

Vagrant

Table des matières

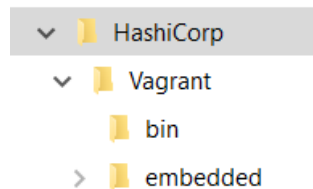
I - Vagrant (présentation).....	3
1. Présentation de Vagrant.....	3
1.1. Caractéristiques et principes.....	3
1.2. Installation de Vagrant.....	3
2. Prise en main de Vagrant.....	4
2.1. Initialisation élémentaire d'un projet vagrant.....	4
2.2. Lancement/démarrage d'une "box" vagrant	4
2.3. Vagrant ssh pour piloter (en mode texte) la machine virtuelle.....	5
2.4. Arrêt ou suppression de la machine virtuelles.....	6
2.5. Répertoire(s) partagé(s).....	7
2.6. Vagrant (opérations diverses).....	7
II - Essentiel de vagrant (config,usage).....	8
1. Concepts / terminologies de Vagrant.....	8
1.1. Provider.....	8
1.2. Box.....	8
1.3. Provisioning.....	9
2. Exemples de configurations (Vagrantfile).....	10
3. Provisioning via copie de fichier(s) ou répertoire(s).....	10
4. Provisioning via script shell.....	10
5. Provisioning via Docker.....	11

6. Provisioning via Ansible.....	11
7. Construire une box à partir de la machine virtuelle.....	11
8. Déploiement via ftp (ou sftp).....	11

I - Vagrant (présentation)

1. Présentation de Vagrant

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**.



Un des Principaux intérêts de Vagrant : **démarrer facilement une machine virtuelle linux légère (sur un poste Microsoft-Windows) de façon à démarrer des conteneurs Dockers dans cette machine virtuelle.**

Depuis la version 1.1, Vagrant n'impose plus VirtualBox, mais fonctionne également avec d'autres logiciels de virtualisation tels que VMware.

Depuis la version 1.6, Vagrant fournit un support natif des conteneurs Docker à l'exécution, au lieu d'un système d'exploitation entièrement virtualisé.

Vagrant prend en charge certains environnements de serveurs comme Amazon EC2.

Bien qu'écrit en Ruby, il est utilisable dans des projets écrits dans d'autres langages de programmation tels que PHP, Python, Java, C# et Javascript/typescript.

1.1. Caractéristiques et principes

Vagrant s'utilise en **ligne de commande** et se configure en mode "projet" via un **fichier de configuration** nommé **Vagrantfile**.

Vagrant manipule des "**Box**" (*machines virtuelles pré-configurées*) que l'on peut télécharger (et personnaliser/enrichir).

1.2. Installation de Vagrant

Téléchargement de l'installateur (pour Linux ou Windows ou ...) via <https://www.vagrantup.com/downloads.html>.

NB1 : installer préalablement **VirtualBox**.

NB2 : **Lorsque Vagrant est utilisé sous Windows avec VirtualBox, il faut désactiver (si besoin) Hyper-V**. Ceci peut s'effectuer via la commande (power-shell) suivante :

```
Disable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V-All
```

Normalement pas de "Hyper-V" intégré dans Windows 10 Home edition.

pour tester l'installation : **vagrant --help**

2. Prise en main de Vagrant

2.1. Initialisation élémentaire d'un projet vagrant

1. **Préparer un répertoire** (pour projet vagrant) (ex : val) et se **placer dedans**
2. Lancer la commande **vagrant init** (avec ou pas certaines options)

La commande vagrant init permet de générer le fichier de configuration nommé **Vagrantfile** .

Avec l'option **--minimal** , le fichier Vagrantfile généré est le suivant :

```
Vagrant.configure("2") do |config|
  config.vm.box = "base"
end
```

Par défaut, sans l'option **--minimal**, le fichier Vagrantfile généré comporte plein d'exemples de configurations en commentaires .

La commande **vagrant init** prend deux paramètres optionnels **[box-name]** et **[box-url]**.

En spécifiant un "box-name" on génère un Vagrantfile depuis un modèle approprié ("Box").

