Introduction à l'outil Chef

Laurent Dang – UMR 8628 (LMO, Orsay)

MATHEMATIQUES ORSAY

Journées Mathrice (26-28 septembre 2017) - Calais (LMPA)

Qu’est ce que Chef ?

- Chef est un système de gestion de configuration très bien fait écrit en Ruby. Il utilise un langage dédié en pure-Ruby pour l’écriture de configuration du système d’exploitation sous forme de recettes (recipes) ou de livres de recettes (cookbooks)
- Chef existe depuis 2009 et sa version actuelle est la version 12.16.14
- Il a été publié en open source Apache 2.0 et il est maintenu par Opscode.
- En février 2013, Opscode publie la version 11 de Chef comprenant une totale réécriture du code en Erlang

Qu’est ce que Chef ?

Chef n’est pas qu’un simple moteur de scripts shell : il a plutôt une approche descriptive. Vous décrivez ce que vous voulez, et Chef se charge de l’installer.

- Il est idempotent : Vous pouvez le relancer plusieurs fois sans problèmes, il ne fera que le nécessaire à chaque fois.

- Il est multi-plateforme : Chef est basé sur Ohai, qui fournit des informations sur le système d’exploitation, de manière agnostique. Chef est aussi un début d’abstraction multi platforme : il fonctionne aussi bien sous Debian que sous Fedora, ou sous Mac OS ou Windows.

Qu’est ce que Chef ?

Chef est un outil qui se positionne comme Puppet, Ansible ou CfEngine.

Au LMO, nous avons choisi Chef, parce qu’il fallait en faire un choix et que Chef nous paraissait plus extensible et plus simple que Puppet à mettre en place (car il utilise le langage Ruby au lieu d’un DSL).

Chef a été d’autre part abordé lors d’une formation « Linux avancé » que j’ai suivie fin 2015 et assurée par PLB.

Pourquoi utiliser Chef ?

Pourquoi utiliser un système de gestion de configuration de serveur ?

- pour gagner du temps
- éviter de faire des erreurs
- éviter de faire des choses redondantes
- oublier de faire des choses
- centraliser la configuration des machines sur un même serveur

Infrastructure de Chef

- Il existe deux modes de fonctionnement de Chef :

