Infrastructure as Code (IaC) avec Vagrant

Vagrant, c’est quoi?

- Lancé en 2010 par Mitchell Hashimoto en tant que projet parallèle et est devenu plus tard l'un des premiers produits de HashiCorp

- Vagrant
  - un outil en ligne de commande
  - multiplateforme
  - créer des environnements virtuels légers, reproductibles et portables.

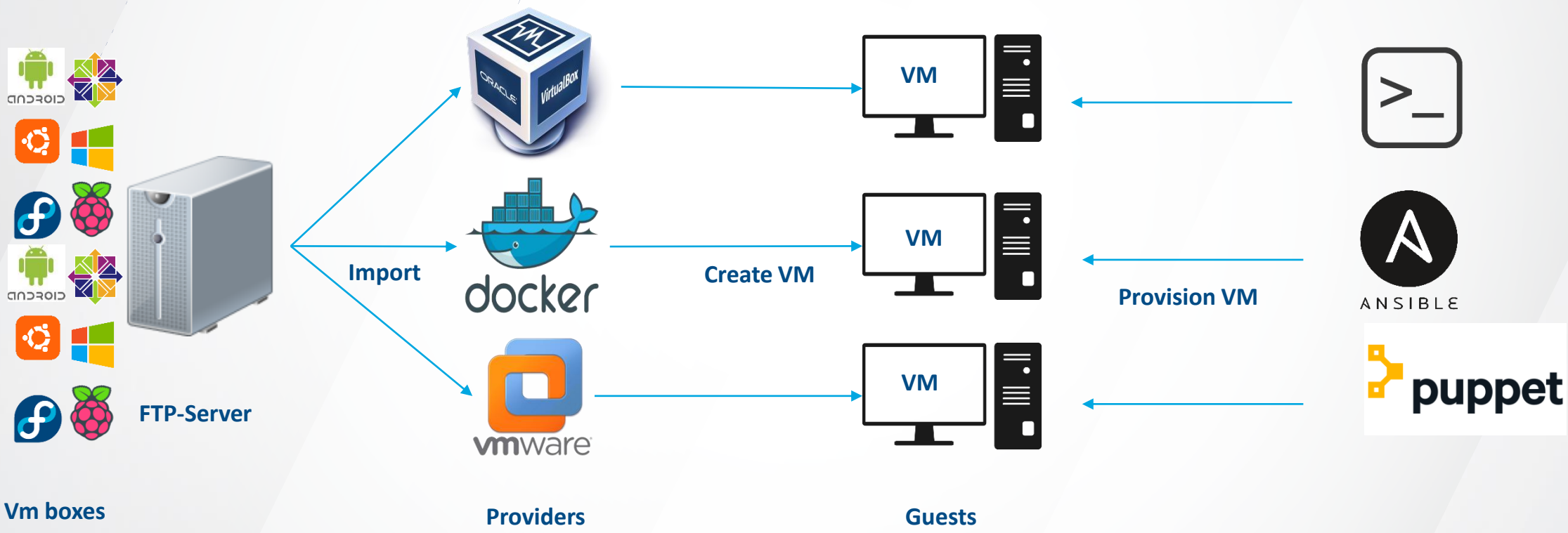
Vagrant, c’est quoi?

- Inetrmédiaire entre les solutions de virtualisation (logiciel, matériel, PaaS et IaaS) et les utilitaires de gestion de configuration (Puppet, Ansible ..).
- Configurer des environnements de développement
- Egalement utilisé pour:
  - Création de lab de démonstration
  - Test des outils de gestion de configuration
  - Accélérer le travail avec des outils non multiplateformes tels que Docker

Terminologie

- **Box**: un box est un environnement Vagrant empaqueté, est généralement une machine virtuelle
- **Provider**: Un provider fournit un support de virtualisation, les providers peuvent être de 2 types:
  - Local: VirtualBox, Vmware, Docker, Hyper-V
  - Cloud: AWS, Google Cloud, DigitalOcean Cloud, OpenStack
- **Vagrantfile**: fichier de configuration pour la configuration de la machine cible. Il contient des informations telles que: Image de base à utiliser, Le nom d'hôte, Le réseau, Configuration du Provisionnement,…
- **Provisioner**: Un provisioner est un outil pour configurer l'environnement virtuel, et il peut être un simple script shell, il peut également être un outil d'automatisation comme Ansible, Puppet…

Vagrant Workflow



Vagrant, pour quoi faire?

