knife-solo

Table of Contents

Description

knife-solo ajoute des commandes qui visent à rendre le travail avec **chef-solo** aussi puissant que **chef-server**. Il ajoute actuellement 5 sous-commandes à **knife** :

- **knife solo init** est utilisé pour créer une nouvelle structure de répertoires qui correspond à la structure standard de **Chef** et peut être utilisée pour créer et stocker des recettes.
- **knife solo prepare** installe **Chef** sur un hôte donné. Il est structuré pour détecter automatiquement le système d'exploitation cible et modifier le processus d'installation en conséquence.
- **knife solo cook** télécharge le dépôt actuel sur l'hôte cible et exécute **chef-solo** sur cet hôte.
- **knife solo bootstrap** combine les deux précédents (prépare et cook. **knife-solo** ajoute également l'option de ligne de commande **–solo** et le paramètre de configuration **+knife+** au bootstrap de **knife** qui peut être utilisé pour déclencher **knife solo bootstrap** au lieu du bootstrap chef-client basé sur un modèle normal.
- **knife solo clean** supprime le dépôt téléchargé de l'hôte cible.

Installation de knife-solo

Installation avec ruby gem standard:

```
gem install knife-solo

# ou si on utilise ChefDK
chef gem install knife-solo
```

Installation à partir de la source git:

```
```

```
bundle && bundle exec rake install
```

knife-solo intègre le support automatique à **Berkshelf** et **Librarian-Chef** pour gérer des cookbooks prêts à l'emploi, si l'un des deux gems est installé (si les deux sont installés, **knife-solo** utilisera par défaut**Berkshelf**/.
Pendant **knife solo init**, un fichier de configuration approprié est généré pour l'une ou l'autre des gems. Ensuite, pendant **knife solo cook**, on exécute l'étape d'installation pour le fichier de configuration qui se trouve dans le répertoire.

Les deux commandes acceptent les indicateurs d'option pour désactiver cette fonctionnalité si nécessaire **–no-berkshelf** et/ou **–no-librarian**.

La commande **knife solo init** propose également des indicateurs d'activation pour générer des fichiers de configuration, que le support soit installé ou non.

Une erreur courante est qu'on peut avoir installé **Berkshelf** ou **Librarian-Chef** sans le savoir. Cela générera une dossier configurée pour les utiliser, ce qui n'était peut-être pas voulu. Une fois le fichier de configuration disponible, le répertoire des cookbooks sera réservé aux cookbooks résolus via l'un de ces outils. Tous les cookbooks qu'on y crée seront supprimés lorsqu'on exécute **knife solo cook**.
Il est préférable d'utiliser `site-cookbooks` pour les cookbooks personnalisés ou (mieux encore) leur donner leurs propres référentiels git qui sont ensuite inclus à l'aide de **Berkshelf** ou **Librarian-Chef**.

Commandes knife-solo

Commande init

La commande **init** prend simplement un nom de répertoire pour stocker la structure du répertoire. Utiliser “.” pour initialiser le répertoire courant.

```
knife solo init mychefrepo
```

Actuellement, la structure du répertoire ressemble à ceci, mais pourrait changer au fur et à mesure du développement.

```
mychefrepo/
├── .chef
│   └── knife.rb
├── cookbooks
├── data_bags
```

```
├── nodes
├── roles
└── site-cookbooks
```

Commande prepare

La commande **prepare** prend un argument d'hôte de style ssh comme suit :

```
knife solo prepare ubuntu@10.0.0.201
```

Il recherchera les informations SSH à partir de `~/.ssh/config` ou dans le fichier spécifié par **-F**. On peut également transmettre des informations de port **-p**, des informations d'identité **-i** ou un mot de passe **-P**. Il utilisera **sudo** pour exécuter certaines de ces commandes et demandera le mot de passe s'il n'est pas fourni sur la ligne de commande.

Cette commande fera de son mieux pour détecter et installer **Chef Solo** sur le système d'exploitation cible, en utilisant le programme d'installation **Opscode** dans la mesure du possible.

Si on a besoin d'un comportement spécifique, on peut revenir à une commande d'amorçage de knife avec une liste d'exécution vide en utilisant ce qui suit :

```
knife bootstrap --template-file bootstrap.centos.erb -u root 172.16.144.132
echo '{"run_list":[]}' > nodes/172.16.144.132.json
```

Les modèles de bootstrap sont assez simples, comme indiqué dans cet exemple:

```
bash -c '
<%= "export http_proxy=\"#{knife_config[:bootstrap_proxy]}\"" if
knife_config[:bootstrap_proxy] -%>

yum install -y wget
wget <%= "--proxy=on " if knife_config[:bootstrap_proxy]
%>http://rbel.co/rbel5
rpm -Uvh rbel5

yum install -y rubygem-chef
'
```

Commande Cook

La commande **cook** prend également un argument d'hôte de style ssh :

```
knife solo cook ubuntu@10.0.0.201
```

La commande **cook** télécharge le référentiel actuel sur le serveur et exécute **chef-solo** sur ce

serveur. Si on ne spécifie qu'un seul argument, il recherchera une configuration de nœud dans `nodes/<hostname>.json`. Si on souhaite spécifier une configuration de nœud, on peut indiquer le chemin d'accès au fichier comme deuxième argument.

Cela télécharge tous les cookbooks en plus d'un correctif qui permet d'utiliser le contenu du dossier `data_bags` en lecture seule.

Cela prend également en charge les sacs de données cryptés. Pour les utiliser, définir le chemin vers la clé avec **encryption*data*bag_secret** dans `.chef/knife.rb`.

Les commandes de knife intégrées pour travailler avec des sacs de données ne fonctionnent pas bien sans un serveur Chef, il est donc recommandé d'utiliser la gem `knife-solo_data_bag`, pour fournir des versions “solo” de toutes les commandes de sac de données typiques. La structure par défaut générée par **knife solo init** doit être compatible avec toutes les opérations répertoriées dans la documentation de cette gem.

Si on veut exécuter **chef-solo** en mode hérité, vous pouvez utiliser l'option **–legacy-mode** ou mettre **+solo*legacy*mode true+** dans `.chef/knife.rb`.

Commande bootstrap

La commande **bootstrap** prend les mêmes arguments et la plupart des options que **prepare** et **cook**:

```
knife solo bootstrap ubuntu@10.0.0.201
```

Il appelle d'abord **+knife solo prepare+** puis **+knife solo cook+** avec les arguments et les options spécifiés.

knife-solo s'intègre également à **knife bootstrap** en y ajoutant l'option de ligne de commande **–solo** et le paramètre de configuration **+knife+**. Lorsqu'il est demandé, **knife solo bootstrap** est utilisé à la place du **bootstrap chef-client** basé sur un modèle normal. Ceci est particulièrement utile avec d'autres plugins de **knife** comme **knife-ec2** qui invoquent **knife bootstrap** après avoir créé une instance de serveur. Même si ces plugins n'ont pas l'option **–solo**, on peut mettre **knife[:solo] = true** dans `knife.rb`.

Commande clean

La commande **clean** prend les mêmes arguments que **prepare** et **cook**:

```
knife solo clean ubuntu@10.0.0.201
```

La commande **clean** supprime complètement le référentiel téléchargé de l'hôte cible. Cela améliore la sécurité car les mots de passe, etc. ne sont pas oubliés sur cet hôte.