CHEF: chef-repo

Table of Contents

- CHEF: chef-repo
- Structure du répertoire
  - .chef/
  - cookbooks/
  - data_bags/
  - environnements/
  - roles/
- Fichier chefignore
  - Structure
  - Exemples
- Partage de repo avec beaucoup d'utilisateurs

Le chef-repo est un répertoire sur votre poste de travail qui stocke:

- Livres de recettes (y compris recettes, attributs, ressources personnalisées, bibliothèques et modèles)
- Rôles
- Sacs de données
- Environnements

Le répertoire chef-repo doit être synchronisé avec un système de contrôle de version, tel que git. Toutes les données du chef-repo doivent être traitées comme du code source.

Knife est utilisé pour télécharger des données sur le serveur Chef à partir du répertoire chef-repo. Une fois téléchargées, le chef-client utilise ces données pour gérer tous les noeuds enregistrés auprès du serveur Chef et pour s'assurer que les livres de recettes, environnements, rôles et autres paramètres corrects sont appliqués correctement aux noeuds.

Structure du répertoire

Chef-repo contient plusieurs répertoires, chacun avec un fichier README décrivant son objectif et la façon de l'utiliser pour gérer des systèmes.

La commande suivenate permet de créer un chef-repo :

```
$ chef generate repo REPO_NAME
```

Les sous-répertoires du chef-repo sont:

Répertoire	Description
.chef/	Répertoire caché utilisé pour stocker les fichiers de clé et éventuellement un fichier config.rb.
cookbooks/	Contient des livres de cuisine téléchargés du supermarché Chef ou créés localement.
data_bags/	Stocke les sacs de données (et leurs éléments) au format JSON (.json).
environments/	Enregistre l'environnement en Ruby (.rb) ou JSON (.json).
roles/	Stocke les rôles dans Ruby (.rb) ou JSON (.json).

.chef/

Le répertoire .chef est un répertoire caché utilisé pour stocker les fichiers de clé et éventuellement un fichier config.rb.

cookbooks/

Le répertoire cookbooks/ est utilisé pour stocker les livres de recettes utilisés par le chef-client lors de la configuration des différents systèmes de l'organisation. Ce répertoire contient les livres de recettes utilisés pour configurer les systèmes de l'infrastructure. Chaque livre de recettes peut être configuré pour contenir des données de copyright, de courrier électronique et de licence spécifiques à ces livres.

data_bags/

Le répertoire data_bags/ est utilisé pour stocker tous les sacs de données existants pour une organisation. Chaque sous-répertoire correspond à un seul sac de données sur le serveur Chef et contient un fichier JSON pour chaque élément de sac de données. Si un sous-répertoire n'existe pas, le créer à l'aide de commandes SSL. Une fois créé, un article de sac de données peut être téléchargé sur le serveur Chef.

environnements/

Le répertoire environments/ sert à stocker les fichiers qui définissent les environnements disponibles pour le serveur Chef. Les fichiers d'environnement peuvent être des fichiers Ruby DSL (.rb) ou JSON (.json). Utiliser knife pour installer des fichiers d’environnement sur le serveur Chef.

roles/

Le répertoire roles/ est utilisé pour stocker les fichiers qui définissent les rôles disponibles pour le serveur Chef. Les fichiers de rôles peuvent être des fichiers Ruby DSL (.rb) ou JSON (.json). Utiliser Knife pour installer les fichiers de rôle sur le serveur Chef.

Fichier chefignore

Le fichier chefignore est utilisé pour indiquer à knief quels fichiers de livre de recettes du chef-repo doivent être ignorés lors du téléchargement de données sur le serveur Chef. Le type de données à ignorer comprend les fichiers d'échange, les données de contrôle de version, les données de sortie, etc.

Structure

Le fichier chefignore utilise la syntaxe File.fnmatch Ruby pour définir les modèles de File.fnmatch à l'aide des caractères joker * , ** et ? .