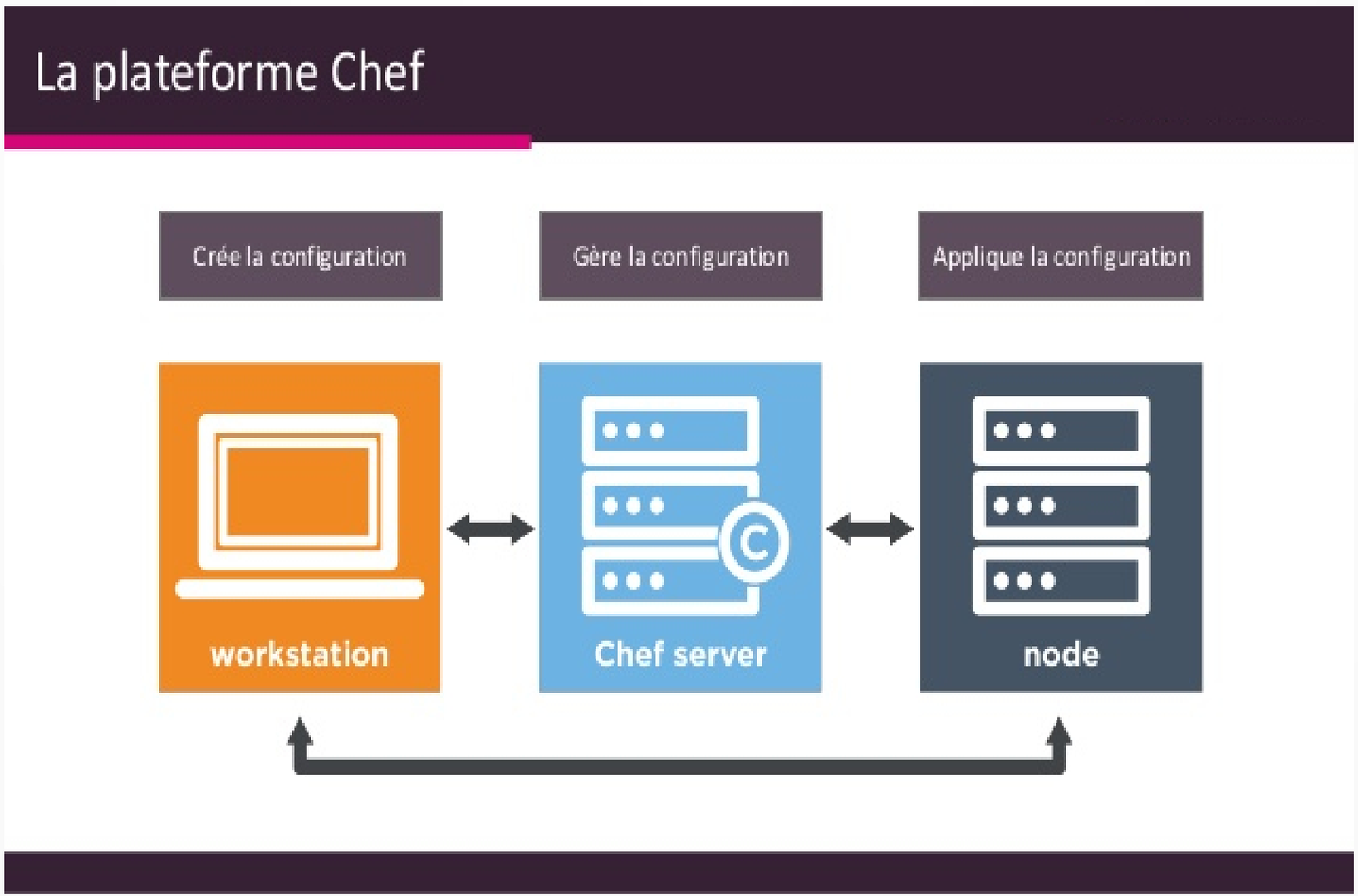
➢ En mode client-serveur

Il s'agit du mode client/serveur de Chef, dans lequel le serveur contient les recettes. Dans ce cas, il y a généralement un démon qui tourne sur chaque machine à configurer, qui se lance toutes les X minutes. Il télécharge alors les recettes depuis le « serveur Chef », pour les exécuter localement.

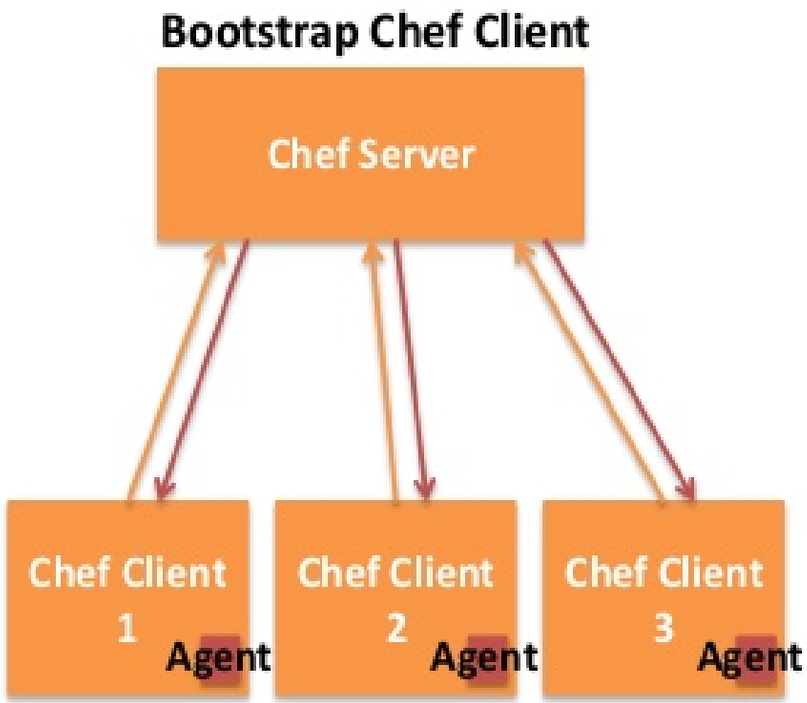
➢ En mode solo nommé « chef-solo »