L'option "box-url" est l'URL indiquée où la "Box" peut être téléchargée. Ces informations peuvent également être configurées dans le Vagrantfile.

Exemple :

vagrant init hashicorp/bionic64

génère le fichier **Vagrantfile** suivant :

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/bionic64"
end
```

2.2. Lancement/démarrage d'une "box" vagrant

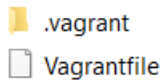
```
vagrant up
```

Cette commande va permettre (au premier lancement) de :

- *télécharger l'image "virtualBox" de la machine virtuelle*
- *de l'initialiser un peu (Add On "guest addition" ...)*

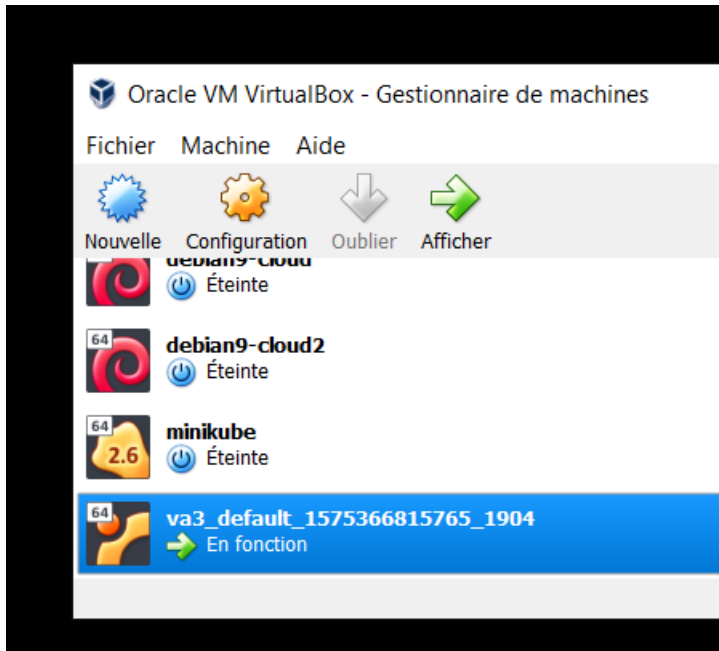
avant de

- *démarrer cette machine virtuelle*



Le répertoire caché **.vagrant** placé à coté de Vagrantfile comporte quelques fichiers de configurations dynamiques (rangés dans des sous répertoires) .

La machine virtuelle lancée/démarrée par vagrant (et gérée par VirtualBox) est visible via l'ihm de virtualBox :



2.3. Vagrant ssh pour piloter (en mode texte) la machine virtuelle

la commande **vagrant ssh** permet de démarrer un "remote shell" (un peu comme putty) de façon à se connecter à la machine virtuelle démarrée et lui adresser des instructions .

vagrant ssh

```
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)Welcome to Ubuntu
18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)
```

- * Documentation: <https://help.ubuntu.com>
- * Management: <https://landscape.canonical.com>
- * Support: <https://ubuntu.com/advantage>

```
System information as of Tue Dec 3 10:05:13 UTC 2019
```

```
System load: 0.0          Processes:      88
Usage of /:  2.5% of 61.80GB Users logged in:    0
Memory usage: 11%         IP address for eth0: 10.0.2.15
Swap usage:  0%
```

