Vagrant : un gestionnaire de Cloud privé

Guillaume Urvoy-Keller

1er novembre 2022

Vous devez faire un compte-rendu électronique à envoyer à Anderson.LOURENCO-DE-ARAUJO@univ-cotedazur.fr en fin de séance

1 Virtualbox : autopsie d'une VM

1. Créez une nouvelle machine depuis l'interface de Virtualbox : donnez lui comme nom Ubuntu et choisissez toutes les valeurs par défaut.
2. Démarrez la VM (c'est tout !)
3. Listez les processus actifs de VirtualBox et leur rôle en vous aidant du chapitre 10 de l'aide en ligne `https://www.virtualbox.org/manual/ch10.html#technical-components`. **Faites des réponses courtes**
4. Placez-vous dans le répertoire dans lequel VirtualBox stocke les fichiers de la VM que vous venez de créer : analyser chaque fichier (est-ce de l'ASCII, du binaire. Si ascii, quel type de données ?) et donner son rôle. **Faites des réponses courtes**
5. Concernant le fichier contenant le disque : quelle est sa taille et pourquoi diffère-t-elle de la taille par défaut ?

2 Vagrant

2.1 Création d'une première VM Ubuntu

1. Créer un répertoire Vagrant à la racine de votre compte
2. Placez-vous dans ce répertoire et créez un répertoire VM1. Version rapide qui créée les 2 répertoires simultanément : `mkdir -p ~/Vagrant/VM1` .
3. Dans VM1, tapez la commande : `vagrant init ubuntu/xenial64` qui créée un fichier de configuration **Vagrantfile** dans le répertoire, que nous allons inspecter plus tard.
4. Démarrez la VM avec `vagrant up`
5. Logez-vous sur la VM en tapant `vagrant ssh`. Quelles sont les interfaces réseaux de la machine ?
6. Placez-vous dans le répertoire (de la machine physique) où VirtualBox stocke les fichiers de la VM. Quel est le type de disque (en se basant sur l'extension du fichier) ? Cherchez dans `https://www.virtualbox.org/manual/ch05.html` pour comprendre les différentes extensions et notamment celles de VM1 et de la machine Ubuntu créée en section 1.

3 Vagrant Customisation VM et construction d'image

3.1 Customisation de l'instance - Partie 1

Votre fichier Vagrantfile contient surtout des commentaires comme peut vous en convaincre un simple :

```
egrep -v -e"\#|^$" Vagrantfile
```

(elle fait quoi cette jolie commande ? Vous devez être capable de l'expliquer à l'enseignant)

1. Editer le fichier Vagranfile et dé-commenter les lignes en rapport avec vm.provision :

   ```
   config.vm.provision "shell", inline: <<-SHELL
         apt-get update
         apt-get install -y apache2
   SHELL
   ```

   Le "provisioning" aura lieu soit lors du prochain démarrage, soit à chaud. Nous allons prendre la seconde option en tapant la commande :

   ```
   vagrant provision
   ```

2. Prouvez que vous avez bien un serveur Web qui tourne dans la VM en montrant :
   (a) que le service apache2 est actif (quelle commande avez-vous tapé ?).
   (b) que le programme apache2 écoute bien sur le port 80 (utilisez netstat du paquet net-tools)
   (c) en téléchargeant la page d'accueil (comment ? Quelle commande avez-vous tapé ?).
3. Visualiser la page d'accueil depuis le navigateur de la machine physique est impossible. Pourquoi ? (vous pouvez essayer pour vous en convaincre)
4. La configuration réseau est bien compliquée. Analysez le fichier Vagrantfile et montrez qu'il y a 3 solutions (vous devez être capable de dire où elles se situent dans le fichier) pour permettre un accès simple au serveur Apache dans la VM.
5. Implémentez les 3 options simultanément. Il faut redémarrer la VM une fois le fichier Vagrantfile modifié à l'aide de la commande :

   ```
   vagrant reload
   ```

6. Montrez que vous pouvez atteindre le serveur Web au travers de ces 3 options : **il faut retirer le pare-feu de Firefox sinon toutes les requêtes partent vers le proxy**.
7. Quels sont (en quelques mots) les avantages/inconvénients des 3 options d'après vous ?

