IVC-Vagrant-1 : Création d'environnements virtuels avec Vagrant

Lom M. HILLAH

Table des matières

1 Introduction

Travaux pratiques sur la création d'environnements virtuels avec Vagrant.

Pour lire une introduction rapide sur Vagrant :

- Wikipedia https://en.wikipedia.org/wiki/Vagrant_%28software%29
- Sur le site de Vagrant : https://www.vagrantup.com/intro

1.1 Fichiers sources pour cet atelier

Tous les fichiers sources utiles pour réaliser les travaux dans cet atelier sont fournis dans l'archive ou le répertoire que vous avez téléchargé, qui contient le présent document.

1.2 Objectifs

Nous allons étudier :

- la création et l'initialisation d'un projet d'environnement virtuel avec Vagrant ;
- l'importation une distribution Linux de base à utiliser comme système d'exploitation de l'environnement virtuel ;
- le contrôle de la machine virtuelle (VM) créée grâce à Vagrant :
  - démarrage, arrêt, suspension et reprise,
  - connexion à la VM via SSH,
- la configuration des éléments d'intégration entre notre machine hôte et la VM contrôlée grâce à Vagrant :
  - redirection de port entre l'hôte et la VM,
  - synchronisation de répertoires dans la VM et la machine hôte,
  - configuration réseau,
- la connexion à une VM contrôlée par Vagrant avec ssh.

1.3 Conventions

Nous utiliserons principalement la ligne de commande dans un terminal. Le prompt sera symbolisé par :

```
$>
```

Les configurations seront indiquées dans une police monofont comme dans l'exemple suivant :

```
config.vm.synced_folder "/Users/aUser/deployed-assets/", \
"/var/www/assets", disabled: true
```

Dans l'exemple ci-dessus, le caractère \ indique que la configuration est écrite sur une seule ligne. Le passage à la ligne avec \ est employé pour améliorer la lisibilité et éviter la troncation de la ligne par la marge.

1.4 Pré-requis

Il faut s'assurer que VirtualBox et Vagrant sont bien installés dans votre système :

- VirtualBox : https://www.virtualbox.org/
- Vagrant : https://www.vagrantup.com/

1. Lancez VirtualBox. S'il démarre bien, éteignez-le ensuite. Vous n'aurez pas besoin de démarrer explicitement VirtualBox avant d'utiliser Vagrant. Ce dernier se lancera automatiquement (sans lancer interface graphique).
2. Ouvrez un terminal, et vérifiez que Vagrant est accessible dans votre `$PATH` :

```
$> vagrant -v
```

La commande ci-dessus rapportera la version de vagrant installée dans votre système.

2 Créer et initialiser un projet Vagrant

- Créez un répertoire dans votre `home directory` (cela dépend de votre système, p. ex. `Mes Documents` sous Windows) ou quelque part d'autre dans votre système de fichiers, là où vous souhaitez travailler sur cet atelier. Dans la suite de cet atelier nous supposons que l'utilisateur de Windows saura transposer les commandes utilisées à son système, y compris en utilisant d'autres interfaces que le terminal (p. ex., PowerShell).

Créez le répertoire suivant et positionnez-vous-y :

```
$> mkdir -p devenv_vagrant && cd devenv_vagrant
```

Le nouveau répertoire servira de **répertoire projet** à Vagrant.

- Créons un `Vagrantfile` pour initialiser le projet :

```
$> vagrant init
```

Vagrant créera un fichier `Vagrantfile` dans ce répertoire et affichera un message de confirmation. Vous trouverez dans ce fichier une configuration de base dans laquelle la plupart des lignes sont commentées. Ces lignes commentées sont des exemples de configuration que l'utilisateur pourra réutiliser à sa convenance. La version de l'API de Vagrant utilisée est indiquée entre parenthèses dans `Vagrant.configure(2)do |config|`, et la distribution de base pour l'OS est inconnue pour le moment : `config.vm.box = "base"`.