```
vagrant@vagrant:~$ ls
vagrant@vagrant:~$ cd /
vagrant@vagrant:/$ ls
```

```
bin dev home    initrd.img.old lib64    media opt root sbin srv tmp vagrant vmlinuz
boot etc initrd.img lib        lost+found mnt  proc run  snap sys usr var   vmlinuz.old
vagrant@vagrant:/$ exit
```

2.4. Arrêt ou suppression de la machine virtuelles

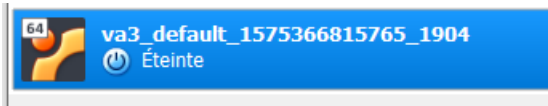
La commande **vagrant halt** permet d'arrêter la machine virtuelle démarrée via vagrant up .
==> les fichiers modifiés à l'intérieur de la machine virtuelle ne seront pas perdus au prochain redémarrage (via vagrant up) .

La commande **vagrant destroy** permet en plus de supprimer toutes les ressources utilisées .

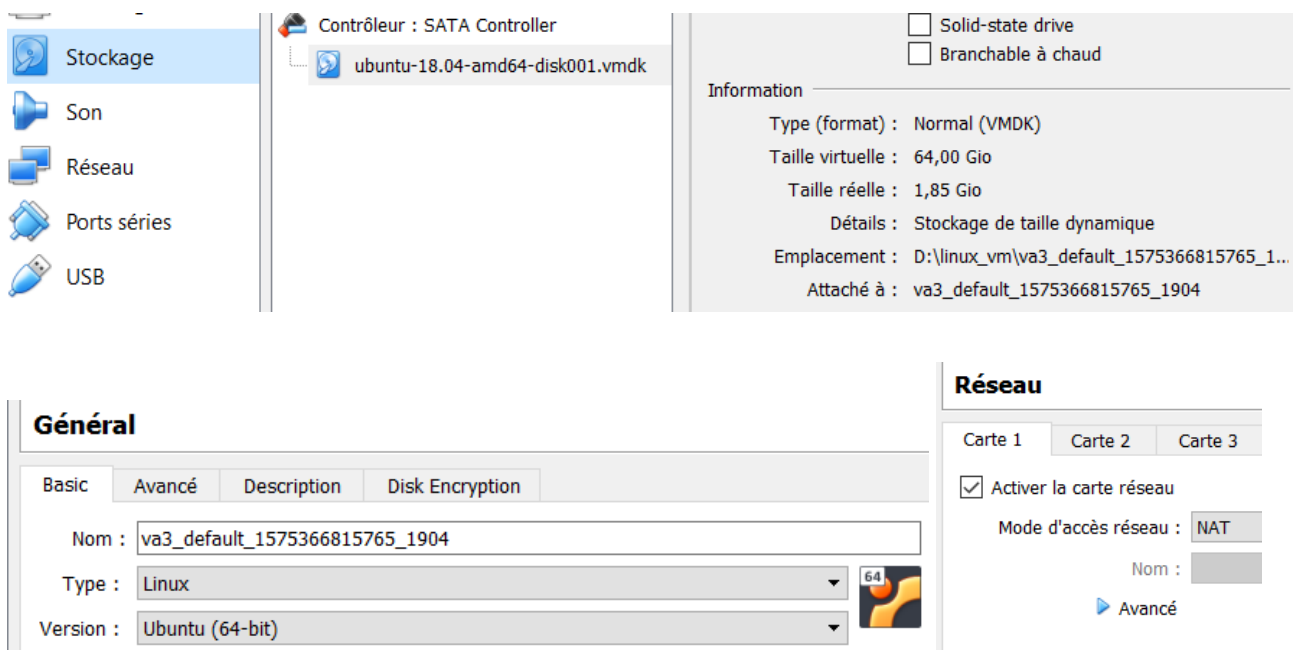
Ces 2 commandes ont une éventuelle option **-f** (pour forcer l'action en cas de problème) .

vagrant halt

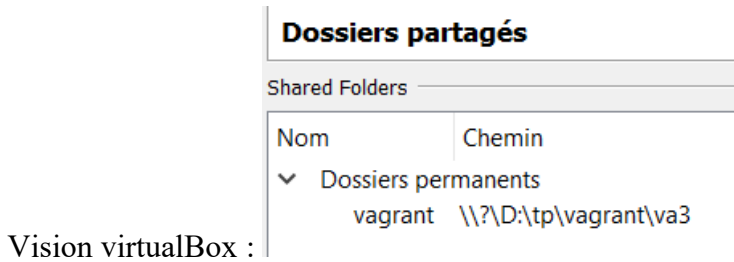
==> default: Attempting graceful shutdown of VM...



Les principales ressources utilisées sont gérées par VirtualBox :



2.5. Répertoire(s) partagé(s)



Par défaut vagrant partage le répertoire du projet vagrant (ex : va3) avec la machine virtuelle associée à la box . Il est possible de configurer d'autres partages additionnels .

NB : Au sein de la machine virtuelle (*accessible via vagrant ssh*) , le répertoire partagé est **/vagrant** (à ne pas confondre avec *home/vagrant*) .

Exemple : le fichier **va3/my-dir/fa.txt** (édité sous l'hôte windows avec notepad++)
est vu en interne comme **/vagrant/my-dir/fa.txt** (éditable via nano ou bien vi)

2.6. Vagrant (opérations diverses)

