TRAVAUX DIRIGÉS
PROVISIONNING DE VM - VAGRANT

I. INTRODUCTION

Nous allons voir comment déployer rapidement un environnement de VM afin de pouvoir par la suite créer un cluster pour l'orchestration de conteneurs. Pour cela nous allons utiliser Vagrant (`https://www.vagrantup.com`), un outil développé par la société HASHICORP.

II. ENVIRONNEMENT

Nous allons travailler sur le système Debian des salles du plateau technique. Vous sauvegardez donc tous vos fichiers sur un projet Github afin de pouvoir redéployer rapidement vos infrastructures si nous changeons de salles

Vous allez installer Vagrant en suivant la procédure d'installation décrite sur le site de vagrant (`https://developer.hashicorp.com/vagrant/install`). Vérifier aussi que Virtualbox est bien installé sur la machine, il servira d'hyperviseurs pour nos VM

III. PROVISIONNING DE VOTRE PREMIÈRE VM

Vagrant a un mode de fonctionnement se rapprochant de Docker. En effet, un hub vagrant (`https://portal.cloud.hashicorp.com/vagrant/discover`, héberge un ensemble de VM déjà pré-configurées, répondant à différents besoins.

Showing 25 results

Box name	Latest Version	Downloads	Latest Release	Providers	Architectures
ubuntu/trusty64	20191107.0.0	30,793,012	256 weeks ago	virtualbox	unknown
laravel/homestead	14.0.2	14,596,745	263 days ago	parallels, virtualbox, libvirt, vmware_desktop	amd64, arm64, unknown
hashicorp/precise64	1.1.0	6,815,686	552 weeks ago	virtualbox, vmware_fusion, hyperv	unknown
centos/7	2004.01	6,055,174	229 weeks ago	virtualbox, vmware_desktop, libvirt, hyperv, vmware_fusion, vmware_workstation, vmware	unknown
ubuntu/xenial64	20211001.0.4	3,629,614	20 days ago	virtualbox	unknown

Pour instancier notre première VM, nous allons créer un répertoire pour stocker le fichier *Vagrantfile*, coeur de l'utilisation de Vagrant, puis utiliser la commande **vagrant init bento/debian-12.6 –box-version 202407.22.0**. Cette commande va créer un fichier *Vagrantfile* dans le répertoire. Pour démarrer cette VM, il faut lancer la commande **vagrant up**.

Remarques : La VM est lancée dans le principe d'un serveur, c'est à dire pour être utilisée à distance. Le mode de connexion simplifié utilisé par Vagrant est un utilisateur ssh *vagrant* avec une clé privée. Pour s'y connecter simplement, il faut utiliser la commande **vagrant ssh** qui peut être suivi du nom de la VM si nécessaire

Pour arrêter la VM, il faut utiliser la commande **vagrant halt**

III.1. **exercice :** Faite de même en sélectionnant une autre box (vous trouverez . Vous remarquerez que la commande **vagrant init** remplace le fichier *Vagrantfile*. En effet, un seul fichier Vagrantfile par répertoire, par contre il pourra comporter plusieurs VM à instancier.

IV. commandes de base Vagrant

Les commandes de bases de Vagrant permettant de manipuler les box et les VM sont :

(1) création d'une VM
- **vargant init** : permet de créer un fichier Vagrantfile vide et le sous répertoire .vagrant
- **vargant init path/box** : permet de créer un fichier Vagrantfile permettant d'utilser la box précisée et le sous répertoire .vagrant. La box fait partie du catalogue public de box vagrant

(2) démarrer une VM
- **vagrant up (#nom de la VM)** : démarre la ou les VM précisées dans le Vagrantfile, en forçant le provisionnement.
- **vagrant reload (#nom de la VM)** : redémarre la VM, charge une nouvelle configuration de Vagrantfile
- **vagrant reload –provision (#nom de la VM)** : redémarre la VM et force le provisionnement

(3) connexion à une VM
- **vagrant ssh (#nom de la VM)** : permet de se connecter sur un terminal de la VM en ssh. par défaut le mode ssh direct est interdit sur les box vagrant. Les box n'étant pas sécurisées, vagrant sert essentiellement dans le cadre de banc de test.

(4) arrêter une VM
- **vagrant halt (# nom de la VM)** : permet d'arrêter la VM.
- **vagrant suspend (#nom de la VM)** : suspend l'exécution d'une VM

(5) détruire une VM
- **vagrant destroy (#nom de la VM)** : arrête et supprime la VM

(6) gestion des VM
- **vagrant global-status** : permet d'obtenir la liste des VM disponibles, avec leurs informations

```
[→ vagrant global-status
id       name    provider       state       directory
--------------------------------------------------------------------------------
8682225  master  vmware_desktop not running /Users/nono/VM
ef44524  slave   vmware_desktop not running /Users/nono/VM
759b7b4  default vmware_desktop running     /Users/nono/VM/VM2

The above shows information about all known Vagrant environments
on this machine. This data is cached and may not be completely
up-to-date (use "vagrant global-status --prune" to prune invalid
entries). To interact with any of the machines, you can go to that
directory and run Vagrant, or you can use the ID directly with
Vagrant commands from any directory. For example:
"vagrant destroy 1a2b3c4d"
```

(7) Gestion des box la gestion des box passe par la commande **vagrant box** suivi de la commande spécifique à exécuter. En cela, elle fonctionne sur le même mode que la gestion des images docker (docker image #commmande)
- **vagrant box list** : donne la liste des box disponibles dans la système
- **vagrant box add #path/box** : télécharge une box en local
- **vagrant box remove #box** : supprime une box. Nécessite alors de la télécharger à nouveau pour la réutiliser pour une VM

