'
    value: '0.0.0.0'
    state: present
- name: Redémarrage de MySQL
  service: name=mysql state=restarted
```