

---

---

# Vagrant



Thierry Vaira  
LaSalle Avignon BTS SN IR

v0.1 22/05/2020

---



---

# Présentation

**Vagrant** est un logiciel libre et open-source pour la **création et la configuration des environnements de développement virtuel**.

Il peut être considéré comme un *wrapper* autour de logiciels de virtualisation comme VirtualBox ou VMware.

Vagrant s'utilise via une interface en **ligne de commande** (CLI).

<https://www.vagrantup.com/>

---

---

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

---

# Vagrantfile

**Vagrantfile** est un fichier texte qui permet de décrire le type de machine requis pour un projet, et comment le configurer.

Un fichier **Vagrantfile** par projet.

La syntaxe des fichiers **Vagrantfile** est **Ruby** (la connaissance de Ruby n'est pas vraiment nécessaire).

Les fichiers **Vagrantfile** sont portables sur toutes les plateformes prises en charge par Vagrant.

---

---

# Box (boîte)

Une **Box** est une machine virtuelle préconfigurée (*template*).

Cela permet d'accélérer le processus de développement et la distribution de logiciels.

Le nom d'une boîte est par convention : "développeur/box", par exemple "**ubuntu/bionic64**".

Vagrant Cloud (<https://app.vagrantup.com/boxes/search>) sert de plateforme d'échange pour trouver des boîtes et y déposer ses propres boîtes.

---

---

# Providers (Fournisseurs)

Vagrant est prêt à l'emploi avec la prise en charge de **VirtualBox**, Hyper-V et **Docker**.

Vagrant a également la capacité de gérer d'autres types de machines en utilisant d'autres *providers* (VMware, AWS, ...).

L'installation d'autres *providers* se fait via le système de *plugin*.

---

# Provisioning (Approvisionnement)

Le *provisioning* permet d'installer automatiquement des logiciels, de modifier des configurations, etc. sur la machine.

Intérêt : automatiser le processus afin qu'il soit reproductible.

Il est possible d'utiliser les outils de *Provisioning* suivants : File, **Shell**, Ansible, CFEngine, Chef Solo, Chef Client, Docker, Puppet Apply, Puppet Agent et Salt.

---

# Installation (sous GNU/Linux Ubuntu)

```
$ sudo apt update
```

```
$ sudo apt install virtualbox
```

```
$ sudo apt install vagrant
```

*Solution 1*

```
$ wget
```

```
https://releases.hashicorp.com/vagrant/2.2.9/vagrant\_2.2.9\_x86\_64.deb
```

```
$ sudo apt install ./vagrant_2.2.9_x86_64.deb
```

*Solution 2*

```
$ vagrant --version
```

---

---

# Initialisation du projet

Il est conseillé d'initialiser un projet dans un répertoire.

Il faut exécuter la commande : **\$ vagrant init**

Un fichier **Vagrantfile** est créé.

La commande **vagrant init** prend deux paramètres optionnels :

**\$ vagrant init [box-name] [box-url]**

Le fichier **Vagrantfile** est généré à partir d'une box.

---



---

# Démarrage rapide

```
$ mkdir ubuntu-bionic64; cd ubuntu-bionic64
```

```
# on recupere un Vagrantfile definissant la box
```

```
$ vagrant init "ubuntu/bionic64"
```

```
# on crée la machine avec VirtualBox
```

```
$ vagrant up --provider=virtualbox
```

```
# on se connecte à la machine en SSH
```

```
$ vagrant ssh
```

---

---

# Arrêt

# arrête la machine virtuelle

\$ vagrant halt

# éventuellement, supprime la machine virtuelle

\$ vagrant destroy

# Voir aussi : suspend, reload

---

---

# Gestion des box

# ajouter une box

\$ vagrant box add box-name [box-url]

# supprimer une box

\$ vagrant box remove box-name

# lister les box

\$ vagrant box list

---

# Configuration

Toute la configuration se fait dans le **Vagrantfile** :

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip:
  "127.0.0.1"
  # config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |v|
    v.gui = true
    v.memory = 8192
    v.cpus = 2
    v.customize ["modifyvm", :id, "--vram", "128"]
    v.customize ["modifyvm", :id, "--usb", "on"]
  end

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

---

# Création d'une box

Il y a plusieurs possibilités, en voici une pour VirtualBox :

