Vagrant - Créateur de VM

Benoit Métrot
benoit.metrot@math.univ-poitiers.fr

UMR 7348 - Laboratoire de Mathématiques et Applications (Poitiers)

Journée de veille technologique ARGOS
Orsay, Décembre 2014

Plan

Progression

Le contexte

- Organisation d'une action nationale de formation
  - Titre : *Les systèmes d'authentification dans la communauté enseignement supérieur et recherche : étude, mise en œuvre et interfaçage dans un laboratoire de mathématique*
  - Partenaires : CNRS, INSMI, GDS Mathrice
  - 50 participants
  - Du 22 au 26 septembre 2014 à Angers

- Besoin d'organiser des TPs
  - Connexion réseau avec un débit relativement faible
  - Pas de salle machine à disposition
  - Diversité des OS installés sur les ordinateurs portables des participants

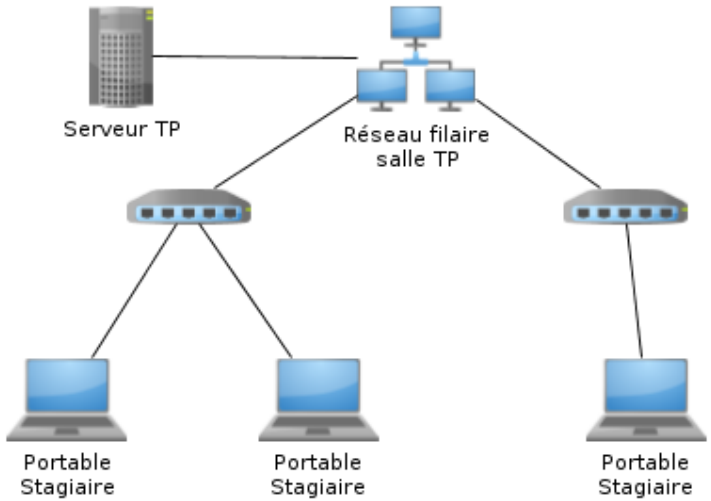
- → `http://anf2014.mathrice.fr`

Mathrice

- Un réseau de métier pour les informaticiens des laboratoires de Mathématiques
  - Echange, entraide
  - Formation
  - Projets
- Des services en ligne à destination des chercheurs de la communauté Mathématique - Plateforme en Ligne Mathrice (PLM)
  - Consultation de revues en ligne
  - Messagerie électronique, Listes de diffusion
  - Agendas partagés
  - Disque internet (PLMbox)
  - Webconférence
  - ...
- Un Groupement De Service (GDS) du CNRS (GDS 2754)
- → `http://www.mathrice.fr`

Organisation retenue

- Les participants apportent leur ordinateur portable
- Construction d'un réseau filaire au sein de la salle de formation
- Intégration d'un serveur
  - DHCP
  - DNS
  - Mirroir Debian local
  - Supports de cours et de TP (HTTP)
- Distribution de machines virtuelles aux participants



Progression

Fonctionnalités

Vagrant est un outil de construction d'environnements de développement

- Création de machines virtuelles à partir de modèle (*Box*)
- Etablit les liens réseau entre les machines (réseau privé) et avec le monde extérieur (Internet)
- Définition des noms d'hote et des adresses réseaux des machines virtuelles
- Mise en place d'un dossier partagé entre la machine hôte et les machines virtuelles
- Personnalisation des machines virtuelles à partir d'un script, d'un programme *Puppet* ou *Chef* (*Provisioning*)

Historique

- Projet initié en janvier 2012 par Mitchell Hashimoto
- Sortie de la version 1.0 en mars 2012
- Création de HashiCorp en novembre 2012
- Introduction du nouveau format de configuration en mars 2013 (v1.1+)
- Publication du livre *Vagrant : UP and Running* chez *O'Reilly* en juin 2013
- Sortie de la version 1.6.5 en mai 2014

Installation

- S'installe sur :

- Pour construire des machines virtuelles :

- En s'appuyant sur :

Pourquoi virtualiser ?

- Séparer les différents projets en cours
- Homogéniser les environnements de travail des développeurs d'un projet
- Un environnement de développement n'est pas un environnement de production
- Indépendance de l'OS hôte de la machine
- Bugs *Works on my machine*

Pourquoi automatiser ?

