**TP Vagrant**

**I. Introduction à vagrant :**

Vagrant est un outil open source pour travailler avec des environnements virtuels. Il agit comme un wrapper et se trouve au-dessus de la solution de virtualisation (Vmware, Hyper-V, VirtualBox, libvirt) et fournit des CLI (command line interface) simples pour gérer la MV avec des configurations personnalisées.

Avantages :

* Pas besoin d'apprendre les commandes CLI des fournisseurs de virtualisation (VirtualBox, VMware, etc…). Vagrant prend en charge la gestion des MV sous-jacentes avec son interface CLI simple.
* La configuration utilisée pour créer l'environnement est décrite dans un simple fichier texte. L'utilisateur peut créer autant d'instances de Vagrant en utilisant ce même fichier.
* Vagrant est capable d'exécuter un logiciel de gestion de configuration comme Puppet/ Ansible/ Chef une fois que le système de base est prêt. Cela aide à configurer l'environnement (OS + App) sur la MV cible de manière automatisée.
* Le Développeur peut créer/détruire plusieurs environnements de développement en quelques minutes.

Fonctionnement :

* Crée une VM à partir d’une image de base.
* Configure la MV à l’aide du fichier Vagrantfile.
* Automatise la gestion de configuration des logiciels.

Terminologie:

* Box: machine virtuelle (MV)
* Provider: fournit le support de virtualisation :
  + Local: VirtualBox, Vmware, Docker, Hyper-V
  + Remote: AWS Cloud, OpenStack
* Vagrantfile: fichier de configuration qui spécifie;
  + Quelle image de base à utiliser.
  + Quel sera le nom de l’hote.
  + Quel sera le réseau.
  + Provisioner config (shell/outil automatique)
* Provisioner: un outil pour installer l’environnement virtuel. Il peut être un simple script shell, ou un outil automatique (chef, Ansible, etc...).

II. Prérequis :

* Installation de VirtualBox à partir de <http://www.virtualbox.org/>. Ce guide utilise la version 5.2 de VirtualBox.
* Installation de Vagrant à partir de<http://www.vagrantup.com/>. Ce guide utilise la version 2.2.6.

**Remarque :** sous Windows la version 2.2.6 de Vagrant exige la version 3 ou plus de PowerShell.

III. Expérimentation de vagrant :

Pour exécuter les commandes CLI de vagrant, lancer un ***terminal*** sous Linux, et ***windows power shell*** sous windows.

|  |  |
| --- | --- |
| **Linux** | **Windows** |
| Lancer un terminal | Lancer PowerShell  Démarrer->Accessoires->Windows PowerShell  (exécuter en tant qu’un administrateur) |

1) Vérification de l’installation de vagrant :

Taper la commande vagrant –v et vérifier la version de vagrant que vous avez installé.



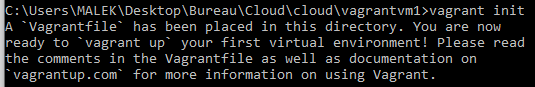
2) Création d’une machine virtuelle (Box):

- Créer un répertoire de travail (par exemple vagrantvm1)

