LAB 2- Creating Virtualboxes with Vagrant

Qu'est ce que vagrant

Vagrant nous permet de lancer à la demande des VirtualBoxes. Nous pouvons créer un box Vagrant, le configurer et essayer de faire des déploiements sur cette machine virtuelle.

L'utilisation principale de Vagrant est de créer et de configurer des environnements de développement virtuels. Nous pouvons le voir comme un wrapper autour des logiciels de virtualisation tels que VirtualBox , VMware et autour des logiciels de gestion de

configuration tels que Chef, Salt ou Puppet.

Alors, allons-y et voyons comment utiliser Vagrant pour créer une machine virtuelle Linux.

Vagrant vocabulary

1. **Virtualbox / Box** Une application pour exécuter des «machines virtuelles (VM)» sur notre ordinateur portable / bureau. C'est une VM préconfigurée avec un logiciel. Il est utilisé comme unité de départ de toute configuration.

2. **Vagrant** Une application qui automatise la configuration des VM. En bref, Vagrant est un constructeur automatisé de VM.

3. **Host** Notre ordinateur portable / de bureau qui exécute Virtualbox et Vagrant.

4. **Guest / Guest VM / Insance** Une machine virtuelle s'exécutant sur l'hôte.

Vagrantfile

Nous utilisons **Vagrantfile** pour configurer Vagrant

1. Marquer le répertoire racine de notre projet. Une grande partie de la configuration de Vagrant est relative à ce répertoire racine.

2. Décrire le type de machine et de ressources dont nous avons besoin pour exécuter notre projet, ainsi que les logiciels à installer et comment nous voulons y accéder

3. La commande **vagrant init** créera ce **Vagrantfile**.

Vagrant init

La commande **vagrant init** est une commande intégrée et initialise un répertoire à utiliser avec Vagrant.

```
$ mkdir my_vagrant
$ cd my_vagrant/
```

```
$ vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
Cela placera un **Vagrantfile **dans notre répertoire actuel.
$ ls
Vagrantfile
```

Nous pouvons également exécuter **vagrant init** dans un répertoire préexistant pour configurer Vagrant pour un projet existant.

Pour voir la liste des box Vagrant:

```
$ vagrant box list
There are no installed boxes! Use `vagrant box add` to add some.
```

Vagrant Boxes

Au lieu de créer une machine virtuelle à partir de zéro, Vagrant utilise une image de base pour cloner rapidement une machine virtuelle. Ces images de base sont appelées **boxes** dans Vagrant, et la spécification du box à utiliser pour notre environnement Vagrant est toujours la première étape après la création d'un nouveau **Vagrantfile**.

Boxes sont ajoutés à Vagrant avec **vagrant box add**. Cela stocke le box sous un nom spécifique afin que plusieurs environnements Vagrant puissent le réutiliser.

Sélectionnez un box Vagrant sur https://app.vagrantup.com

Nous allons choisir quelques boxes que nous voulons installer. Accédez à "Popular": Pour l'ajouter à notre liste de boxes.

```
$ vagrant box add ubuntu/xenial64
==> box: Loading metadata for box 'ubuntu/xenial64'
box: URL: https://vagrantcloud.com/ubuntu/xenial64
==> box: Adding box 'ubuntu/xenial64' (v20210203.0.0) for provider:
virtualbox
box: Downloading:
https://vagrantcloud.com/ubuntu/boxes/xenial64/versions/20210203.0.0/provider
s/virtualbox.box
Download redirected to host: cloud-images.ubuntu.com
==> box: Successfully added box 'ubuntu/xenial64' (v20210203.0.0) for
```

```
'virtualbox'!
```

Il télécharge simplement l'image mais n'installe rien sur notre système.

Notez que nous avons téléchargé le box nommé 'ubuntu/xenial64' du catalogue de HashiCorp, un endroit où nous pouvons trouver et héberger des boxes.

Notez également que les boxes sont namespaced. Les boxes sont divisées en deux parties - le nom d'utilisateur et le nom du box - séparées par un slash (/). Dans l'exemple ci-dessus, le nom d'utilisateur est "ubuntu" et le box est "xenial64".

Nous pouvons vérifier quels boxes nous avons:

```
$ vagrant box list
ubuntu/xenial64 (virtualbox, 20210203.0.0)
```

On souhaite parfois savoir quels boxes sont stockés **globalement** pour l'utilisateur actuel.

En fait, nous n'avons pas besoin d'ajouter un box à l'aide de la commande **vagrant box add**. Comme cela sera décrit dans la section suivante, si nous **spécifions** quel box utiliser dans **Vagrantfile**, Vagrant le téléchargera.

Maintenant, nous pouvons créer autant de boxes 'ubuntu/xenial64' que nous le voulons tant que notre machine peut le gérer.

