CHEF: Environnement de développement

Table of Contents

Fix Me! Cet article présent la manière de développer et tester une configuration avec vagrant et chef-solo.

Créer une box de base

Au préalable se placer dans un répertoire de travail

```
$ vagrant init debian/jessie64
$ vagrant up
```

Lorsqu'on aura édité les configurations il faudra utiliser la commande suivante pour provisionner à nouveau :

```
$ vagrant provision
```

Créer un cookbook privé

Nous allons créer une petite structure avec des cookbooks chef. Ils seront très basiques et écris en ruby (comme l'est le Vagrantfile d'ailleurs).

Créer la structure globale :

```
$ chef generate cookbook nginx
```

On peut également créer une structure basique d'un cookbook chef

(https://docs.chef.io/index.html#Cookbooks) du type :

```
#share sera notre répertoire partagé
$ mkdir -p cookbook/recipes cookbook/templates/default share
$ touch cookbook/Berksfile cookbook/metadata.rb
```

Editer metadata.rb

Dans le metadata.rb nous aurons quelque chose comme ca :

```
name 'perso'
maintainer 'soyuka'
maintainer_email 'trucmuche@gmail.com'
description 'Une super box vagrant de la mort qui tue'
version '1.0.0'

recipe 'perso', 'Mon cookbook perso'

depends 'apt'
depends 'nvm'
depends 'nginx'
depends 'php'

%W{ debian ubuntu }.each do |os|
    supports os
end
```

Créer les recettes

Recette pour les paquets

Première recette toute basique qui va nous servir à installer des paquets depuis la configuration. La configuration est en fait la configuration chef que nous allons créer dans notre Vagrantfile. Chaque nœud de celle-ci se retrouve ensuite dans la variable globale “node”.

```
$ vi cookbooks/nginx/recipe/default.rb

node['perso']['packages'].each do |p| #
  package p do                              # traitement des paquets en
séquence
     source "/tmp/#{p}"                     # concaténation chemin/nom du
paquet
     action :nothing                        # ne rien faire sinon définir la
ressource
  end
  cookbook_file "/tmp/#{p}" do              # traitement en séquence des
fichiers
```

```
    source p                              # définir la ressource
    action :create
    notifies :install, "package[/tmp/#{p}]", :immediately
  end
end
```

Recette pour templates

Ajoutons d'abord notre template de vhost dans cookbook/nginx/templates/default/nginx/perso :

```
server {
    listen 80;
    listen [::]:80 ipv6only=on;
    root /var/www/perso;

    server_name localhost;
    index index.php;
    access_log /var/log/nginx/perso-access.log;
    error_log /var/log/nginx/perso-error.log notice;


    location ~ \.php$ {
            try_files $uri =404;
            fastcgi_pass unix:/var/run/php5-fpm.sock;
            fastcgi_index index.php;
            fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;
            include /etc/nginx/fastcgi_params;
    }
}
```

Aller plus loin

On peut aisément ajouter des recettes “inline” via le shell dans le Vagrantfile. Pour ce faire il suffit d'utiliser des “shell provisioners”, par exemple :
`config.vm.provision :shell, :inline => "apt-get update"`

Bien entendu celui-ci est inutile car le cookbook apt se charge déjà de ca !

Configurer Vagrant avec le Vagrantfile

Exemple de config :

```
# -*- mode: ruby -*-
```

```
# vi: set ft=ruby :

# Ici je met des paramètres 'globaux' (pour plus tard)
ip_address = '192.168.33.10'
project_name = 'perso'

Vagrant.configure(2) do |config|
  config.vm.box = 'debian/jessie64'
  # Vagrant va chercher sur l'url officielle de base mais c'est ici que vous
pouvez mettre des box custom
  config.vm.box_url = 'debian/jessie64'

  # Le répertoire partagé (magie, ou presque...)
  config.vm.synced_folder './share', '/var/www/perso/', :mount_options =>
['dmode=777', 'fmode=666']

  config.vm.provider 'virtualbox' do |vb|
   # Mettez max 1/4 de votre ram au cas où, plus pourrait nuire à votre
système
    vb.memory = '4024'
    # d'autres options :
https://docs.vagrantup.com/v2/virtualbox/configuration.html
  end
  # Setup de l'ip par rapport aux paramètres globaux
  config.vm.hostname = project_name + '.local'
  config.vm.network :private_network, ip: ip_address

  # Enfin notre configuration chef pour les recettes !
  config.vm.provision :chef_solo do |chef|
    # Ici on appelle nos cookbooks chef
[img]/assets/images/smileys/smile.png[/img]
    chef.add_recipe 'apt'
    # toujours utile car on pourrait avoir besoin de compiler une extension
php par exemple
    chef.add_recipe 'build-essential'
    chef.add_recipe 'perso::packages'
    chef.add_recipe 'perso::nodejs'
    chef.add_recipe 'perso::php'
    chef.add_recipe 'perso::nginx'
    # La c'est la configuration des paquets dont je parle plus haut
(variable node)
    chef.json = {
      :perso => {
        :packages => %W{ vim git curl httpie jq }, # Mettez ceux que vous
voulez [img]/assets/images/smileys/smile.png[/img]
        :npm_packages => %W{ gulp mocha bower pm2 }
      },
      :php => {
         # On ajoute ca au .ini (voir
https://github.com/opscode-cookbooks/php#attributes)
        :directives => {
```

```
          'date.timezone' => 'Europe/Paris'
        },
        :fpm_user => 'vagrant',
        :fpm_group => 'vagrant'
      },
      :nginx => {
        :user => 'vagrant',
        :default_site_enabled => false,
        :sendfile => 'off' # à cause d'un bug de VirtualBox
      }
    }
  end
end
```

Lancer vagrant

```
$ vagrant up
```

Quand c'est fini, ouvrir "http://192.168.33.10/" sur le navigateur, on doit voir le phpinfo() !

Pour faire des vérifications, ou rapidement vérifier qu'un changement de configuration fonctionnera au prochain vagrant provision, on peut se connecter en ssh avec :

```
$ vagrant ssh
#puis on peut vérifier que tout est bien installé par ex :
$ node -v
$ pm2 -v
```

Vagrant Host-manager

Si on veut gérer plusieurs box, où même par question de simplicité, avoir une gestion des hosts serait un plus non négligeable

```
$ vagrant plugin install vagrant-hostmanager
```

Éditer le VagrantFile afin de remplacer juste après config.berkshelf.enabled :

```
# Configuration du host manager
config.hostmanager.enabled = true
config.hostmanager.manage_host = true

# Setup de l'ip par rapport aux paramètres globaux
config.vm.hostname = project_name + '.local'
config.vm.network :private_network, ip: ip_address
config.hostmanager.aliases = [ "www." + project_name + ".local" ]
```

```
config.vm.provision :hostmanager
```

Il suffit maintenant de provisionner à nouveau la box :

```
$ vagrant provision
```

Et maintenant on a accès à la box sur “http://perso.local” !

Pour ne pas avoir à saisir de mot de passe lorsque vagrant modifie le fichier hosts utiliser cette astuce