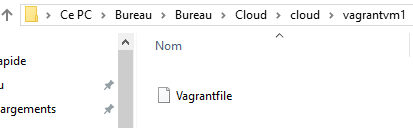


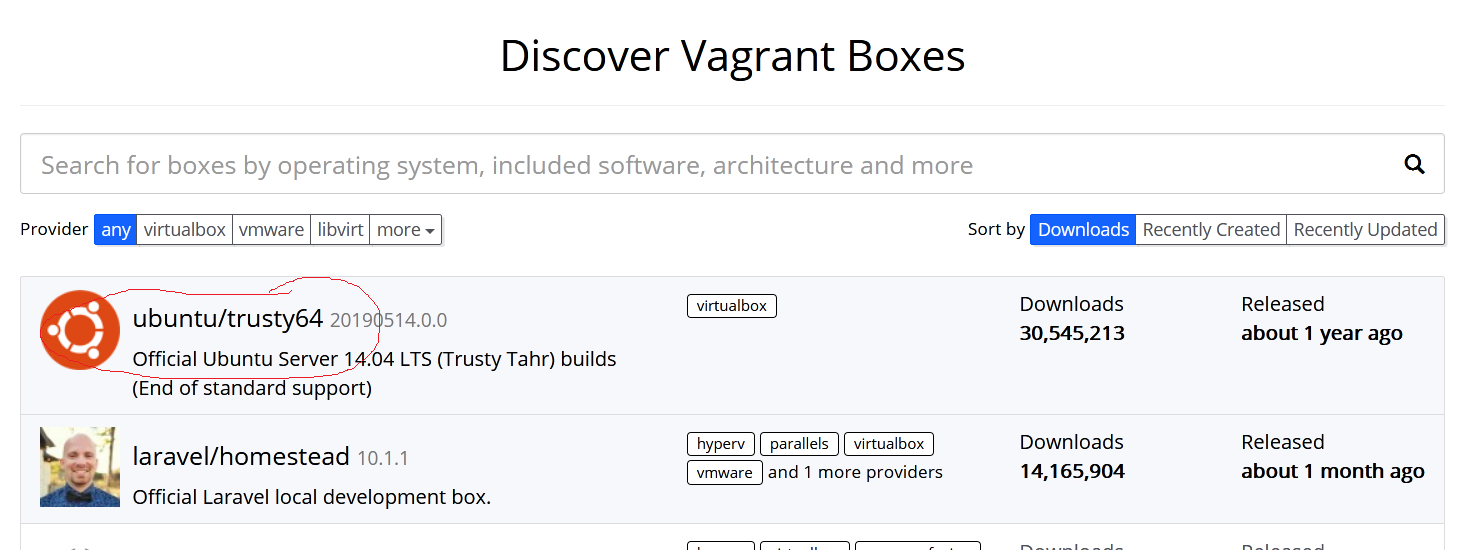
- Consulter le site <https://app.vagrantup.com/boxes/search> : ce site contient des boxes vagrant (image d’une machine virtuelle préinstallée)

- Accéder au dossier vagrantvm1 et taper la commande vagrant init



- Après l’étape d’initialisation, le fichier Vagrantfile est créé



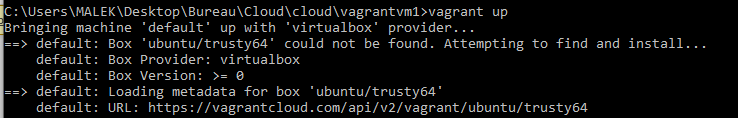


On va télécharger le box ubuntu/trusty64 qui est un box blanc (contient seulement le OS).

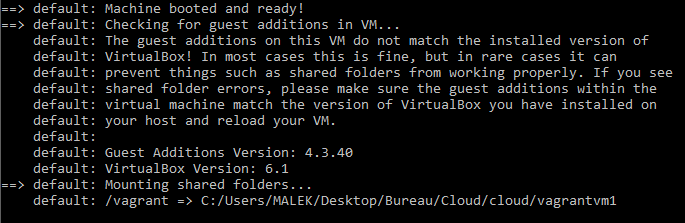
* Vérifier votre répertoire vagrantvm1, vous devez trouver un fichier vagrantfile. Examiner ce fichier. Ce fichier est le programme qui contient les instructions pour créer et définir la configuration de la MV. Si on ignore les commentaires le fichier contient le code suivant :

|  |
| --- |
| Vagrant.configure("2") do |config|  config.vm.box = "ubuntu/trusty64"  end |

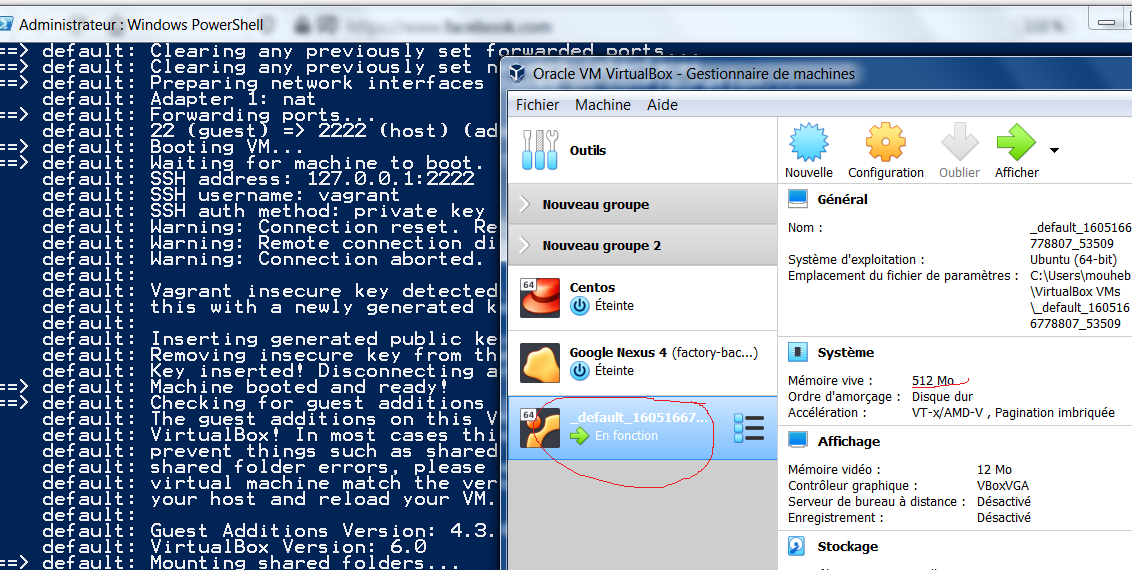
* Pour créer la MV taper la commande vagrant up