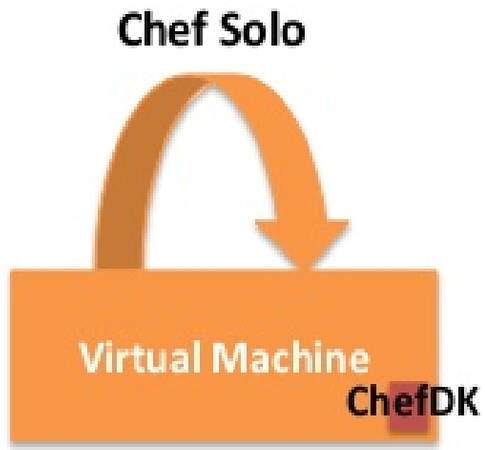
Dans ce mode, les recettes sont des fichiers locaux sur la machine. Chef-solo est autonome, ne fait pas d'appels distants sur un serveur, et est lancé par une commande shell.

Plateforme de Chef



Chef : comment ça marche ?





- Bootstrapping using **KNIFE**
- Start chef-client **RUN**

- Install Chef package
- Start chef-solo **RUN**

Chef : Installation serveur

- wget https://packages.chef.io/files/stable/chef-server/12.16.9/ubuntu/16.04/chef-server-core_12.16.14-1_amd64.deb
- dpkg -i chef-server-core_12.16.14-1_amd64.deb
- chef-server-ctl reconfigure
- chef-server-ctl user-create user_name first_name last_name email ‘PASSWORD’ --filename user.pem
- chef-server-ctl org-create short_name ‘full_organization_name’ --association_user user_name -– filename ORGANIZATION-validator.pem

Chef : Installation serveur

Exemple :

chef-server-ctl user-create admin admin admin admin@exemple.org 12345678 -f admin.pem

chef-server-ctl org-create exemple-org « Exemple.org » --association_user admin –filename exemple-org-validator.pem

Chef-manage : Installation serveur

- wget https://packages.chef.io/files/stable/chef-manage/2.5.4/ubuntu/16.04/chef-manage_2.5.4-1_amd64.deb
- dpkg -i chef-manage_2.5.4-1_amd64.deb
- chef-server-ctl reconfigure
- chef-manage-ctl reconfigure
- Version actuelle de Chef-manage : 2.5.4

Chef-manage : Installation serveur

Remarque :

Une fois cela terminé il faut modifier le fichier new.html.erb du répertoire

/opt/chef-manage/embedded/service/chef-manage/app/views/sessions pour y enlever la

partie Already have an account.

- cd /opt/chef-manage/embedded/service/chef-manage/app/views/sessions
- chef-manage-ctl graceful-kill
- chef-manage-ctl start

Chef : Mise à jour du serveur

wget https://packages.chef.io/files/stable/chef-server/12.16.14/ubuntu/16.04/chef-server-core_12.16.14-1_amd64.deb

- dpkg -i chef-server-core_12.16.14-1_amd64.deb
- chef-server-ctl upgrade
- chef-server-ctl start
- chef-server-ctl cleanup

Chef : installation du référenciel

- wget https://packages.chef.io/files/stable/chefdk/2.3.4/ubuntu/16.04/chefdk_2.3.4-1_amd64.deb
- dpkg -i chefdk_2.3.4-1_amd64.deb
- chef verify
- echo 'eval "$(chef shell-init bash)"' >> ~/.bash_profile
- source ~/.bash_profile
- mkdir ~/.chef

Chef : installation du référentiel

- scp root@server_domain_or_IP:/root/admin.pem ~/.chef
- scp root@server_domain_or_IP:/root/lmo-validator.pem ~/.chef
- nano ~/.chef/knife.rb
- knife ssl fetch

Fichier ~/.chef/knife.rb

```
current_dir = File.dirname(__FILE__)
 log_level                :info
 log_location             STDOUT
 node_name                "name_for_workstation"
 client_key               "#{current_dir}/name_of_user_key"
 validation_client_name   "organization_validator_name"
 validation_key           "#{current_dir}/organization_validator_key"
 chef_server_url
"https://server_domain_or_IP/organizations/organization_name"
 cookbook_path            ["#{current_dir}/../cookbooks"]
```

Exemple de fichier ~/.chef/knife.rb

```
log_level                :info
log_location             STDOUT
node_name                "admin"
client_key               "#{current_dir}/admin.pem"
validation_client_name   "lmo-validator"
validation_key           "#{current_dir}/exemple-org-validator.pem"
chef_server_url
"https://server_domain_or_IP/organizations/exemple-org"
cookbook_path            ["#{current_dir}/../cookbooks"]
```

Qu’est ce qu’une recette ?