3.2 Customisation de l'instance - Partie 2

1. Lors du reboot de la machine, Vagrant vous a demandé sur quelle interface physique de la machine doit être fait le pont (eth1 en l'occurence). Pour ne pas avoir à faire cela à chaque reboot, on peut modifier le fichier Vagrantfile en complétant la ligne sur l'interface réseau par le nom **exact** de l'interface. Cela donne (à adapter si besoin)

   ```
   config.vm.network "public_network", bridge: "eth1"
   ```

2. Vagrant utilise les possibilités de répertoire partagé fournies par Virtualbox. Nous allons l'illustrer ici pour permettre de modifier les fichiers du serveur Web de VM1.
   (a) Dans le répertoire où se trouve le fichier Vagrantfile, créez un répertoire vagrantsite et placez dedans le fichier index.html suivant :

   ```
   <html>
     <body>
     <div align="center">Vagrant Machine</div>
           </body>
   </html>
   ```

   (b) Puis éditez le fichier Vagrantfile, et modifiez la partie montage de fichier et la partie liée au montage de fichier ainsi que celle concernant le script de configuration

```
....
config.vm.synced_folder "vagrantsite", "/opt/vagrantsite"
....
config.vm.provision "shell", inline: <<-SHELL
     apt-get update
     apt-get install -y apache2
     ln -s /opt/vagrantsite /var/www/html/vagrantsite
SHELL
```

3. Redémarrez votre instance avec la commande

```
vagrant reload --provision
```

Attention : il faut faire reload pour redémarrer une VM et –provision pour que la partie *provisioning* se fasse. Pourquoi est-ce que la partie *provisioning* est-elle optionnelle et non exécutée à chaque boot ?

4. Montrer que les modifications escomptées ont bien été faites en affichant la page d'accueil du site vagrantsite (`http://yyy/html/vagrantsite/index`)

3.3 Préparation d'une image

Supposons que l'image actuelle nous satisfasse, en terme de configuration. Nous allons en faire un *template*.

1. Faire un template va nous permettre de créer d'autres VMs suivant la même configuration. Cela pose un problème avec l'utilisation de la redirection de port que nous avons utilisée. Pourquoi ?
2. Il faut modifier la directive ainsi :

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

3. Créer la *box* avec la commande :

```
vagrant package
```

Si il y a un problème, il se peut que ce soit à cause du manque des *Guest additions* dans VM1, un complément indispensable pour Vagrant, voir `https://www.vagrantup.com/docs/virtualbox/boxes.html`.

4. Une fois la *box* faite, il faut l'ajouter à votre liste avec la commande

```
vagrant box add --name my-box ./package.box
```

5. Listez vos *boxes* avec :

```
vagrant box list
```

6. Vagrant stocke les images (*boxes*) dans un répertoire particulier, `~/.vagrant.d/boxes`. Faites un `ls -R ~/.vagrant.d/boxes` pour lister ce répertoire. Quelles sont les images ?
7. Est-ce que l'on peut effacer `~/Vagrant/VM1/package.box` (avec une commande rm) ? Dit autrement, est-ce que cela a une incidence sur l'image dans vagrant ?

4 Vagrant *advanced version !*

4.1 Vagrant Hub

Hashicorp fournit un service de partage des images : `https://app.vagrantup.com/boxes/search` que l'on atteint aussi en partant du site principal de vagrant en cliquant sur *Find Boxes* `https://www.vagrantup.com/`.

1. Les images sont associées à un compte utilisateur et ont un nom. Par exemple ubuntu/xenial64. Ubuntu est le nom du compte et xenial64 est le nom de l'image.
   (a) Choisissez l'image ubuntu/xenial64 et cliquez dessus. Cliquez ensuite sur le nom du compte ubuntu pour avoir le profil de l'utilisateur. Est-ce une image en laquelle on peut avoir confiance ?
   (b) Même question pour debian/jessie64
   (c) Même question pour centos/7 (il faut chercher un peu plus...)
2. Ajouter les images debian/jessie64 et centos/7 à votre bibliothèque locale (`vagrant box --help`)

4.2 Autres commandes Vagrant et interaction avec Virtualbox

1. Pour créer l'image à partir de VM1, Vagrant a stoppé VM1. Vous pouvez le vérifier avec :

```
vagrant status
```

   Vous pouvez aussi voir le statut de toutes les VMs actives et pas seulement celle du répertoire courant avec :

```
vagrant global-status
```

2. Redémarrez VM1. (avec quelle commande ?)
3. Vous pouvez faire des *snapshots* (instantanés) de la machine avec (le dernier paramètre est le nom utilisé par Vagrant) :

```
vagrant snapshot save $(date "+%Y%m%d")
```

   Quels nouveaux fichiers sont créés dans le répertoire de VM1 de Virtualbox (contentez vous de les lister) ?
   Note : la commande inverse est **vagrant snapshot restore nom_du_snapshot** et repart dans l'état précédent.
4. Vous pouvez mettre la VM en veille avec

```
vagrant suspend
```

   Vous pouvez vérifiez dans l'interface Virtualbox l'état de la VM.
5. Mettre en veille ou faire un instantané sont des opérations quasiment similaires. Quels fichiers ont été créés par Virtualbox pour cette opération et quelle est la différence avec l'opération snapshot ?
6. Redémarrez la VM avec

```
vagrant resume
```

   Vous pouvez voir les redirections de port avec la commande `vagrant port`. Que voyez-vous ?
7. Vagrant interagit avec l'hyperviseur qui gère les VMs, ici VirtualBox. Souvent, il peut utiliser des fonctions avancées. Dans le fichier Vagrantfile suivant, quelles fonctions avancées sont utilisées et quel est leur intérêt ?