```
vagrant global-status
```

id	name	provider	state	directory

1102325	default	virtualbox	running	D:/tp/vagrant/va3

On peut suspendre et reprendre l'exécution d'une machine virtuelle (associée à une box) via les commandes suivantes :

vagrant suspend [name|id]

vagrant resume [name|id]

Attention : les fichiers temporaires alors générés nécessitent beaucoup d'espace disque .

Eventuelle gestion de plugins pour vagrant :

vagrant plugin list

```
vagrant plugin install vagrant-hostmanager
vagrant plugin install vagrant-proxyconf
```

II - Essentiel de vagrant (config,usage)

1. Concepts / terminologies de Vagrant

1.1. Provider

Vagrant appelle "**provider**" la technologie qui gère la machine virtuelle (par défaut "**VirtualBox**")






Rappel : machine **hôte** = machine physique (ex : windows avec le logiciel virtualBox)

machine/os **invité** = machine virtuelle (interne)

1.2. Box

Une **Box** (de vagrant) correspond à une **configuration de machine virtuelle** (et indirectement à la machine virtuelle sous-jacente).

NB: on peut rechercher une "**box prédéfinie**" sur le site <https://app.vagrantup.com/boxes/search>

	hashicorp/precise64 1.1.0 A standard Ubuntu 12.04 LTS 64-bit box.	hyperv virtualbox vmware_fusion	Downloads 6,762,611	Released over 5 years ago
	centos/7 1905.1 CentOS Linux 7 x86_64 Vagrant Box	hyperv libvirt virtualbox vmware and 3 more providers	Downloads 5,080,470	Released 5 months ago
	ubuntu/xenial64 20191126.1.0 Official Ubuntu 16.04 LTS (Xenial Xerus) builds	virtualbox	Downloads 3,328,414	Released 7 days ago
	debian/jessie64 8.11.1 Vanilla Debian 8 "Jessie"	libvirt lxc virtualbox	Downloads 2,360,991	Released 6 months ago
	hashicorp/bionic64 1.0.282 A standard Ubuntu 18.04 LTS 64-bit box	hyperv virtualbox vmware_desktop	Downloads 9,520	Released 4 months ago

On peut explicitement télécharger une box via la commande

vagrant box add [box-name] et

inversement supprimer une box via la commande **vagrant box remove [box-name]** .

La liste des box téléchargées sur la machine locale peut être affichée via la commande

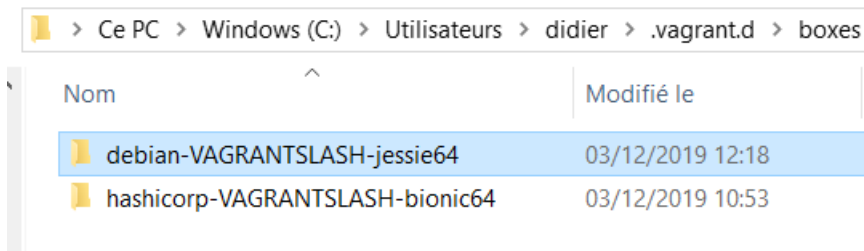
vagrant box list

vagrant box add debian/jessie64

vagrant box list

debian/jessie64 (virtualbox, 8.11.1)

hashicorp/bionic64 (virtualbox, 1.0.282)



Nom	Modifié le
debian-VAGRANTSLASH-jessie64	03/12/2019 12:18
hashicorp-VAGRANTSLASH-bionic64	03/12/2019 10:53

NB : hashicorp/bionic64 est basée sur **ubuntu 18 LTS** et comporte dès le départ l'éditeur "**nano**" (un peu plus facile à utiliser que "vi") .

1.3. Provisioning

Opérations de "post-configuration" de la machine virtuelle
(ex : **installation et configuration de logiciels**) .

Par défaut (*sauf si option run: "always" de config.vm.provision*) , le "**provisioning**" (à configurer dans le fichier Vagrantfile) ne sera **automatiquement déclenché que lors du premier démarrage (via vagrant up)**

Il pourra être explicitement déclenché via la commande **vagrant provision**

On pourra déclenché explicitement un "redémarrage + provisioning" via la commande **vagrant reload -provision**

Finalement, l'option **--no-provision** permet de lancer "**vagrant up**" en étant certain de ne pas déclencher de "provisioning"

et l'option **--provision** permet de lancer "**vagrant up**" en demandant à déclencher le "provisioning" même s'il s'agit pas du premier lancement (ex : n ème redémarrage après "vagrant halt") .

Concrètement le "provisioning" (installation & configuration d'applications dans la machine virtuelle) peut être effectué par :

- des simples **copies de fichiers**
- **des scripts "shell/linux" (.sh)** (comportant par exemple des instructions apt-get ...)
- des instructions **"Docker"** (ex : téléchargement d'images et démarrage de conteneurs)
- **des technologies spécialisées** (ex : "**Chef**" , "**Puppet**" , "**Ansible**" , "**SaltStack**" , ...) avec souvent des versions "limitées/free" et "complètes/payantes".

Chef étant plus récent que "Puppet" , il vaut mieux (en général) utiliser "Chef" que "Puppet" .

	Chef	Ansible
syntaxe	DSL (ruby)	YAML (python)
...

Ansible is **available** for **free** and runs on Linux, Mac or BSD. Aside from **the free** offering, **Ansible** also has an enterprise product called **Ansible Tower**.

On peut configurer et utiliser de multiples "provisionings" complémentaires .

2. Exemples de configurations (Vagrantfile)

Fixer la taille de la mémoire vive allouée à la machine virtuelle (de type virtualBox) :