NB : Pour créer un `Vagrantfile` minimaliste, sans les lignes commentées fournies à titre d'exemple, vous pouvez (supprimez d'abord le `Vagrantfile` précédemment généré) :

```
$> vagrant init -m
```

- Pour démarrer une machine avec Vagrant, utilisez :

```
$> vagrant up
```

Puisque Vagrant ne connaît pas la machine `box`, le démarrage échouera. Les distributions d'OS (Linux) sont gérées globalement sur votre ordinateur, chaque projet Vagrant disposant de sa propre configuration dans le répertoire que vous avez créé pour le projet (VM démarrée avec une copie de la distribution qui vous intéresse).

Les distributions sont fournies en ligne dans le dépôt HashiCorp. HashiCorp est l'entreprise qui édite Vagrant et gère le dépôt des distributions d'OS pour Vagrant. Les distributions fournies sont listées à l'adresse suivante : https://app.vagrantup.com/boxes/search.

La distribution basée sur Ubuntu,`bento/ubuntu-20.04` est fortement recommandée pour cette version de l'atelier.

Il existe d'autres distributions. Vous pourrez choisir lors de vos futurs travaux celle qui vous paraît appropriée. Par exemple, `ubuntu/bionic64` (c.-à-d. Ubuntu 18.04 LTS 64-bit), ou `centos/7`, ou encore `bento/fedora-33`.

Il est possible d'indiquer à Vagrant la distribution que nous souhaitons utiliser lors de l'initialisation :

```
$> vagrant init bento/ubuntu-20.04
```

Dans cette commande, la partie avant le caractère \ indique le nom du fournisseur de la distribution (ici bento). Le nom après \ est celui de la distribution.

Pour forcer directement le remplacement du `Vagrantfile existant`, vous pouvez :

```
$> vagrant init -m --force bento/ubuntu-20.04
```

NB : cette distribution fonctionne qu'avec le provider (de l'hyperviseur) VirtualBox, ainsi que parallels et vmware. Si vous voulez utiliser un autre provider, cherchez une distribution compatible avec ce dernier à l'adresse : https://app.vagrantup.com/boxes/search

2.1 Exemple de configuration dans Vagrantfile

Voici par exemple un extrait de la configuration (éditée) de VM dans le `Vagrantfile` fournie pour cet atelier :

```
Vagrant.configure(2) do |config|
  config.vm.box = "bento/ubuntu-20.04"
  config.vm.network :forwarded_port, guest: 80, host: 8088
  config.vm.provider :virtualbox do |vb|
    vb.customize ["modifyvm", :id, "--memory", "1024"]
  end
end
```

La configuration ci-dessus indique que la distribution à utiliser pour construire l'environnement est `bento/ubuntu-20.04`. C'est la seule configuration requise pour démarrer une VM.

Le reste de la configuration fournie indique la redirection du port 8088 de votre machine (l'hôte) vers le port 80 de la VM (le guest). Toute connexion à `http://localhost:8088` sur la machine hôte sera redirigée vers le port 80 de la VM. Enfin, nous indiquons au provider VirtualBox d'allouer 1024 Mo au plus de mémoire à la VM.

NB : Si la distribution n'est pas dans le dépôt géré par HashiCorp, alors on peut spécifier son URL. Par exemple :

```
config.vm.box_url = "https://files.myprovider.com/bionic64.box"
```

Si la distribution est déjà installée dans votre système et dans le répertoire courant, alors il suffit juste d'indiquer son nom au lieu d'une URL complète. On peut aussi utiliser une URI de la forme :

```
file://chemin/absolu/bionic64.box
```

Il est également possible d'indiquer la version de la distribution :

```
config.vm.box_version = "20210114.0.0"
```

2.2 Documentation sur le Vagrantfile et Vagrant

- Vagrantfile : https://www.vagrantup.com/docs/vagrantfile
- Documentation générale sur Vagrant : https://www.vagrantup.com/docs