C’est une description de plusieurs ressources qui forment un ensemble cohérent. Concrètement, une recette, c’est un ensemble de fichiers écrits en Ruby, organisés d’une certaine façon. Une recette est un dossier contenant les sous dossiers suivants :

- attributes : des paramètres de configuration pour la recette
- recipes : les recettes
- templates : les modèles de fichiers (notamment de configuration) que nous allons déployer
- definitions, libraries, ressources, providers : des dossiers pour étendre les recettes, créer des nouvelles ressources

- Chaque fichier Ruby (.rb) du dossier recipes est une recette décrivant l’installation d’un logiciel, le fichier **default.rb** étant la recette par défaut.

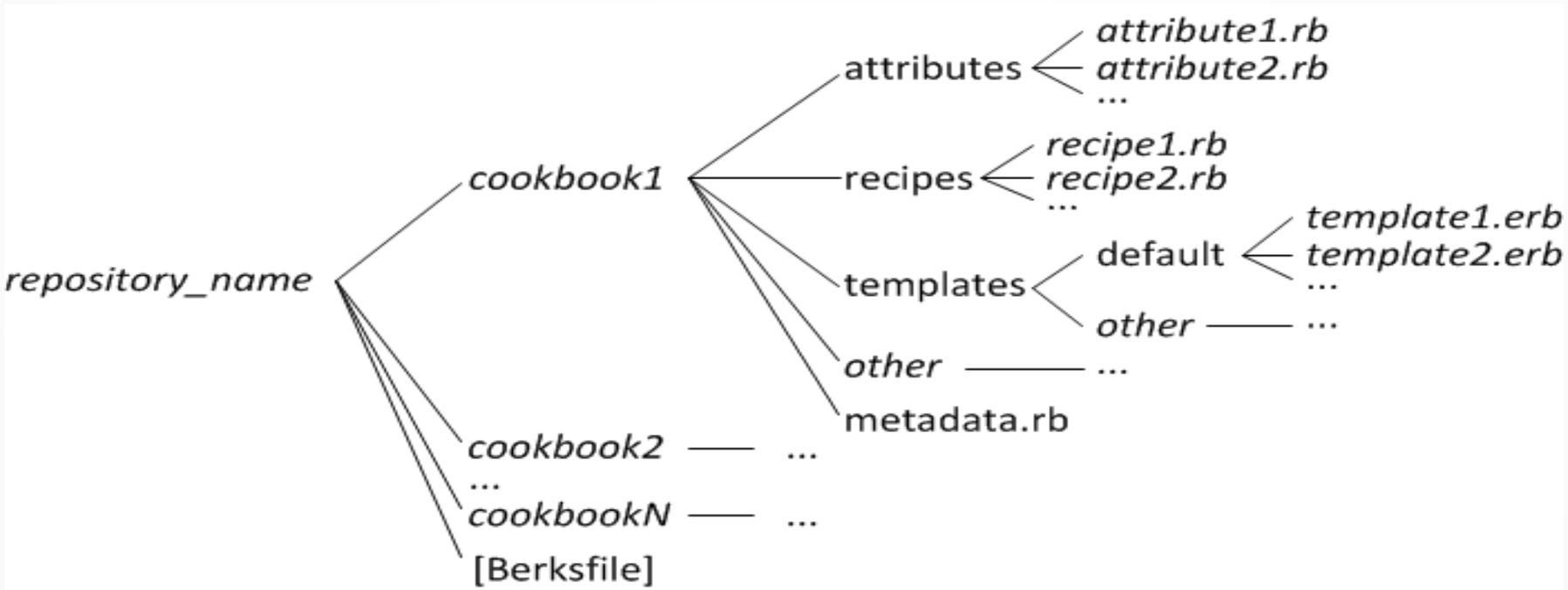
Qu’est ce qu’un rôle ?