- Avec une installation manuelle
  $\rightarrow$ risque d'oubli d'un composant
  $\rightarrow$ configuration pas forcément identique sur N machines
  $\rightarrow$ long et fastidieux
- Passage à l'échelle
- Accélérer le processus de mise en oeuvre
- Portage vers un environnement de production

Principe de fonctionnement

- Abstraction du système de virtualisation
- Un seul fichier de configuration (*Vagrantfile*)
- Introduction d'une unique commande vagrant pour tout piloter
  - Création
  - Allumage
  - Mise en veille
  - Extinction
  - Ouverture de session
- Utilise des protocoles et outils existants (SSH, RDP...)

Progression

Vos machines en 3 étapes

1. Installer Vagrant

2. Création d'un unique fichier pour décrire
   - Les machines à créer
   - Les liens réseau
   - Les logiciels à installer
   - Les moyens par lesquels accéder à la machine

3. La commande *vagrant up* se charge du reste

Vos machines en 3 étapes

1. Installer Vagrant

2. Création d'un unique fichier pour décrire
   - Les machines à créer
   - Les liens réseau
   - Les logiciels à installer
   - Les moyens par lesquels accéder à la machine

3. La commande *vagrant up* se charge du reste

Vos machines en 3 étapes

1. Installer Vagrant

2. Création d'un unique fichier pour décrire
   - Les machines à créer
   - Les liens réseau
   - Les logiciels à installer
   - Les moyens par lesquels accéder à la machine

3. La commande *vagrant up* se charge du reste

Commande *vagrant*

vagrant up	Construit et démarre les machines définies dans le fichier de configuration *Vagrantfile*
vagrant suspend	Place en pause l'ensemble des VM
vagrant resume	Réveille les machines virtuelles qui ont été placées en pause avec *vagrant suspend*
vagrant halt [-f]	Arrête (éteint) les machines virtuelles via un signal ACPI (arrêt logiciel). L'option *-f* permet de forcer l'arrêt
vagrant status	Affiche l'état des machines virtuelles

Toutes ces commandes peuvent être suffixées par le nom d'une machine virtuelle. Dans ce cas l'action s'applique uniquement à la machine spécifiée.

Commande *vagrant*

vagrant up	Construit et démarre les machines définies dans le fichier de configuration *Vagrantfile*
vagrant suspend	Place en pause l'ensemble des VM
vagrant resume	Réveille les machines virtuelles qui ont été placées en pause avec *vagrant suspend*
vagrant halt [-f]	Arrête (éteint) les machines virtuelles via un signal ACPI (arrêt logiciel). L'option *-f* permet de forcer l'arrêt
vagrant status	Affiche l'état des machines virtuelles

Toutes ces commandes peuvent être suffixées par le nom d'une machine virtuelle. Dans ce cas l'action s'applique uniquement à la machine spécifiée.

Accès aux machines

1. Avec SSH
   - Commande : *vagrant ssh <mavm>*
   - Ouvre un shell SSH sur la machine *mavm* avec l'utilisateur *vagrant*
   - Nécessite d'avoir un client SSH définit dans le PATH et un service SSH fonctionnel sur la VM
2. Avec RDP
   - Commande : *vagrant rdp <mavm>*
   - Ouvre une session bureau à distance sur la machine *mavm* avec l'utilisateur *vagrant*
   - Nécessite de disposer d'un client bureau à distance sur la machine hote et un service RDeskop sur la VM
3. Dossier partagé
   - Le répertoire courant (d'où sont lancés les commandes vagrant) sur la machine hôte est partagé avec le */vagrant* de chaque VM
   - Exemple : */home/user* $\rightarrow$ */vagrant*