- Créer de nouvelles machines virtuelles rapidement et facilement
  - Une seule commande! `vagrant up`
- Garder le nombre de VM sous contrôle
- Reproductibilité
- Environnement identique en développement et en production
- Portabilité
  - No more 4GB+ .vmdk/.vdi files to share
  - `git clone` and `vagrant up`

What happens under the hood?

```
$ vagrant up
```

- Le box de base est téléchargé et stocké localement
  - ~/.vagrant.d/boxes/
- Une nouvelle machine virtuelle est créée et configurée avec le box de base comme modèle
- La VM est démarrée
- Le box est approvisionné
  - seulement la première fois, doit être fait manuellement après

What happens under the hood?

- **DONE!**
- Vous disposez maintenant d'une VM fonctionnelle, prête à l'emploi:

```
$ vagrant ssh ↵

[vagrant@localhost ~]$ cat /etc/redhat-release
CentOS Linux release 7.1.1503 (Core)
[vagrant@localhost ~]$
```

NB : Login et mot de passe par défaut : vagrant

Using Vagrant

- Initialiser un nouvel environnement Vagrant

```
# mkdir /path/to/directory
# cd /path/to/directory
# vagrant init <box_image>
```

- "**init**" crée un nouveau **Vagrantfile** dans le répertoire courant,
- vous pouvez personnaliser le fichier selon les exigences

Using Vagrant

- Liste des VMs initialisées

```
# vagrant global-status

id       name    provider   state   directory
------------------------------------------------------------
d4adff9  default virtualbox running /home/user/Vagrant
```

Using Vagrant

- Démarrer et approvisionner l'environnement avec vagrant up
  - à partir du répertoire où se trouve le Vagrantfile

```
# vagrant up
```

  - De n’importe quel repertoire

```
# vagrant up <box_id>
```

Using Vagrant

- Connecter au VM via SSH

```
# vagrant ssh
```

  - OU

```
# ssh -p <port> 127.0.0.1 -l vagrant
```

  - En cas d'erreur d'autorisation refusée, ajoutez l'option:

```
-i <path/to/private_key>
```

- Vagrant alloue une nouvelle redirection de port ssh pour chaque nouvelle VM.
- Les informations d'identification par défaut pour la machine Vagrant sont généralement «vagrant / vagrant»

Vagrant CLI commandes souvent utilisées

```
$ vagrant status          #status of the vagrant machine
$ vagrant global-status   #status of the vagrant environments of this user
$ vagrant up              #start and provision the vagrant environment
$ vagrant halt            #stop the vagrant machine
$ vagrant suspend         #suspend the vagrant machine
$ vagrant destroy         #stop and delete all traces of the vagrant machine
$ vagrant reload          #restart the vagrant machine and reload the Vagrantfile config
$ vagrant port            #displays information about the guest port mapping
```

Configuring vagrant boxes: VAGRANTFILE

- Vagrantfile = Ruby
- Minimal Vagrantfile:

```
VERSION = '2'
Vagrant.configure(VERSION) do |config|

        config.vm.box = 'centos/7'

end
```

PROVISIONING

- From *Just Enough Operating System* to fully functional configured box
  - **Shell script**
  - **Ansible**
  - **Puppet**
  - Chef
  - Docker
  - Salt

Cas Pratique

Créer Vagrantfile

- Créer un dossier de travail
  - $ mkdir apache
  - $ cd apache/

- Créer le Vagrantfile:
  - $ nano Vagrantfile
  - $ cat Vagrantfile

```
Vagrant::Config.run do |config|
    config.vm.box = "precise64"
    config.vm.forward_port 80, 8080
end
```

Lancer la VM

- $ vagrant up

```
Bringing machine 'default' up with 'virtualbox' provider...
[default] Importing base box 'precise64'...
[default] Matching MAC address for NAT networking...
[default] Setting the name of the VM...
[default] Clearing any previously set forwarded ports...
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
[default] -- 22 => 2222 (adapter 1)
[default] -- 80 => 8080 (adapter 1)
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
[default] VM booted and ready for use!
[default] Configuring and enabling network interfaces...
[default] Mounting shared folders...
[default] -- /vagrant
```

Lancer la VM

- En cas de problème de système de fichier vbguestfs, installer le plugin vbguest
- $ vagrant plugin install vagrant-vbguest

Etat de la VM

- $ vagrant status

```
Current machine states:
default  running (virtualbox)
The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
```

Contrôler la VM

- $ vagrant ssh

```
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)
* Documentation: https://help.ubuntu.com/
Welcome to your Vagrant-built virtual machine.
Last login: Fri Sep 14 06:23:18 2012 from 10.0.2.2
vagrant@precise64:~$
```

Répertoire partagé

- **Vagrant** prend en charge la configuration de dossiers partagés afin que les fichiers et les dossiers soient synchronisés vers et depuis la machine virtuelle et la machine hôte.
- Avoir un système de fichiers partagé entre la machine hôte et la VM présente de nombreux avantages:
  - Développeur: Modifier des fichiers dans la hôte et utiliser dans la VM
  - stocker des fichiers qui ne seront pas détruits suite à `vagrant destroy .`
  - sauvegarder facilement les fichiers des projets.