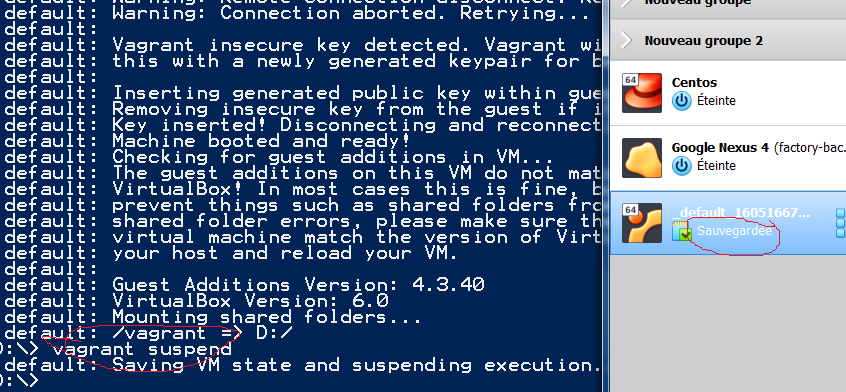
A la fin vous trouvezles lignes suivantes :



* Lancer l’application virtualBox et vérifier que la MV est en fonction :



* Pour arrêter la MV taper la commande vagrant suspend (avec sauvegarde de son état) ou vagrant halt (sans sauvegarde de son état). La commande vagrant resume permet de restaurer la MV sauvegardé.



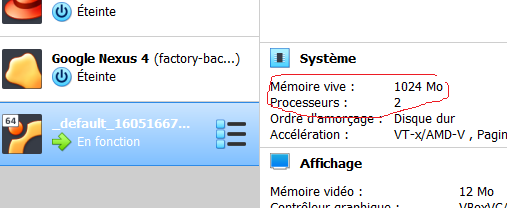
* Pour détruire la MV utiliser la commande vagrant destroy.
* Pour voir la liste des boxes téléchargés : vagrant box list

2) Modification des paramètres de la MV.

Pour changer la taille de la mémoire et le nombre de CPU il faut modifier le fichier vagrantfile en ajoutant le bloc suivant :

|  |
| --- |
| config.vm.provider "virtualbox" do |vb|  vb.memory = 1024  vb.cpus=2  end |

* Taper la commande vagrant reload pour tenir en compte de la nouvelle configuration.



3) Se Connecter à la MV :

Pour se connecter à la machine virtuelle on utilise le protocole ssh :

* Taper la commande vagrant ssh

4) Port forwarding :

On veut installer un serveur web sur la machine virtuelle qui sera accessible à partir du monde extérieur.

Sur la MV :

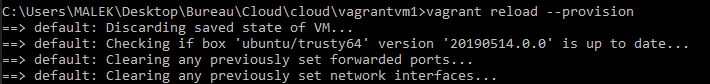
* Faire un update : sudo apt-get update
* Installer le serveur apache: sudo apt-get install apache2

Sur la machine hôte:

* Modifier le fichier vagrantfile en ajoutant la ligne suivante:

|  |
| --- |
| config.vm.network "forwarded\_port", guest: 80, host: 8080 |

* Se déconnecter de la MV en tapant la commande exit
* Taper la commande : vagrant reload (pour tenir en compte de la nouvelle config)



* Lancer le navigateur et taper dans la barre d’adresse : 127.0.0.1 :8080. Vous êtes connectés au serveur web de la machine virtuelle.