```
#...
config.vm.provider "virtualbox" do |vb|
  # # Display the VirtualBox GUI when booting the machine
  # vb.gui = true
  #
  # # Customize the amount of memory on the VM:
  vb.memory = "2048"
end
#...
```

Partager un répertoire supplémentaire entre machines "hôte" et "invitée" :

```
config.vm.synced_folder "../data", "/vagrant_data"
```

...

3. Provisioning via copie de fichier(s) ou répertoire(s)

```
config.vm.provision "file", source: "~/gitconfig", destination: ".gitconfig"
config.vm.provision "file", source: "~/path/to/host/folder", destination: "$HOME/newfolder"
```

4. Provisioning via script shell

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", path: "my-install-script.sh"
end
```

Le script "my-install-script.sh" (du projet vagrant de la machine "hôte") sera recopié et exécuté sur la machine virtuelle.

Et plein d'autres possibilités décrites sur le site de référence de Vagrant :

<https://www.vagrantup.com/docs/provisioning/shell.html>

5. Provisioning via Docker

...

6. Provisioning via Ansible

```
Vagrant.configure("2") do |config|  
  # Run Ansible from the Vagrant VM  
  config.vm.provision "ansible_local" do |ansible|  
    ansible.playbook = "playbook.yml"  
  end  
end
```

7. Construire une box à partir de la machine virtuelle

vagrant **package** -h

...

8. Déploiement via ftp (ou sftp)

vagrant **push**

à paramétrer (dans Vagrantfile) par

```
config.push.define "ftp" do |push|  
  push.host = "ftp.test.com"  
  push.username = "utilisateur1"  
  push.password = "mot de passe"
```

```
push.secure = false  
push.destination = "/"  
push.dir = "/"  
end
```

ANNEXES