Répertoire partagé

- À partir de la VM:

```
vagrant@precise64:~$ ls /vagrant/
Vagrantfile
```

=> C'est le Vagrantfile dans le dossier du projet

- L'emplacement du dossier partagé par défaut peut être remplacé à partir du fichier Vagrant:

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

  config.vm.synced_folder "src/", "/srv/website"
end
```

Répertoire partagé

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

  config.vm.synced_folder "src/", "/srv/website"
end
```

- La directive config.vm.synced_folder définit un dossier partagé pour Vagrant. Cela prend quelques paramètres:
  - /foo est le chemin où le dossier existera sur la VM. Ce chemin sera créé s'il n'existe pas déjà. S'il existe déjà, l'emplacement sera remplacé par le contenu du dossier partagé.
  - Le troisième paramètre, "." est le chemin du dossier à partager depuis la machine hôte. Cela peut être un chemin absolu ou relatif. S'il est relatif, comme dans l'exemple, il est relatif à la racine du projet. Ainsi, dans l'exemple, il partage la racine du projet.

Des bases sur le Réseau

- Vagrant configure automatiquement diverses options de mise en réseau avec la machine virtuelle.
- Cela permet aux développeurs utilisant Vagrant de communiquer avec la machine.
- Cette fonctionnalité est importante pour des environnements tels que des projets Web,
- Les développeurs peuvent continuer à utiliser leur propre navigateur et leurs propres outils de développement pour accéder à leur projet, tandis que le code de l'application Web lui-même et toutes ses dépendances s'exécutent de manière isolée dans la VM.

Des bases sur le Réseau

- config.vm.forward_port 80, 8080

```
$ vagrant reload
[default] Attempting graceful shutdown of VM...
[default] Setting the name of the VM...
[default] Clearing any previously set forwarded ports...
[default] Fixed port collision for 22 => 2222. Now on port 2200.
[default] Creating shared folders metadata...
[default] Clearing any previously set network interfaces...
[default] Preparing network interfaces based on configuration...
[default] Forwarding ports...
[default] -- 22 => 2200 (adapter 1)
[default] -- 80 => 8080 (adapter 1)
[default] Booting VM...
[default] Waiting for VM to boot. This can take a few minutes.
[default] VM booted and ready for use!
[default] Configuring and enabling network interfaces...
[default] Mounting shared folders...
[default] -- /vagrant
```