3 Démarrer une machine virtuelle

Pour démarrer une VM contrôlée par Vagrant, une seule commande :

```
$> vagrant up
```

ou pour démarrer une VM et exécuter les instructions de provisioning :

```
$> vagrant up --provision
```

3.1 Provider

Pour spécifier un provider directement en ligne de commande :

```
$> vagrant up --provider=virtualbox
```

La variable d'environnement `VAGRANT_DEFAULT_PROVIDER` permet également de spécifier votre préférence de provider.

Avec la commande `up`, Vagrant cherchera l'environnement spécifié dans le `Vagrantfile` sur la machine hôte (i.e., votre machine) ; si elle n'y est pas il le téléchargera du dépôt de Hashicorp.

La VM démarrée est headless, c.-à-d., elle n'ouvre pas d'interface utilisateur graphique. Vous pouvez cependant vérifier que la VM est bien démarrée via l'interface graphique de VirtualBox, en lançant vous-même Virtualbox comme vous le feriez normalement pour n'importe quelle application.

Si vous souhaitez configurer une VM avec interface graphique, il faut lui allouer au moins de la mémoire vidéo dans la configuration du provider dans le `Vagrantfile`.

Pour résumer, avec la commande `vagrant up` :

- Vagrant vérifie d'abord s'il existe un environnement déjà en place, tel que spécifié dans le `Vagrantfile`. Si un environnement existant pour ce projet fut suspendu, il sera redémarré.
- Si aucun environnement n'existe pour ce projet, Vagrant vérifiera s'il a été téléchargé, sinon le téléchargera du dépôt en ligne.
- L'environnement téléchargé sera stocké sur votre machine pour une réutilisation future, copié pour chaque nouveau projet (y compris le projet courant pour lequel il est téléchargé pour la première fois).

- Une VM sera créée avec le provider par défaut actif sur votre machine (ici VirtualBox, mais VMWare, Parallels, Docker et d'autres sont également supportés).
- La redirection de port sera mise en place. Par défaut, le port 22 (SSH) sur votre machine sera redirigé vers le port 2222 sur la VM hébergée sur votre machine. Nous pourrons alors nous connecter plus tard à la VM par SSH **(username : vagrant, authentication method : private key)**.
- La VM sera démarrée.
- Le réseau sera activé sur la VM, pour rendre possible les communications entre l'hôte et la VM.
- Les répertoires partagés et synchronisés entre l'hôte et la VM seront montés dans la VM. Par défaut, le répertoire du projet sur l'hôte (où se trouve le Vagrantfile) sera synchronisé avec `/vagrant` dans la VM.
- Vagrant lancera enfin l'outil de provisioning appelé dans la configuration du `Vagrantfile`.

4 Redémarrer une machine virtuelle en cours d'exécution

Si la configuration dans le `Vagrantfile` a changé pendant l'exécution de la VM, vous pouvez la redémarrer avec :

```
$> vagrant reload
```

5 Suspendre une machine virtuelle

Pour suspendre la VM du projet courant et sauvegarder son état courant sur votre disque, utilisez :

```
$> vagrant suspend
```

La VM n'utilisera plus de resource CPU ou mémoire sur la machine hôte.

6 Redémarrer une machine virtuelle suspendue

Pour redémarrer la VM précédemment suspendue dans ce projet, utilisez :

```
$> vagrant resume
```

6.1 Arrêter une machine virtuelle

Pour arrêter la VM du projet courant, utilisez :

```
$> vagrant halt
```

7 Tout reprendre à partir de rien

Si vous constatez des problèmes de stabilité de votre VM dans le projet courant (ceci peut arriver pour tout un tas de raisons), pas de panique, il est simple de tout recommencer à zéro grâce à :

```
$> vagrant destroy
```

Cette commande supprime la VM courante sans supprimer le fichier de configuration `Vagrantfile`, qui est un véritable asset. Puis, utilisez comme d'habitude :

```
$> vagrant up
```

pour recréer une nouvelle copie et la démarrer.