Fichier de configuration : Vagrantfile

```
# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
VAGRANTFILE_API_VERSION = "2"
box_repo = "file:///home/myname/Boxes/"

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|

  config.vm.define "linux1" do |linux1|
    linux1.vm.box = "debian7"
    linux1.vm.box_url = "#{box_repo}/debian_wheezy.box"
    linux1.vm.network "private_network", ip: "172.16.20.10"
    linux1.vm.network "public_network"
    linux1.vm.provision "shell", path: ".vagrant.d/provisionning/apache.sh"
  end

  config.vm.define "linux2", autostart: false do |linux2|
    linux2.vm.box = "debian7"
    linux2.vm.box_url = "#{box_repo}/debian_wheezy.box"
    linux2.vm.network "private_network", ip: "172.16.20.20"
    linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
  end

  config.vm.define "windows7", autostart: false do |windows7|
    windows7.vm.box = "win7"
    windows7.vm.box_url = "#{box_repo}/windows_seven.box"
    windows7.vm.network "private_network", ip: "172.16.20.30"
    windows7.vm.communicator = "winrm"
    windows7.vm.network "forwarded_port", guest: 3389, host_ip: "127.0.0.1",
                        host: 8902, protocol: "tcp"
  end
end
```

Vagantfile : syntaxe globale

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

→ Définition de la version du fichier de configuration

```
box_repo = "file:///home/myname/Boxes/"
```

→ Définition d'une variable, c'est du Ruby !

```
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
```

→ Début de la section de configuration

```
config.vm.define "linux2", autostart: false do |linux2|
  linux2.vm.box = "debian7"
  linux2.vm.box_url = "#{box_repo}/debian_wheezy.box"
  linux2.vm.network "private_network", ip: "172.16.20.20"
  linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
end
```

→ Un bloc par machine virtuelle à construire

Vagantfile : syntaxe globale

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

→ Définition de la version du fichier de configuration

```
box_repo = "file:///home/myname/Boxes/"
```

→ Définition d'une variable, c'est du Ruby !

```
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
```

→ Début de la section de configuration

```
config.vm.define "linux2", autostart: false do |linux2|
  linux2.vm.box = "debian7"
  linux2.vm.box_url = "#{box_repo}/debian_wheezy.box"
  linux2.vm.network "private_network", ip: "172.16.20.20"
  linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
end
```

→ Un bloc par machine virtuelle à construire

Vagantfile : syntaxe globale

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

→ Définition de la version du fichier de configuration

```
box_repo = "file:///home/myname/Boxes/"
```

→ Définition d'une variable, c'est du Ruby !

```
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
```

→ Début de la section de configuration

```
config.vm.define "linux2", autostart: false do |linux2|
  linux2.vm.box = "debian7"
  linux2.vm.box_url = "#{box_repo}/debian_wheezy.box"
  linux2.vm.network "private_network", ip: "172.16.20.20"
  linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
end
```

→ Un bloc par machine virtuelle à construire

Vagantfile : syntaxe globale

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

→ Définition de la version du fichier de configuration

```
box_repo = "file:///home/myname/Boxes/"
```

→ Définition d'une variable, c'est du Ruby !

```
Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
```

→ Début de la section de configuration

```
config.vm.define "linux2", autostart: false do |linux2|
  linux2.vm.box = "debian7"
  linux2.vm.box_url = "#{box_repo}/debian_wheezy.box"
  linux2.vm.network "private_network", ip: "172.16.20.20"
  linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
end
```

→ Un bloc par machine virtuelle à construire

Vagantfile : définition d'une machine (1)

```
linux1.vm.box = "debian7"
```

→ Le nom de la machine virtuelle

```
linux1.vm.box_url = "#{box_repo}/debian_wheezy.box"
```

→ Emplacement où prendre le modèle (*box* de machine à utiliser pour la création (URL possible))

```
linux1.vm.network "public_network"
```

→ Ajout d'une interface réseau avec pont sur l'interface physique de la machine hôte

```
linux1.vm.network "private_network", ip: "172.16.20.10"
```

→ Réseau privé entre les machines virtuelles

Vagantfile : définition d'une machine (1)

```
linux1.vm.box = "debian7"
```

→ Le nom de la machine virtuelle

```
linux1.vm.box_url = "#{box_repo}/debian_wheezy.box"
```

→ Emplacement où prendre le modèle (*box* de machine à utiliser pour la création (URL possible))

```
linux1.vm.network "public_network"
```

→ Ajout d'une interface réseau avec pont sur l'interface physique de la machine hôte

```
linux1.vm.network "private_network", ip: "172.16.20.10"
```

→ Réseau privé entre les machines virtuelles

Vagantfile : définition d'une machine (1)

```
linux1.vm.box = "debian7"
```

→ Le nom de la machine virtuelle

```
linux1.vm.box_url = "#{box_repo}/debian_wheezy.box"
```