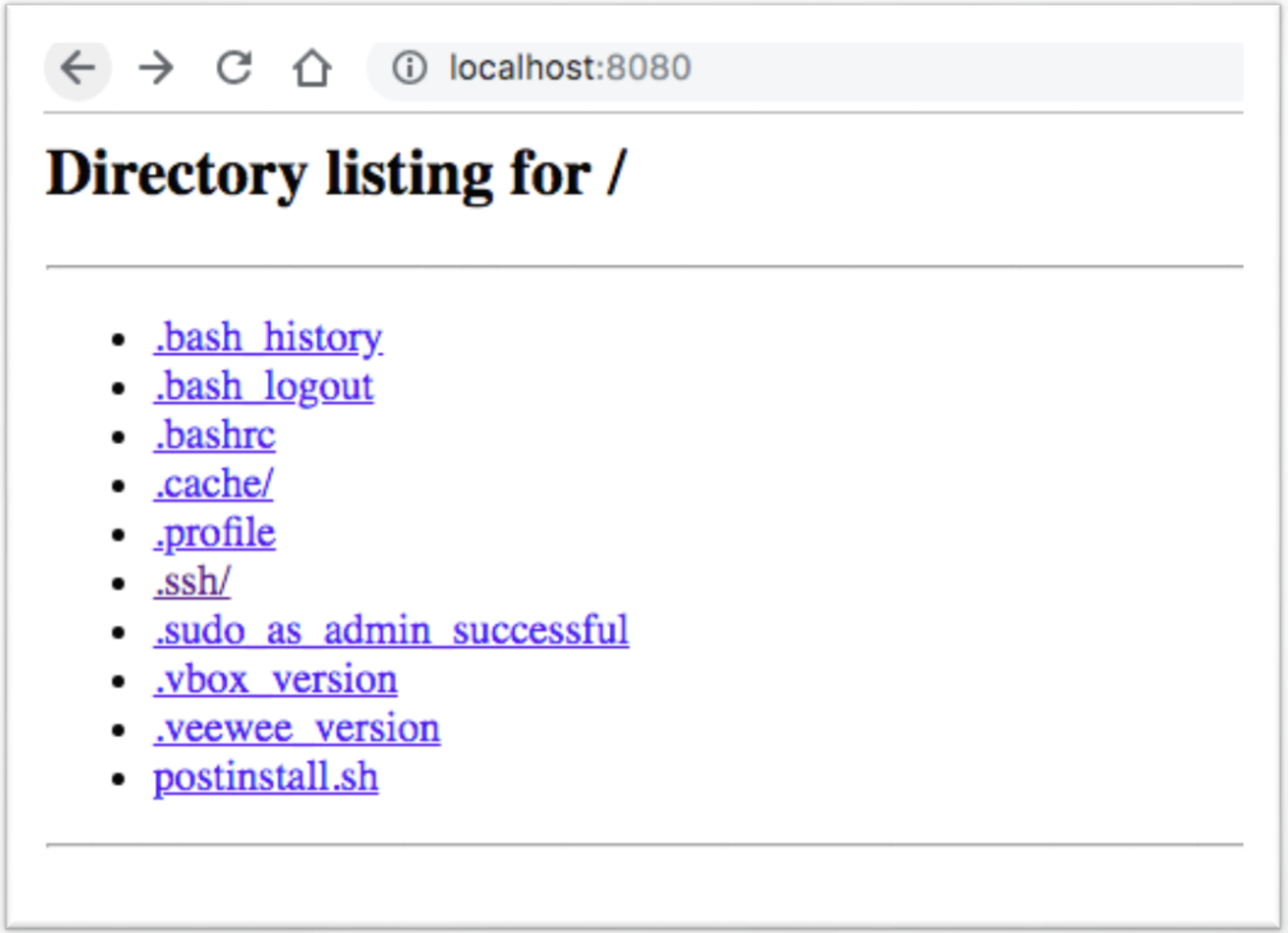
Des bases sur le Réseau

- Tester le port

```
$ vagrant ssh
vagrant@precise64:~$ cd /vagrant
vagrant@precise64:/vagrant$ sudo python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

- On utilise une ligne de commande simple en ligne utilisant Python pour démarrer un serveur Web sur le port 80.
- Ouvrir un navigateur sur localhost:8080 sur la machine hôte et une liste de répertoires de /vagrant, servie depuis la VM, apparaît.

Des bases sur le Réseau



Suspendre la VM

- Vagrant suspendra la machine en utilisant vagrant suspend. L'état de la machine peut être inspecté avec la commande d'état:

```
$ vagrant suspend
[default] Saving VM state and suspending execution...

$ vagrant status
Current machine states:

default                   saved (virtualbox)

To resume this VM, simply run `vagrant up`.
```

Arrêter la VM

- Vagrant arrêtera la machine lorsque vagrant halt est appelé. L'état de la machine, comme toujours, peut être inspecté en utilisant vagrant statut

```
$ vagrant halt
[default] Attempting graceful shutdown of VM...

$ vagrant status
Current machine states:

default                   poweroff (virtualbox)

The VM is powered off. To restart the VM, simply run `vagrant up`
```

Détruire la VM

- $ vagrant destroy