7.1 Bonnes pratiques

Cette situation met en évidence quelques **bonnes pratiques** à mettre en place :

- versionner votre fichier de configuration (sur git ou svn, etc.) ;
- ne rien mettre dans votre VM qui sera irrécupérable si celle-ci est supprimée ;

- vous devez pouvoir supprimer une VM instable et en créer une nouvelle copie sans préjudice pour votre projet.

Autrement dit, ne cédez pas à la tentation de configurer à la main votre VM une fois démarrée, c.-à-d., en allant taper dans le terminal de la VM les commandes d'installation de packages de logiciels.

Vous pouvez même aller plus loin dans la remise à zéro de votre environnement en supprimant également le **box** utilisé dans le catalogue présent sur votre machine (étudié dans la section Gestion des distributions d'OS et également dans le troisième atelier) :

```
vagrant box remove nom-du/box
```

8 Répertoires synchronisés

Les répertoires synchronisés vous permettront de partager des fichiers entre la machine hôte et la VM. Par défaut, Vagrant monte le répertoire contenant le `Vagrantfile` (répertoire projet donc) sur `/vagrant` dans la VM.

Si vous souhaitez synchroniser d'autres répertoires, il faut utiliser la configuration suivante dans le `Vagrantfile` (la répéter, une par ligne pour chaque répertoire synchronisé) :

```
config.vm.synced_folder "/Users/aUser/deployed-assets/", \
  "/var/www/assets"
```

Le premier chemin est local, c.-à-d., sur la machine hôte. Le deuxième chemin est le point de montage dans la VM. Il est possible, pour le chemin local, d'indiquer un chemin relatif au répertoire du projet Vagrant (où se trouve le `Vagrantfile`).

On peut désactiver une synchronisation spécifique avec :

```
config.vm.synced_folder "/Users/aUser/deployed-assets/", \
  "/var/www/assets", disabled: true
```

Il est également possible de spécifier le `owner` et le `group` du chemin monté dans la VM :

```
config.vm.synced_folder "/Users/aUser/deployed-assets/", \
  "/var/www/assets", owner: "vagrant", group: "vagrant"
```

9 Configuration du réseau

Par défaut, la VM est accessible uniquement à partir de la machine hôte, pas des autres machines sur le réseau privé local sur lequel la machine hôte est joignable. Si vous souhaitez partager l'accès à la VM à d'autres personnes ou programmes dont les machines hôtes se trouvent dans le même réseau local que votre machine, vous devriez utiliser une configuration comme dans l'exemple suivant (`Vagrantfile`) :

```
config.vm.network "private_network", ip: "10.33.110.112"
```

Il est également possible d'utiliser DHCP, VirtualBox récupérera alors automatiquement une adresse de votre serveur DHCP dans votre réseau privé local et l'assignera à la VM :

```
config.vm.network "private_network", type: "dhcp"
```

10 Gestion de plusieurs VM

On peut gérer plusieurs VM dans le même `Vagrantfile` avec la configuration suivante (à insérer dans le bloc `Vagrant.configure`) :

```
config.vm.define "web" do |web|
    web.vm.box = "apache"
end

config.vm.define "db" do |db|
    db.vm.box = "mysql"
end
```

La configuration ci-dessus définit une VM pour le serveur Web, et une VM pour la base de données.

Pour démarrer une VM spécifique, il faut donner son nom à la commande `vagrant up` :

```
$> vagrant up web
```

Sans le nom de la VM à démarrer, Vagrant les démarrera toutes par défaut.