**5) Réseau privé :**

On peut définir une adresse IP privée à notre MV. Pour comprendre ça on va accéder au serveur web de la section précédente à partir de l’adresse IP de la MV et non pas en utilisant port forwarding.

* Dans le fichier vagrantfile, supprimer la ligne qu’on a ajouté dans la section précédente, et ajouter la ligne

|  |
| --- |
| config.vm.network "private\_network", ip: "192.168.33.10" |

* Taper la commande : vagrant reload
* Lancer le navigateur sur la machine hôte et taper dans la barre d’adresse l’adresse ip de la MV : 192.168.33.10

6) Synchronisation des dossiers :

La page index.html que vous afficher sur le navigateur est sauvegardé dans le répertoire /var/www/html de la machine virtuelle. Donc chaque fois on veut modifier ce fichier, on doit se connecter sur la machine invitée. La synchronisation des dossiers permet d’éviter ce problème en partageant un répertoire de la machine hôte avec la MV.

* Dans le fichier vagrantfile, ajouter la ligne suivante :

|  |
| --- |
| config.vm.synced\_folder ".", "/var/www/html" |

Le premier répertoire “.” (répertoire courant) est le répertoire de la machine hôte qui sera accessible à partir de dur repertoire "/var/www/html" de la MV.

* Taper la commande : vagrant reload
* Accéder au repertoire /var/www/html  de la MV et taper la commande ls. Que remarquez-vous ?
* Changer la page d’accueil : <p>Bonjour tout le monde</p>

7) Provision :

- Dans le fichier vagrantfile ajouter le bloc suivant :

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

Ceci veut dire que lorsqu’on recrée le box ces provisions (update et installation d’apache) seront exécutées automatiquement.

* Exécuter la commande vagrant provision et observer le terminal.

IV Activité :

Créer une MV avec les paramètres suivants :

* Image : ubuntu/xenial64
* RAM : 1GO
* Processeur : 1
* Provision : apache, mysql, php (installer automatiquement apache, mysql et php)

**Compte rendu :** le compte rendu doit inclure des captures d’écran et le fichier vagrantfile.

NB : L’évaluation tient en compte du temps de réalisation du travail demandé.

**8) Environnement multi machines :**

La principale raison pour laquelle nous souhaitons créer des réseaux de machines Vagrant est souvent parce que nous souhaitons modéliser un environnement de plusieurs machines. Un exemple courant pourrait être le désir de modéliser une application Web avec des serveurs Web et une base de données dédiés machines, ou même un environnement qui crée un cluster de machines virtuelles identiques.

1. Créer un nouveau répertoire.
2. Commencez avec un simple Vagrantfile sans définition de machine ou de boîte:

|  |
| --- |
| # -\*- mode: ruby -\*- # vi: set ft=ruby : VAGRANTFILE\_API\_VERSION = "2" Vagrant.configure(VAGRANTFILE\_API\_VERSION) do |config| end |

1. Dans le | config | , définissez un serveur de base de données à l'aide d'une définition de machine Vagrant. . Dans un environnement multimachine, nous voulons généralement  
   pour définir une adresse IP de machine (en particulier dans le cas où une machine Vagrant (un serveur web) devra se connecter à une autre (un serveur de base de données).

|  |
| --- |
| config.vm.define "database" do |db| db.vm.box = " ubuntu/trusty64" db.vm.hostname = "db01" db.vm.network "private\_network", ip: "192.168.55.100" end |

1. Créez une deuxième machine définie dans un bloc après le bloc de| db |. Ce sera la machine Web:

|  |
| --- |
| config.vm.define "web" do |web| web.vm.box = " ubuntu/trusty64" web.vm.hostname = "web01" web.vm.network "private\_network", ip:"192.168.55.101" end |

1. Démarrez les machines
2. Pinger sur les deux machines.
3. Taper la commande vagrant status.
4. Connecter-vous sur web et pinger sur la machine db.
5. Nous utiliserons le provisioner pour définir le fichier /etc/hosts sur le serveur Web. Cela permet au serveur Web de faire référence au serveur de base de données avec le nom d'hôte db01. La définition complète du serveur Web comprendra cette commande d'approvisionnement.

|  |
| --- |
| config.vm.define "web" do |web| web.vm.box = " ubuntu/trusty64" 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 |

1. Démarrez les machines
2. Accédez à la machine Web en utilisant la commande vagrant ssh.
3. Vérifier que le serveur web peut contacter la machine database avec son nom db01 : ping db01.