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
VAGRANTFILE_API_VERSION = "2"
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  config.vm.define "web" do |web|
    web.vm.box ="ubuntu/xenial64"
    web.vm.network "forwarded_port", guest:80, host:8888
    web.vm.synced_folder "vagrantsite/", "/opt/vagrantsite", type:"rsync",
    rsync__args: ["--verbose", "--rsync-path='sudo rsync'", "--archive", "--
        delete", "-z"]
    web.vm.provision "shell", inline: "apt-get install -y nginx;
     ln -s /opt/vagrantsite /var/www/html/vagrantsite"
    web.vm.provider "virtualbox" do |v|
      v.linked_clone = true
      v.memory = 1024
      v.cpus = 2
      v.gui = true
    end
 end
end
```

4.3 Vagrant Multi-machines

Soit le Vagrantfile ci-après :

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  # All Vagrant configuration is done here. The most common configuration
  # options are documented and commented below. For a complete reference,
  # please see the online documentation at vagrantup.com.

  # Every Vagrant virtual environment requires a box to build off of.
  config.vm.define "database" do |db|
    db.vm.box = "puppetlabs/ubuntu-14.04-64-nocm"
    db.vm.hostname = "db01"
    db.vm.network "private_network", ip: "192.168.55.100"
  end

  config.vm.define "web" do |web|
    web.vm.box = "puppetlabs/ubuntu-14.04-64-nocm"
    web.vm.hostname = "web01"
    web.vm.network "private_network", ip:"192.168.55.101"
    web.vm.provision "shell",
      inline: "echo '127.0.0.1 localhost web01\n192.168.55.100 db01' > /etc/hosts"
  end

end
```

1. Modifiez le script de manière à ce que ce soit l'image ubuntu/xenial64 qui soit utilisée.
2. Vous pouvez vous logez sur une machine ou sur l'autre avec la commande

```
vagrant ssh nom_VM
```

   Logez vous sur *web* et montrer que l'on peut pinger 192.168.55.100
3. Pourquoi est-ce que

```
ping db01
```

   fonctionne aussi ?
4. Peut-on pinger les 2 VMs depuis la machine physique ?
5. Notez, en remarque finale, que l'on peut être plus générique en créant n machines où n est un nombre fixé par l'administrateur, comme on peut le voir dans le script ci-après. Quelle sera la contrainte physique du serveur à prendre en compte dans le choix de n ?

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# Vagrantfile API/syntax version. Don't touch unless you know what you're
    doing!
VAGRANTFILE_API_VERSION = "2"

# Define a variable - the number of web nodes.
$cluster_nodes = 3
$consul_server_ip = "192.168.30.130"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  # Define a global box file to be used by all machines.
  config.vm.box = "puppetlabs/ubuntu-14.04-64-puppet"

  # Create and provision a Consul server machine.
  config.vm.define "consul" do |consul|
    consul.vm.hostname = "consul"
    consul.vm.network "private_network", ip: $consul_server_ip
    consul.vm.provision "shell", inline: "apt-get update && apt-get install
        unzip"

    consul.vm.provision "puppet" do |puppet|
      puppet.manifests_path = "puppet/manifests"
      puppet.module_path = "puppet/modules"
      puppet.manifest_file = "site.pp"
    end
  end

  # Create a number of cluster members.
  (1..$cluster_nodes).each do |i|
    config.vm.define vm_name = "cluster%02d" % i do |cluster|
      cluster.vm.hostname = vm_name
      cluster.vm.provision "shell", inline: "apt-get update && apt-get
          install -y unzip"
      cluster.vm.provision "puppet" do |puppet|
        puppet.manifests_path = "puppet/manifests"
        puppet.module_path = "puppet/modules"
        puppet.manifest_file = "site.pp"
      end
```

```
      cluster.vm.provision "shell", inline: "consul join #{$consul_server_ip}
          "
    end
  end
end
```

5 Ce que l'on ne couvre pas dans ce TP...et qui est important

— Vagrant est indépendant du *provider*, c'est-à-dire de l'hyperviseur (ex : VirtualBox ou KVM) ou du gestionnaire de cloud (ex : AWS) et il aurait été intéressant de jouer avec un autre *provider*

— La phase de *provisioning* peut-être faite à base de script, mais un intérêt de Vagrant est aussi de pouvoir utiliser des outils modernes de configuration comme Puppet, Chef ou Ansible. Ces outils sont plus puissants et fiables que les scripts.

— Packer `https://www.packer.io/`, toujours de Hashicorp, qui permet de créer à partir d'une ISO, des images pour différents hyperviseurs : VirtualBox, VMware, AWS,..