De même, pour se connecter à une VM spécifique, on utilisera :

```
$> vagrant ssh db
```

Vous pouvez ensuite configurer le réseau pour permettre à ces différentes VM de communiquer via leur adresse IP, par exemple en leur affectant des adresses IP statiques dans la plage réservée aux adresses privées :

```
web.config.vm.network "private_network", ip: "192.168.50.4"
```

NB : Notez l'utilisation de la variable `web` correspondant au bloc de configuration de la machine `web`.

11 Gestion des distributions d'OS

Vagrant gère de manière globale sur votre machine les distributions d'OS (**box**) utilisées pour créer des VM. Pour faire simple, on peut dire qu'une VM est un box instancié, configuré à partir de votre `Vagrantfile`.

La commande principale pour gérer les box est `vagrant box`. Elle possède des sous-commandes spécifiques. Pour les connaître :

```
$> vagrant box --help
```

11.1 Lister les distributions installées

Rien de plus simple :

```
$> vagrant box list
```

Vous devriez retrouver dans cette liste la distribution que vous avez utilisée pour créer la VM courante ainsi que l'indication de son provider (`virtualbox`) et sa version.

11.2 Ajouter une nouvelle distribution

La sous-commande `add` permet d'ajouter une nouvelle box. Elle prend en argument un nom, une URL ou un chemin local vers le fichier de la distribution. Pour expérimenter cette commande, nous avons placé dans le répertoire des fichiers sources de cet atelier une distribution : `precise64.box`.

D'abord arrêter la VM courante :

```
$> vagrant halt
```

Puis, installer la distribution ci-dessus :

```
$> vagrant box add --force hashicorp/precise64 \
  ./precise64.box
```

L'option `--force` indique à Vagrant de forcer le remplacement d'une distribution existante ayant le même nom que celui que nous avons indiqué (c.-à-d., `hashicorp/precise64`).

Après exécution de cette commande avec succès, utilisez de nouveau `vagrant box list` pour lister les distributions existantes.

Il existe d'autres options à la sous-commande `add` :

- `--clean` demande à Vagrant de nettoyer les répertoires temporaires utilisés pour télécharger des fichiers
- `-- provider` demande à Vagrant d'utiliser le provider indiqué après cette option pour supporter la distribution installée. Les providers supportés par Vagrant sont : VirtualBox (par défaut), VMware, Parallels, Amazon EC2 et Docker.

11.3 Vérifier les mises à jour des distributions installées

Il suffit d'utiliser :

```
$> vagrant box outdated
```

11.4 Supprimer une distribution

Pour supprimer une distribution actuellement installée, vous pouvez utiliser la sous-commande `remove`, comme dans l'exemple suivant :

```
$> vagrant box remove hashicorp/precise64 \
   --provider virtualbox
```

Il est également possible d'indiquer la version de la distribution avec l'option `--box-version`.

11.5 Empaqueter une distribution pour la redistribution et la réutilisation

La sous-commande `package` permet de d'empaqueter une distribution que vous avez configurée (par exemple en y installant des logiciels spécifiques) afin de la distribuer pour permettre sa réutilisation, au même titre que celle pré-packagée que nous utilisons tout au long de cet atelier pour créer une VM.

Nous reviendrons un peu plus tard sur ce point.

11.6 Mettre à jour la distribution de l'environnement courant

La sous-commande `update` vous permet de mettre à jour la box de l'environnement Vagrant courant :

```
$> vagrant box update
```

Vous pouvez également mettre à jour une distribution spécifique, qui n'est pas utilisée dans l'environnement courant, en utilisant l'option `--box` pour indiquer son nom. Vous pouvez aussi indiquer son provider si vous le souhaitez avec l'option `--provider`.

Dans le cas d'une distribution installée (p. ex., hashicorp/precise64) provenant d'un fichier local au lieu d'un catalogue en ligne, Vagrant ne pourra pas effectuer la mise à jour. On peut alors utiliser la configuration de l'URL `config.vm.box_url` pour indiquer l'endroit où se trouve le fichier de la box à utiliser pour effectuer la mise à jour.