```
Are you sure you want to destroy the 'default' VM? [y/N] y
[default] Forcing shutdown of VM...
[default] Destroying VM and associated drives...
```

- La destruction d'une machine vous fera perdre toutes les modifications apportées à la machine pendant son fonctionnement, y compris tous les fichiers ou dossiers créés en dehors des dossiers partagés.

Approvisionnement

Approvisionnement Manuel

vagrant@precise64:~$ sudo apt-get update

vagrant@precise64:~$ sudo apt-get install apache2

vagrant@precise64:~$ sudo rm -rf /var/www

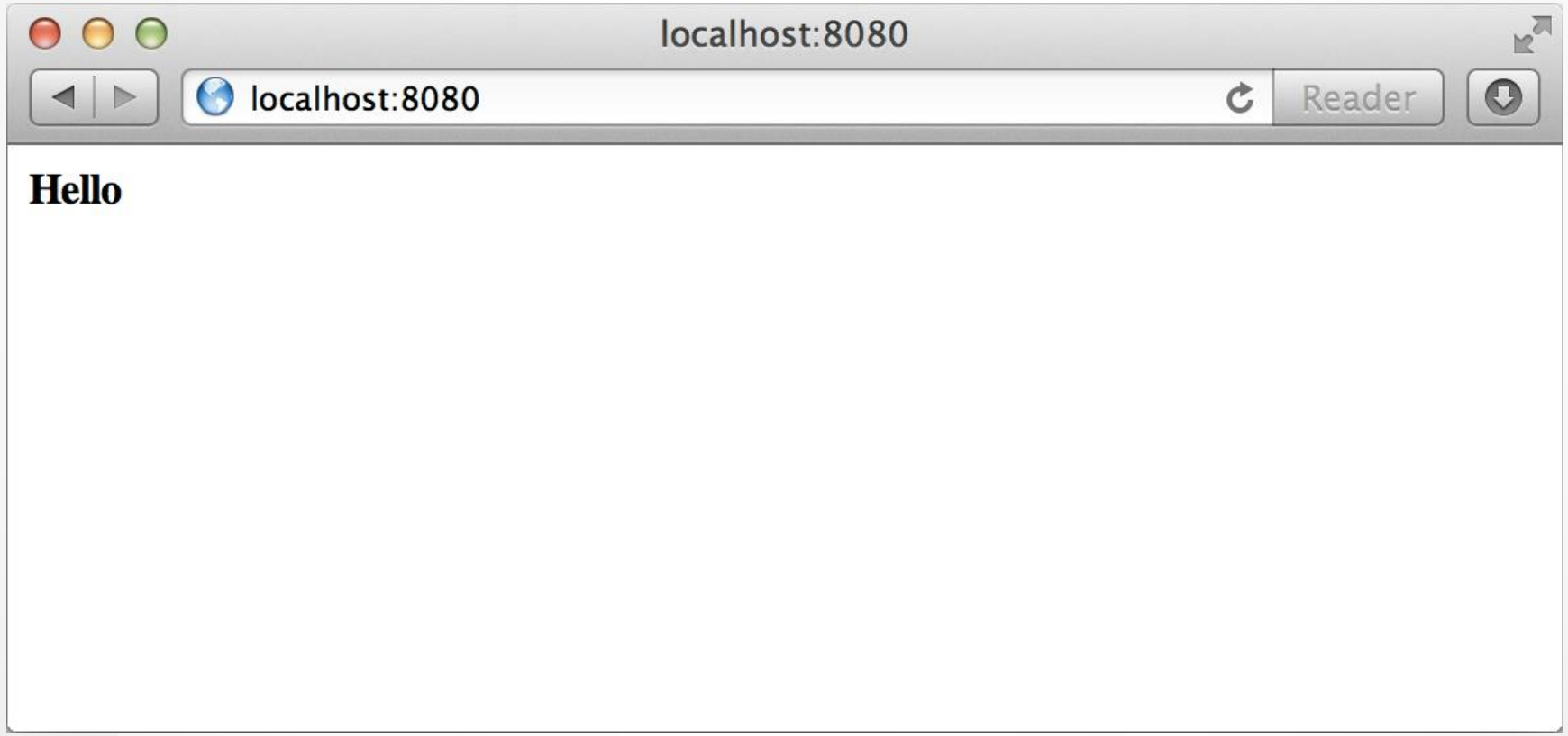
vagrant@precise64:~$ sudo ln -fs /vagrant /var/www

vagrant@precise64:~$ logout

Connection to 127.0.0.1 closed.

$ echo "Hello!" > index.html

Approvisionnement Manuel



Approvisionnement par script Shell

$ nano provision.sh

$ cat provision.sh

```
#!/usr/bin/env bash
echo "Installing Apache and setting it up..."
apt-get update > /dev/null 2>&1
apt-get install -y apache2 >/dev/null 2>&1
rm -rf /var/www
ln -fs /vagrant /var/www
```

Approvisionnement par script Shell

Une fois le script shell créé, l'étape suivante consiste à configurer Vagrant pour utiliser le script. Ajouter la ligne suivante quelque part dans le Vagrantfile:

```
config.vm.provision "shell", path: "provision.sh"
```

Approvisionnement avec Puppet

Préparer le manifest puppet manifests/default.pp

```
exec { "apt-get update":
  command => "/usr/bin/apt-get update",
}

package { "apache2":
  require => Exec["apt-get update"],
}

file { "/var/www":
  ensure => link,
  target => "/vagrant",
  force  => true,
}
```