➡ [Tutoriel](#)

Documentation : [Creating a Base Box](#)

---

# Distribution d'une box

Il y a plusieurs possibilités, en voici une :


➡ [HashiCorp's Vagrant Cloud](#)

Documentation : [Vagrant Cloud](#)

## Discover Vagrant Boxes

Provider any virtualbox vmware libvirt more

Sort by Downloads Recently Created Recently Updated

	<div>tvaira/ubuntu-1604-bts 0.0.1</div> <div>virtualbox</div> <div>Environnement de travail pour mes étudiants de BTS</div>	Downloads 0	Released 34 minutes ago
-----------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------	----------------	----------------------------

# Oh ! Une box !

tvaira / [ubuntu-1604-bts](#) Vagrant box

Versions

New Version

Settings

Access

How to use this box with [Vagrant](#):

[Vagrantfile](#) [New](#)

```
vagrant init tvaira/ubuntu-1604-bts \  
  --box-version 0.0.1  
vagrant up
```

[v0.0.1](#) currently released version [Edit](#)

This version was created 34 minutes ago.

Version initiale

1 provider for this version.

**virtualbox** Hosted by Vagrant Cloud (9.03 GB) [Edit](#)



[Add a provider](#)

---

# C'est parti !

```
$ vagrant box add "tvaira/ubuntu-1604-bts"
```

```
$ vim Vagrantfile
```

```
...  
config.vm.provider "virtualbox" do |v|  
  v.gui = true  
  v.memory = 2048  
  v.cpus = 2  
  v.customize ["modifyvm", :id, "--vram", "128"]  
  v.customize ["modifyvm", :id, "--usb", "on"]  
end
```

```
$ vagrant init "tvaira/ubuntu-1604-bts"
```

```
$ vagrant up
```

---



---

# Provisioning

Le *provisionning* se fait soit :

- la première fois lors du **vagrant up**
- à la demande :
  - **vagrant up --provision**
  - **vagrant provision**

---

# Provisioning (script shell)

La configuration du *provisionning* se fait dans le fichier **Vagrantfile**:

```
Vagrant.configure("2") do |config|  
  config.vm.box = "ubuntu/bionic64"  
  
  # Install the required software  
  config.vm.provision "shell",  
    path: "provision.sh"  
  
  # Run every time the VM starts  
  config.vm.provision "shell",  
    path: "run.sh",  
    args: ENV['SHELL_ARGS'],  
    run: "always"  
  
end
```

---

# Provisioning (Chef)

**Chef** est un *Provisionner* pour **Vagrant**. C'est un logiciel libre de gestion de configuration (initialement écrit en Ruby) qui permet d'installer et de configurer des environnements à partir de « *recettes* » (*recipes*) ou de « **livres de recettes** » (*cookbook*).

Il est possible d'utiliser **Berkshelf** pour gérer les *cookbooks* Chef. Berkshelf est un gestionnaire de dépendance pour les *cookbooks* Chef. A partir d'un fichier **Berksfile**, il est facile d'inclure et d'utiliser les *cookbooks* communautaires. Berkshelf est inclus dans Chef Workstation.

Il faudra installer le *plugin* pour Vagrant et Chef Workstation.

Voir aussi : [Puppet](#), [Ansible](#).

---

# Provisioning (Chef)

```
$ vagrant plugin install vagrant-berkshelf
```

```
$ wget
```

```
https://packages.chef.io/files/stable/chef-workstation/0.18.3/ubuntu/18.04/chef-workstation_0.18.3-1_amd64.deb
```

```
$ sudo dpkg -i chef-workstation_0.18.3-1_amd64.deb
```

```
$ vim Berksfile
```

```
source 'https://supermarket.chef.io'
```

```
cookbook 'apt'
```

```
cookbook 'build-gcc'
```

```
cookbook 'subversion'
```

---

---

# Provisioning (Chef)

```
$ vagrant init ubuntu/bionic64
```

```
$ vim Vagrantfile
```

```
...
```

```
config.berkshelf.enabled = true
config.vm.provision :chef_solo do |chef|
  chef.arguments = "--chef-license accept"
  chef.add_recipe "apt"
  chef.add_recipe "build-gcc"
  chef.add_recipe "subversion"
end
```

```
$ vagrant up
```

```
$ vagrant ssh
```