Dans ce cas, il faut d'abord arrêter la VM, supprimer la distribution installée (p. ex., `hashicorp/precise64`) avant de relancer `vagrant up`.

12 Vagrant et Docker

Vagrant parle Docker ! (il faut avoir installé le gestionnaire Docker sur votre machine hôte).

```
Vagrant.configure("2") do |config|
  config.vm.provider "docker" do |d|
    d.image = "foo/bar"
  end
end
```

Puis, avec la commande `vagrant up --provider=docker` l'image `foo/bar` sera récupérée et démarrée dans son conteneur.

Alternativement, il est également possible de construire une image Docker et de la démarrer dans le conteneur Docker à partir du Dockerfile :

```
Vagrant.configure("2") do |config|
  config.vm.provider "docker" do |d|
    d.build_dir = "."
  end
end
```

La configuration ci-dessus indique que le `Dockerfile` est situé dans le même répertoire que le `Vagrantfile`. Avec la même commande que ci-dessus, `vagrant up --provider=docker` la construction de l'image est effectuée et démarrée dans son conteneur.

L'image est reconstruite avec la commande `vagrant reload`.

13 Lister les environnements actifs

Afin de connaître les environnements actifs (c.-à-d. VM créées sur votre machine dans le cadre de différents projets), la commande `vagrant global-status` vous donnera cette information.

```
$> vagrant global-status
```

Cette commande listera les ID, noms, providers, état d'exécution de la VM associée, ainsi que le répertoire du projet.

Vous pouvez utiliser l'ID de l'environnement pour directement (dans l'exemple suivant on suppose ID = 6ac67c5) :

- le suspendre (`vagrant` **`suspend`** `6ac67c5`),
- le détruire (`vagrant destroy 6ac67c5`), etc., sans avoir besoin d'aller dans le répertoire de son projet.

14 Se connecter à une VM contrôlée par Vagrant

Pour se connecter à la VM en cours d'utilisation, il suffit d'utiliser :

```
$> vagrant ssh
```

Vagrant utilisera la clé privée `insecure` se trouvant dans le répertoire `.vagrant.d` sous votre home directory. Après connexion, vous vous retrouverez dans le répertoire `/home/vagrant` dans la VM.

Un simple `ls /vagrant` vous permettra de constater que `/vagrant` est effectivement synchronisé avec le répertoire de votre projet sur la machine hôte.

Pour fermer la connexion SSH et sortir de la VM, tapez simplement : `Ctrl-D`.

15 Installer des logiciels

Pour installer des logiciels dans votre machine virtuelle, vous pouvez utiliser les commandes usuelles du gestionnaire de package `apt`, comme dans l'exemple suivant :

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install vim
sudo apt-get install apache2
```

```
sudo apt-get install mysql-server libapache2-mod-auth-mysql \
 php5-mysql
sudo mysql_install_db
sudo /usr/bin/mysql_secure_installation
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
sudo apt-get install php5-cgi php5-cli php5-curl php5-common \
 php5-gd php5-mysql
sudo service apache2 restart
```

Une version plus à jour de la configuration ci-dessus est fournie dans le fichier **bootstrap.sh** mis à votre disposition, pour la box `bento/ubuntu-20.04`.

Vous pouvez ainsi configurer votre VM selon vos besoins, avec l'installation et la configuration des logiciels répondant à ces besoins, comme dans l'exemple ci-dessus.

NB : Pour éviter de réaliser cette installation de packages à la main dans le terminal de la VM, il est recommandé de les écrire dans un fichier `.sh`, puis de spécifier le Shell comme `provisioner` (gestionnaire de configuration, étudié dans l'atelier 2), en rajoutant la ligne de configuration suivante (à l'intérieur du bloc `Vagrant.configure`) :