Les boxes ajoutés peuvent être réutilisés par plusieurs projets. Chaque projet utilise un box comme image initiale à partir de laquelle cloner et ne modifie jamais l'image de base réelle. Cela signifie que si nous avons deux projets utilisant tous les deux le box 'ubuntu/xenial64' que nous venons d'ajouter, l'ajout de fichiers dans une machine invitée n'aura aucun effet sur l'autre machine.

Using a box - config.vm.box

Maintenant que le box a été ajouté à Vagrant, nous devons configurer notre projet pour l'utiliser omme base. Ouvrez le **Vagrantfile** et modifiez le contenu comme suit:

```
Vagrant.configure(2) do |config|
   config.vm.box = "ubuntu/xenial64"
end
```

Le **"ubuntu/xenial64"** dans ce cas doit correspondre au nom que nous avons utilisé pour ajouter le box ci-dessus. C'est ainsi que Vagrant sait quel box utiliser. Si le box n'a pas été ajouté auparavant, Vagrant téléchargera et ajoutera automatiquement le box lors de son exécution.

Boot Vagrant environment - vagrant up

Il est temps de démarrer notre premier environnement Vagrant. Exécutez ce qui suit:

```
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/xenial64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/xenial64' is up to date...
==> default: Setting the name of the VM:
my_vagrant_default_1486540487227_74029
==> default: Clearing any previously set forwarded ports...
==> 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: 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: Warning: Remote connection disconnect. Retrying...
default: Warning: Remote connection disconnect. Retrying...
default: Warning: Remote connection disconnect. Retrying...
default: Warning: Remote connection disconnect. Retrying...
default:
default: Vagrant insecure key detected. Vagrant will automatically
replace
default: this with a newly generated keypair for better security.
default:
default: Inserting generated public key within guest...
default: Removing insecure key from the guest if it's present...
default: Key inserted! Disconnecting and reconnecting using new SSH
key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
default: The guest additions on this VM do not match the installed version of
default: VirtualBox! In most cases this is fine, but in rare cases it can
default: prevent things such as shared folders from working properly. If you
see
default: shared folder errors, please make sure the guest additions within
the
default: virtual machine match the version of VirtualBox you have installed
on
default: your host and reload your VM.
default:
default: Guest Additions Version: 4.3.36
default: VirtualBox Version: 5.0
==> default: Mounting shared folders...
default: /vagrant => /home/mtbsoft/my_vagrant
$
```

Nous avons maintenant une machine virtuelle exécutant Ubuntu. Nous avons créé un box «ubuntu/xenial64». Nous ne verrons rien cependant, puisque Vagrant exécute la machine virtuelle sans interface utilisateur. Pour vérifier si notre machine virtuelle est en cours d'exécution, nous pouvons SSH dans la machine comme démontré dans une section ultérieure.

Connection timeout

Parfois, on a des problèmes avec la commande **vagrant up** et on obtient ce qui suit:

```
default: Warning: Connection timeout. Retrying...
```

Pour voir plus d'informations, nous devrions activer l'interface graphique:

```
config.vm.provider "virtualbox" do |vb|
   vb.gui = true
end
```

Avant de recommencer **vagrant up**, nous devons nettoyer les restes de la tentative ratée. Exécutez la commande **vagrant halt** (voir la section suivante):

```
$ vagrant halt
==> default: Attempting graceful shutdown of VM...
```

Maintenant, nous pouvons voir à travers VBox GUI pourquoi il a été suspendu, et le démarrage a échoué:

Nous devons donc activer la virtualisation dans le BIOS (touche F10 au démarrage d'Ubuntu).

Après la configuration du BIOS, j'ai pu exécuter vagrant (**vagrant up**) comme indiqué cidessus.

```
vagrant halt
```

La commande vagrant halt arrête la machine en cours d'exécution que Vagrant gère.

Vagrant tentera d'abord à arrêter correctement la machine en exécutant le mécanisme d'arrêt du système d'exploitation guest. Si cela échoue, ou si l' indicateur **--force** est spécifié, Vagrant coupera simplement l'alimentation de la machine.

SSH into Vagrant environment (virtual machine) - I

Lorsque la commande **vagrant up** est terminée, nous avons une machine virtuelle exécutant Ubuntu. Cependant, nous ne voyons rien, car Vagrant exécute la machine virtuelle sans interface utilisateur.

Pour vérifier si notre machine virtuelle est en cours d'exécution, nous pouvons SSH dans la machine:

```
$ vagrant ssh
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 3.13.0-108-generic x86_64)
* Documentation: https://help.ubuntu.com/
```

```
System information disabled due to load higher than 1.0
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
0 packages can be updated.
0 updates are security updates.
New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
vagrant@vagrant-ubuntu-xenial-64:~$
```

Cette commande **vagrant ssh** nous plongera dans une session SSH à part entière.