- Un motif est relatif à la racine du livre de recettes
- Un modèle peut contenir des noms de répertoire relatifs
- Un modèle peut correspondre à tous les fichiers d'un répertoire

Le fichier chefignore peut être situé dans n'importe quel sous-répertoire d'un chef-repo: /, /cookbooks, /cookbooks/COOKBOOK_NAME/, roles, etc. Il devrait contenir des sections similaires à ce qui suit:

```
# section
* ignore_pattern

# section
ignore_pattern *

# section
** ignore_pattern

# section
ignore_pattern **

# section
? ignore_pattern

# section
ignore_pattern?
```

Exemples

Les exemples suivants montrent comment ajouter des entrées au fichier chefignore .

Ignorer les fichiers de swap des éditeurs

De nombreux éditeurs de texte laissent des fichiers derrière eux. Pour empêcher le téléchargement

de ces fichiers sur le serveur Chef, ajoutez une entrée au fichier chefignore.

- Pour Emacs, faire quelque chose comme:

```
*~
```

- Pour vim, faire quelque chose comme:

```
*.sw[az]
```

Ignorer les données de niveau supérieur de Subversion

Si Subversion est utilisé en tant qu'application de contrôle de source de version, il est important de ne pas télécharger certains fichiers que Subversion utilise pour conserver l'historique de version de chaque fichier. En effet, le chef-client ne l'utilisera jamais lors de la configuration des nœuds. De plus, la quantité de données d'un téléchargement comprenant des données Subversion de niveau supérieur peut être importante.

Pour empêcher le téléchargement de données Subversion de niveau supérieur, ajouter ce qui suit dans le fichier chefignore:

```
*/.svn/*
```

Pour vérifier que les données Subversion de niveau supérieur ne seront pas téléchargées sur le serveur Chef, utiliser knife et exécutee une commande similaire à:

```
$ knife cookbook show name_of_cookbook cookbook_version |  grep .svn
```

Ignorer tous les fichiers d'un répertoire

Le fichier chefignore peut être utilisé pour ignorer tous les fichiers d'un répertoire. Par exemple:

```
files/default/sous-répertoire/*
```

ou:

```
files/default/sous-répertoire/**
```

Partage de repo avec beaucoup d'utilisateurs

La configuration de config.rb peut inclure du code Ruby arbitraire pour étendre la configuration au-delà des valeurs statiques. Ceci peut être utilisé pour charger des variables d'environnement à partir du poste de travail. Cela permet d'écrire un seul fichier config.rb pouvant être utilisé par tous les

utilisateurs d'une organisation. Ce fichier unique peut également être archivé dans le chef-repo, ce qui permet aux utilisateurs de charger différents fichiers config.rb en fonction du chef-repo à partir duquel ils exécutent les commandes. Cela peut être particulièrement utile lorsque chaque chef-repo pointe vers un serveur ou une organisation de chef différents.

Exemple config.rb:

```
current_dir = File.dirname(__FILE__)
 user = ENV['OPSCODE_USER'] || ENV['USER']
 node_name                user
 client_key               "#{ENV['HOME']}/chef-repo/.chef/#{user}.pem"
 validation_client_name   "#{ENV['ORGNAME']}-validator"
 validation_key           "#{ENV['HOME']}/chef-repo/.chef/#{ENV['ORGNAME']}-
validator.pem"
 chef_server_url
"https://api.opscode.com/organizations/#{ENV['ORGNAME']}"
 syntax_check_cache_path  "#{ENV['HOME']}/chef-
repo/.chef/syntax_check_cache"
 cookbook_path            ["#{current_dir}/../cookbooks"]
 cookbook_copyright       "Your Company, Inc."
 cookbook_license         "Apache-2.0"
 cookbook_email           "cookbooks@yourcompany.com"

 # Amazon AWS
 knife[:aws_access_key_id] = ENV['AWS_ACCESS_KEY_ID']
 knife[:aws_secret_access_key] = ENV['AWS_SECRET_ACCESS_KEY']
```