On peut par exemple utiliser les rôles pour surcharger les configurations. Un rôle est un fichier Json qui contient :

- Une liste de recettes ou d’autres rôles à exécuter
- Une surcharge de fichier de configuration

Structure d'un cookbook : livre de recettes Chef

- La structure d'un livre de recettes Chef (cookbook) est la suivante :



- Pour créer un cookbook, il faut utiliser la commande :

knife cookbook create nom_du_cookbook

Structure d'un cookbook : livre de recettes Chef

- Le fichier **metadata.rb** est un fichier de métadonnées qui permet de renseigner le cookbook et de savoir le numéro de version de celui ci quand on le met à jour sur le serveur.
- Ce fichier est crée automatiquement quand on utilise la commande **knife cookbook create**
- Il se trouve au premier niveau de la structure d'un répertoire de cookbook
- Il peut être édité avec un éditeur classique tel que vi,nano
- Il est recopié sur le serveur Chef à chaque nouvelle utilisation de la commande **knife cookbook upload nom_du_cookbook**
- Il permet de renseigner les dépendances du cookbook. On utilise pour cela le mot clé **depends** dans ce fichier

Structure d'un cookbook: exemple

- Le fichier **metadata.rb** du cookbook « installation_logiciels » se présente comme suit :

```
name             'installation_logiciels'
maintainer       'LMO'
maintainer_email 'svp@math.u-psud.fr'
license          'All rights reserved'
description      'Installs/Configures installation_logiciels'
long_description 'Installation des logiciels sur les
machines fixes et portables du personnel labo'
version          '0.1.0'
```

Commandes Chef utiles

knife cookbook upload install_laptop

permet de mettre à jour le livre de recettes install_laptop depuis le référenciel Chef suite à une modification sur le serveur Chef

knife bootstrap @IP_du_noeud –ssh-user login --sudo --node-ssl-verify-mode none --node-name nom_du_client_Chef --run-list 'recipe[installation_logiciels::install_laptop]’

permet depuis le référentiel Chef de rajouter un nouveau client et d’appliquer la recette install_laptop du livre de recette installation_logiciels au « noeud » défini par son adresse IP et sur lequel on se connecte avec le compte login en tant que sudoers

Utilisation de Chef : dans quel cadre ?

Un grand nombre d'actions sont ainsi possibles avec les outils de déploiement de configuration comme Chef:

- Installer un paquet système (RPM, deb, pkg…)
- Installer un fichier à partir du référentiel, en utilisant éventuellement un système de gabarits pour y placer des données dépendantes de chaque machine (exemple : l'adresse IP d'écoute du service)
- Mettre en place ou modifier une tâche planifiée
- Créer un répertoire
- Supprimer un répertoire/fichier
- S'assurer que tel ou tel fichier/répertoire dispose de droits prévus
- Changer le montage d'un système de fichiers (local ou réseau) ...

Chef :référence,documentation

https://docs.chef.io/ : Documentation officielle sur Chef

https://supermarket.chef.io/ : Bibliothèque officielle de recettes Chef

Quelques exemples de recettes Chef

Exemples de recettes Chef : installation simple d'un package

```
package 'filezilla' do
        action [:install]
end

 package 'gnome' do
     action [:install]
end
```

Exemples de recettes Chef : installation de plusieurs packages (notion de boucle)

```
for p in ["geany", "texmaker", "unrar", "okular-extra
  backends", "valgrind", "check", "qpdf"] do
  package p do
            action [:install]
  end
end
```

Exemples de recettes Chef : utilisation des templates et des variables

```
template '/etc/ntp.conf' do
    source 'ntp.erb'
        variables( :ntp_server => "ntp.mon_domaine.fr" )
        notifies :restart, "service[ntp]"
end
```

Fichier template ntp.erb dans templates/default:

Ce fichier est défini à partir d'un fichier ntp.conf standard dans lequel on définit la variable ntp_server comme suit :

```
server <%= @ntp_server %> iburst
```

Exemples de recettes Chef : utilisation des attributs et des templates

Fichier serveurs.rb dans attributes/:

Ce fichier est celui dans lequel on définit les adresses IP des serveur1 **(A.B.C.D)** et serveur2 **(E.F.G.H)** qui nous servent de serveurs DNS et on le définit comme suit :

default['dns']['serveur1']='A.B.C.D'
default['dns']['serveur2']='E.F.G.H'