```
Vagrant.configure(2) do |config|
    config.vm.provision :shell, :path => "bootstrap.sh"
end
```

Dans la configuration ci-dessus, le provisioning est spécifié dans le fichier `bootstrap.sh`.

Si votre VM tourne déjà, il suffit de demander à Vagrant de re-provisionner uniquement sans réexécuter toute de la configuration :

```
$> vagrant provision
```

Sinon, vous pouvez démarrer la VM comme d'habitude, en indiquant cette fois-ci l'option `--provision` :

```
$> vagrant up --provision
```

16 Configurateur de Vagrantfile en ligne

Puphpet est un configurateur en ligne bien fait, qui vous fera gagner du temps dans la mise au point de longues et complexes configurations :

- https://puphpet.com
- Github : https://github.com/puphpet/puphpet

Cet outil a été développé initialement pour permettre la gestion de configuration d'environnements de développement en PHP.

Il faut cependant noter que la configuration générée par Puphpet est ainsi beaucoup plus complexe que votre seul `Vagrantfile`. Cette configuration est structurée en une arborescence de fichiers de programmes (Ruby, Bash) et de fichiers de configuration. Le logiciel du provisionner `Puppet` est également inclus. La configuration générée par Puphpet est donc un bundle.

17 Repackager un box

Après avoir réalisé les installations de logiciels comme dans l'exemple ci-dessus, vous pouvez repackager le box afin de le distribuer (par exemple, le publier sur Hashicorp Vagrant Cloud - atelier 3).

Le repackaging s'effectue à l'aide de la commande :

```
$> vagrant package --output ivc.box
```

Vagrant placera le nouveau box empaqueté à l'emplacement indiqué par l'option `--output`.

NB : Vous pouviez, avant de repackager le box, réduire sa taille en supprimant dans la VM les caches (il faut d'abord s'y connecter par la commande `vagrant ssh`) :

```
$> sudo apt-get clean
$> sudo dd if=/dev/zero of=/EMPTY bs=1M
$> sudo rm -f /EMPTY
$> cat /dev/null > ~/.bash_history && history -c && exit
```

17.1 Contenu du fichier .box

Le box obtenu (tout box Vagrant) est un fichier tar.gz que vous pouvez décompresser avec :

```
$> tar xvzf ivc.box
```

Sous Windows, le logiciel 7-zip devrait pouvoir lire ce type d'archive.

Sa décompression donnera essentiellement les fichiers suivants :

- `box-disk1.vmdk` : c'est le disque de la machine virtuel, qui contient l'OS et vos logiciels. C'est le fichier de plus grande taille.
- `box.ovf` : c'est un fichier XML décrivant la configuration du matériel virtuel du box.
- `Vagrantfile` : celui-ci est auto-généré par la commande `package`. Il décrit l'adresse mac de votre box, charge le `Vagrantfile` de l'utilisateur qui utilisera votre box, et indique le chemin du fichier de la clé privée qui servira à se connecter à la VM.

18 Clés utilisées par Vagrant

Pour se connecter automatiquement par SSH à la VM, Vagrant utilise une méthode d'authentification basée sur clés publique/privée. Ces clés sont génériques dans la mesure où elles sont utilisées par défaut sur les box fournis. Si vous souhaitez les changer, suivez les instructions incluses dans la documentation :

18.1 Insecure Keypair

These keys are the "insecure" public/private keypair we offer to base box creators for use in their base boxes so that vagrant installations can automatically SSH into the boxes.

https://github.com/mitchellh/vagrant/tree/master/keys

If you're working with a team or company or with a custom box and you want more secure SSH, you should create your own keypair and configure the private key in the Vagrantfile with `config.ssh.private_key_path`

Dans l'atelier 2 nous générons et intègrerons une nouvelle paire de clés publique/privée dans la VM afin de permettre à Ansible de réaliser automatiquement les tâches de gestion de configuration.

---

19 Feedback

- **Lom M. Hillah (lhillah@parisnanterre.fr)**

20 Licence