V. configuration Vagrant

Les fichiers *Vagrantfile*, sont écrits en Ruby. Ils présentent une structure en boucle imbriquées. La première brique est

```
1 Vagrant.configure("2") do |config|
2 ...
3 end
```

la première ligne permet de préciser la version de vagrant utilisé, à savoir la version 2. Le mot *config* entre pipe, est une variable utilisée dans le reste de la boucle initiée par *do* et fermée par *end*. Différentes section de configuration sont :

— **config.vm** : permet de paramétrer la VM
— **config.ssh** : permet de définir comment vagrant accéde à la VM
— **config.vagrant** : permet de configurer l'application vagrant

```
Vagrant.configure("2") do |config|
   config.vm.define "master" do |master|
       ...
   end
end
```

Cette second brique permet de définir une nouvelle vm nommée master et de démarrer la boucle de paramétrage de cette VM.

comme paramètre de la VM nous allons définir la box utilisée. On peux aussi définir la version de box, le hostname, le réseau défini, et enfin, nous allons configurer le provider de VM. Vagrant peut fonctionner avec de nombreux hyperviseurs tel de virtualbox, vmware_desktop, libvirt, ... Nous allons définir une nouvelle boucle sur la *config.vm.provider* pour le paramétrer

```
Vagrant.configure("2") do |config|
    config.vm.define "master" do |master|
        master.vm.box = "spox/ubuntu-arm"
        master.vm.box_version = "1.0.0"
        master.vm.hostname = 'master'
        master.vm.provision "docker"

        master.vm.network :private_network, ip: "192.168.56.101"

        master.vm.provider "vmware_desktop" do |v|
            v.vmx["memsize"] = "2048"
            v.vmx["numvcpus"] = "2"
      end
    end
end
```

Les commandes ici sont liées au provider vmware (vmx). Il existe des commandes spécifiques pour virtualbox

— v.customize ["modifyvm", :id, "–memory", 2048] : les options possibles autres que memory sont entre autres, name, cpus, natdnshostresolver1, ...
— v.name = "nom" : écriture raccourcie pour le nom de la VM
— v.memory = 2048 :écriture raccourcie pour la mémoire
— v.cpus = 2 : écriture raccourcie pour le nombre de cpus

Enfin, le provisionnement permet d'exécuter un ensemble de commande pour configurer le système virtualisé. La config est fournie à l'aide de la commande **master.vm.provision #(mode de provision), paramètres** Les modes de provisions sont :

— **shell** : qui admet 2 façons de fournir le script,
  (1) Directement dans un bloc inline : «-SHELL terminé par SHELL,
  (2) Par la fourniture d'un fichier de script avec "path : nom du fichier". Le fichier est hébergé dans le répertoire du Vagrantfile
— **ansible_local** : Démarre alors une boucle pour affecter un playbook ansible

VI. EXERCICES

VI.1. **Exercice 1.** Créer une VM simple à partir d'une image ubuntu. Démarrer la VM et connecter vous à l'aide de SSH.

VI.2. **Exercice 2.** Créer une VM apache. Vous paramètrerez le réseau pour avoir une adresse privée, permettant uniquement à l'hôte d'accéder à la page web initiale. Vous démarrerez le service et testerez l'accès à site web depuis l'hôte. Faite ensuite de même en modifiant les paramètres réseaux pour faire de la translation de port en publiant le port 80 de la VM sur le port 15234 de l'hôte.

VII. CONFIGURATION DE MULTIPLES VM

Il est intéressant de pouvoir démarrer plusieurs VM à partir d'un même fichier Vagrantfile afin de concevoir un cluster de VM, disposant du même sous-réseau et configurée ainsi à la volée. Nous nous servirons de cette fonctionnalité pour créer des clusters swarm ou kubernetes.

Il est donc possible d'avoir plusieurs instructions vm.config.define permettant de démarrer la configuration d'un VM. il faut donc dupliquer les lignes de configuration pour chaque VM, ce qui permet de configurer de façon fine. Maintenant, nous pouvons aussi vouloir plusieurs VM avec la même configuration. Dans ce cadre, nous pouvons utiliser les boucles Ruby pour faire cela.

```
Vagrant.configure("2") do |config|
vmnumber = 3 #variable du nombre de VM
    (1..vmnumber) for each do | i |
      config.vm.define "master#{i}" do |master|
            master.vm.box = "spox/ubuntu-arm"
            master.vm.box_version = "1.0.0"
            master.vm.hostname = "master#{i}"
    master.vm.name = "master#{i}"

            master.vm.network :private_network, ip: "192.168.56.10#{i}" #permet de
    configurer les adresses 192.168.56.101 a 103
          end
    end
end
```

VIII. EXERCICES

VIII.1. **exercice 1.** Créer un Vagrantfile permettant de déployer 2 VM utilisant 2 box différentes. Dans chacune d'elle, procéder à l'installation de la commande curl.

VIII.2. **exercice 2.**

(1) déployer 3 vm identiques, mettant en oeuvre un serveur nginx. Penser à la définition des adresses dans un sous réseau privé en 172.168.2.21 à 172.68.2.23.

(2) faite de même mais en utilisant la translation de port pour exposer le port 80 est VM sur les port 8081 à 8083 en fonction de la VM

pour chaque configuration, vous testerez que la page d'accueil nginx est visible.