Exemples de recettes Chef : utilisation des attributs et des templates

Fichier template resolv.conf.erb dans templates/default:

Ce fichier est défini à partir d'un fichier resolv.conf standard dans lequel on utilise les attributs **default['dns']['serveur1']** et **default['dns']['serveur2']** définis plus haut :

```
domain math.u-psud.fr
search math.u-psud.fr
nameserver <%= @node[:dns][:serveur1] %>
nameserver <%= @node[:dns][:serveur2] %>
```

Exemples de recettes Chef : utilisation des attributs et des templates

```
template '/etc/resolv.conf' do
        source 'resolv.conf.erb'
            mode  0644
            owner "root"
            group  "root"
end
```

Exemples de recettes Chef : utilisation de Bash

```
script 'Installation de Matlab' do
  interpreter 'bash'
  user 'root'
  cwd '/tmp'
  code <<-EOH
#!/bin/bash
  cd /usr/local/
  rm -rf MATLAB
  tar xzfp matlab.tgz
  rm matlab.tgz
EOH
end
```

Exemples de recettes Chef : pour aller plus loin – installation de Chocolatey

```
unless node['platform_family'] == 'windows'
  return "Chocolatey install not supported on #{node['platform_family']}"
end
Chef::Resource.send(:include, Chocolatey::Helpers)
install_ps1 = File.join(Chef::Config['file_cache_path'], 'chocolatey-install.ps1')
cookbook_file install_ps1 do
  action :create
  backup false
 source 'install.ps1'
end
```

Exemples de recettes Chef : pour aller plus loin – installation de Chocolatey

```
powershell_script 'Install Chocolatey' do
    environment node  ['chocolatey']['install_vars']
    cwd Chef::Config['file_cache_path']
    code install_ps1
    not_if { chocolatey_installed? && (node['chocolatey']['upgrade'] == false) }
end
```

Exemples de recettes Chef : pour aller encore plus loin

```
pkg_resource = case
node['platform_family']
  when "debian"
    :dpkg_package
  when "fedora", "rhel", "amazon"
    :rpm_package
end

pkg_path = ( pkg_resource ==
:dpkg_package ) ? "/tmp/foo.deb" :
"/tmp/foo.rpm"

declare_resource(pkg_resource,
pkg_path) do
  action :install
end
```

```
 package "apache2" do
case node["platform"]
when "centos","redhat",
"fedora","suse"

package_name "httpd"
when "debian","ubuntu"
  package_name "apache2"
when "arch"
  package_name "apache"
end
action :install
end
```

Chef : exemple complet pour déployer memcached avec la notion de rôle

Le fichier **memcached.rb** défini dans **memcached/recipes** se présente comme suit:

```
package 'memcached' do
    action [:install]
end

template "/etc/memcached.conf" do
  source "memcached.conf.erb"
  variables :port => node.memcached.port
end
```

Chef : exemple complet pour déployer memcached avec la notion de rôle

Le fichier ERB **memcached.conf.erb** est défini dans **memcached/templates/default** comme suit:

```
# memcached default config file

-d

logfile /var/log/memcached.log

-m 296

-p <%= @port %>

-u nobody

-l 127.0.0.1

-P /var/run/memcached.pid
```

En plus de ces 2 fichiers, il en faut un troisième pour les attributs **default.rb**, dans **memcached/attributes** :

```
default[:memcached][:port] = 11211
```

Chef : exemple complet pour déployer memcached avec la notion de rôle

```
{
"name":"cache",
"default_attributes":{
    "memcached": { "port" : 11212 }
  },
"json_class":"Chef::Role",
"env_run_lists":{},
"run_list":
["recipe[memcached::memcached]",
"recipe[munin::munin]"],
"description":"",
"chef_type":"role",
"override_attributes":{}
}
```

Ce fichier cache.json déclare un rôle cache, pour lequel il faut exécuter les recettes memcached et munin, et où le port de memcached sera 11212.

Pour l'exécuter sur la machine cible, si le fichier json se trouve dans le répertoire /etc/chef/, on utilise la commande :

chef-client -j /etc/chef/cache.json

Questions ?