→ Emplacement où prendre le modèle (*box* de machine à utiliser pour la création (URL possible))

```
linux1.vm.network "public_network"
```

→ Ajout d'une interface réseau avec pont sur l'interface physique de la machine hôte

```
linux1.vm.network "private_network", ip: "172.16.20.10"
```

→ Réseau privé entre les machines virtuelles

Vagantfile : définition d'une machine (1)

```
linux1.vm.box = "debian7"
```

→ Le nom de la machine virtuelle

```
linux1.vm.box_url = "#{box_repo}/debian_wheezy.box"
```

→ Emplacement où prendre le modèle (*box* de machine à utiliser pour la création (URL possible))

```
linux1.vm.network "public_network"
```

→ Ajout d'une interface réseau avec pont sur l'interface physique de la machine hôte

```
linux1.vm.network "private_network", ip: "172.16.20.10"
```

→ Réseau privé entre les machines virtuelles

Vagantfile : définition d'une machine (2)

```
linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
```

→ Script de personnalisation de la machine virtuelle

```
windows7.vm.network "forwarded_port", guest: 3389, host_ip: "127.0.0.1",
                    host: 8902, protocol: "tcp"
```

→ Redirection de port entre VM et machine hote

```
autostart: false
```

→ Pour ne pas créer/allumer la machine lors d'un *vagrant up*

Vagantfile : définition d'une machine (2)

```
linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
```

→ Script de personnalisation de la machine virtuelle

```
windows7.vm.network "forwarded_port", guest: 3389, host_ip: "127.0.0.1",
                    host: 8902, protocol: "tcp"
```

→ Redirection de port entre VM et machine hote

```
autostart: false
```

→ Pour ne pas créer/allumer la machine lors d'un *vagrant up*

Vagantfile : définition d'une machine (2)

```
linux2.vm.provision "shell", path: ".vagrant.d/provisionning/desktop.sh"
```

→ Script de personnalisation de la machine virtuelle

```
windows7.vm.network "forwarded_port", guest: 3389, host_ip: "127.0.0.1",
                    host: 8902, protocol: "tcp"
```

→ Redirection de port entre VM et machine hote

```
autostart: false
```

→ Pour ne pas créer/allumer la machine lors d'un *vagrant up*

Progression

Où trouver des *boxes* ?

- `https://vagrantcloud.com/discover/featured`
- `http://www.vagrantbox.es/`
- `http://puppet-vagrant-boxes.puppetlabs.com/`

Créer sa boîte

- Créer une nouvelle machine dans VirtualBox
- Paramètrages spécifiques (Linux)
  - Création d'un utilisateur *vagrant* avec un mot de passe *vagrant*
  - Autorisation de l'utilisateur *vagrant* dans sudo
  - Ajout des outils VirtualBox
  - Mise en oeuvre du service SSH ou RDP
  - Nettoyage du système (fichiers temporaires, caches)
- *vagrant package –base <mavm-vbox>*
- Quelques recettes :
  - Debian 7
    `http://anf2014.mathrice.fr/box-wheezy.html`
  - Fedora 20
    `http://anf2014.mathrice.fr/box-fedora20.html`
  - Windows 7
    `http://anf2014.mathrice.fr/box-windows7.html`
  - Windows Serveur 2012 R2
    `http://anf2014.mathrice.fr/box-windows2012r2.html`

Providers

Définition

Le *Provider* est un composant logiciel inclus dans Vagrant qui a en charge l'interface entre Vagrant et le système de virtualisation sous-jacent.

D'origine Vagrant est fournit avec le *Provider* pour VirtualBox, mais d'autres sont disponibles

- VMWare
- Hyper-V
- Docker

Progression

Conclusion

- Utilisation facile
- Rapidité de déploiement et duplication d'un environnement
- Accès automatique des VMs à Internet
- Idéal pour la construction :
  - de maquettes
  - de plateforme de tests
  - d'environnements de développement

- Attention aux licences des systèmes virtualisés
- Dans le provisionnement définir un maximum de choses statiquement
- Factoriser les scripts de *provisioning*

Vos questions...

A vous de jouer...