Remarque: Nous devons faire attention à **rm -rf /**, car Vagrant partage un répertoire à **/vagrant** avec le répertoire sur l 'hôte contenant notre **Vagrantfile**, et cela peut supprimer tous ces fichiers.

Qu'est-ce que nous avons fait?

Avec une seule ligne de configuration et une seule commande dans notre terminal, nous avons mis en place une machine virtuelle entièrement fonctionnelle et accessible par SSH. La session SSH peut être terminée avec CTRL+D ou:

```
$ logout
Connection to 127.0.0.1 closed.
```

Encore une chose, nettoyez les ressources!

Lorsque nous avons fini de manipuler la machine, exécutez **vagrant destroy** sur notre machine hôte, et Vagrant mettra fin à l'utilisation de toutes les ressources par la machine virtuelle.

```
$ vagrant destroy
default: Are you sure you want to destroy the 'default' VM? [y/N] y
==> default: Forcing shutdown of VM...
==> default: Destroying VM and associated drives...
```

SSH without using "vagrant ssh" - II

Dans la section précédente, nous avons appris comment accéder à Vagrant vm en utilisant

la commande "vagrant ssh". Cependant, on peut vouloir parfois ssh dans une VM Vagrant

comme dans d'autres machines:

```
$ ssh vagrant@ip-address
```

Pour ce faire, nous devons configurer deux choses:

1. Copiez **~/.ssh/id_rsa.pub** de notre machine locale dans le fichier **/.ssh/authorized_keys** de VM Vagrant.

2. Attribuez une nouvelle adresse IP à notre instance Vagrant. Ajoutez la ligne suivante au fichier **Vagrantfile** :

```
ruby config.vm.network :private_network, ip: "192.168.68.7"
```

Maintenant, nous pouvons ssh dans la VM Vagrant comme d'habitude:

```
$ ssh vagrant@192.168.68.7
...
System load: 0.0 Processes: 76
Usage of /: 2.9% of 39.34GB Users logged in: 1
Memory usage: 25% IP address for eth0: 10.0.2.15
Swap usage: 0% IP address for eth1: 192.168.68.7
...
vagrant@vagrant-ubuntu-xenial-64:~$
```

Uninstall Vagrant

Nous pouvons désinstaller le binaire Vagrant, les données utilisateur ou les deux:

1. **Suppression du programme Vagrant**:

La suppression du programme Vagrant supprimera le binaire **vagrant** et toutes les dépendances de notre machine. Après avoir désinstallé le programme, nous pouvons toujours réinstaller à nouveau en utilisant les méthodes standards. Supprimez le répertoire **/opt/vagrant** et le fichier **/usr/bin/vagrant**

2. **Suppression des données utilisateur**:

La suppression des données utilisateur supprimera toutes les boxes, plugins et tout état stocké pouvant être utilisé par Vagrant. La suppression des données utilisateur fait effectivement penser à Vagrant qu'il s'agit à nouveau d'une nouvelle installation. Sur chaque plate-forme, supprimez le fichier **~/vagrant.d**pour supprimer les données utilisateur.

Controling virtualbox

1. Les machines en cours d'exécution (renvoie le nom et l'UUID):

```
VBoxManage list runningvms
```

2. Arrêtez d'exécuter les VM en les «hibernant» (recommandé pour éviter la perte de données):

```
VBoxManage controlvm <name|uuid> savestate
```

3. des machines virtuelles (non recommandé car nous pouvons perdre des données dans l'invité):

```
VBoxManage controlvm <name|uuid> poweroff
```

4. ACPI dans un OS invité prenant en charge ACPI (préférable à la mise hors tension pour un arrêt progressif des invités):

```
VBoxManage controlvm <name|uuid> acpipowerbutton
```

Exemple:

```
$ VBoxManage list runningvms
"my_vagrant_default_1417676312044_99144" {e94d4542-9693-42f4-ae9f-8fdd2c1588a7}
$ VBoxManage controlvm my_vagrant_default_1417676312044_99144 savestate
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
$ VBoxManage controlvm e94d4542-9693-42f4-ae9f-8fdd2c1588a7 poweroff
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
$ VBoxManage controlvm e94d4542-9693-42f4-ae9f-8fdd2c1588a7 acpipowerbutton
```

Sample Vagrant commands

Voici quelques commandes Vagrant:

1. **vagrant up**: démarre une machine virtuelle.

2. **vagrant provision**: force les approvisionneurs à exécuter à nouveau une machine virtuelle. Utile pour mettre à jour la configuration des machines virtuelles existantes.

3. **vagrant halt**: Arrêtez une machine virtuelle.

4. **vagrant suspend**: mettre en veille une machine virtuelle.

5. **vagrant destroy**: supprimez une machine virtuelle de notre système.