chef-solo

Table of Contents

chef-solo est une commande qui exécute **Chef Infra Client** d'une manière qui ne nécessite pas que le serveur **Chef Infra** fasse converger les cookbooks. **chef-solo** utilise le mode **local Chef** de **Chef Infra Client** et ne prend pas en charge les fonctionnalités suivantes présentes dans les configurations **Chef Infra Client/serveur** :

- Distribution centralisée de cookbooks
- Une API centralisée qui interagit avec et intègre les composants de l'infrastructure
- Authentification ou autorisation

chef-solo peut être exécuté en tant que démon.

L'exécutable **chef-solo** est exécuté comme un outil de ligne de commande.

Référentiel

Cookbooks

chef-solo prend en charge deux emplacements à partir desquels les cookbooks peuvent être exécutés :

- Un répertoire local.
- Une URL à laquelle se trouve une archive tar.gz.

L'utilisation d'une archive tar.gz est l'approche la plus courante, mais nécessite que les cookbooks soient ajoutés à une archive. Par exemple:

```
tar zcvf chef-solo.tar.gz ./cookbooks
```

Si plusieurs répertoires de cookbooks sont utilisés, **chef-solo** s'attend à ce que l'archive tar.gz ait une structure de répertoires similaire à la suivante :

```
cookbooks/
  |---- cbname1/
    |--attributes/ ... etc
  ...
  |---- cbname2/
    |--attributes/
```

La variable **cookbook_path** dans le fichier `solo.rb` doit inclure les deux répertoires. Par exemple:

```
tar zcvf chef-solo.tar.gz ./cookbooks ./site-cookbooks
```

Lorsque l'archive tar.gz contient tous les cookbooks requis par **chef-solo**, il faut la télécharger sur le serveur Web à partir duquel **chef-solo** accédera à l'archive.

Nœuds

Contrairement à **Chef Infra Client**, où l'objet nœud est stocké sur le serveur Chef Infra, **chef-solo** stocke ses objets nœud sous forme de fichiers JSON sur le disque local. Par défaut, **chef-solo** stocke ces fichiers dans un dossier de nœuds dans le même répertoire que le répertoire de cookbooks. On peut contrôler l'emplacement de ce répertoire à l'aide de la valeur **node_path** dans le fichier de configuration.

Attributs

chef-solo n'interagit pas avec le serveur Chef Infra. Par conséquent, les attributs spécifiques au nœud doivent se trouver dans un fichier JSON sur le système cible, un emplacement distant (comme Amazon Simple Storage Service (S3)) ou un serveur Web sur le réseau local.

Le fichier JSON doit également spécifier les recettes qui font partie de la liste d'exécution. Par exemple:

```
{
  "resolver": {
    "nameservers": [ "10.0.0.1" ],
    "search":"int.example.com"
  },
  "run_list": [ "recipe[resolver]" ]
}
```

Sacs de données

Un sac de données est défini à l'aide de JSON. **chef-solo** recherchera les sacs de données dans `/var/chef/data_bags`, mais cet emplacement peut être modifié en modifiant le paramètre dans `solo.rb`. Par exemple, le paramètre suivant dans solo.rb :

```
data_bag_path '/var/chef-solo/data_bags'
```

Créer un sac de données en créant des dossiers. Par exemple:

```
mkdir /var/chef-solo/data_bags
```

et:

```
mkdir /var/chef-solo/data_bags/admins
```

puis créer un fichier JSON à cet emplacement :

```
{
   "id": "ITEM_NAME"
}
```

où le nom du fichier est ITEM_NAME, par exemple :

```
/var/chef-solo/data_bags/admins/ITEM_NAME.json
```

Rôles

Un rôle est défini à l'aide de **JSON** ou de **Ruby DSL**. **chef-solo** recherchera les rôles dans `/var/chef/roles`, mais cet emplacement peut être modifié en modifiant le paramètre de **role_path** dans `solo.rb`. Par exemple, le paramètre suivant dans `solo.rb`:

```
role_path '/var/chef-solo/roles'
```

Les données de rôle ressemblent à ce qui suit en **JSON**:

```
{
  "name": "test",
  "default_attributes": { },
  "override_attributes": { },
  "json_class": "Chef::Role",
  "description": "This is just a test role, no big deal.",
  "chef_type": "role",
```

```
  "run_list": [ "recipe[test]" ]
}
```

et comme ce qui suit en **Ruby DSL**:

```
name 'test'
description 'This is just a test role, no big deal.'
run_list 'recipe[test]'
```

et enfin, les données **JSON** transmises à **chef-solo**:

```
{ 'run_list': 'role[test]' }
```

Environnements

Un environnement est défini à l'aide de JSON ou de Ruby DSL. **chef-solo** recherchera les environnements dans `/var/chef/environments`, mais cet emplacement peut être modifié en modifiant le paramètre pour **environment_path** dans `solo.rb`. Par exemple, le paramètre suivant dans `solo.rb`:

```
environment_path '/var/chef-solo/environments'
```

Les données d'environnement ressemblent à ce qui suit dans **JSON**:

```
{
  "name": "dev",
  "default_attributes": {
    "apache2": {
      "listen_ports": [
        "80",
        "443"
      ]
    }
  },
  "json_class": "Chef::Environment",
    "description": "",
    "cookbook_versions": {
    "couchdb": "= 11.0.0"
  },
  "chef_type": "environment"
  }
```

et comme ce qui suit dans **Ruby DSL**:

```
name 'environment_name'
```

```
description 'environment_description'
cookbook OR cookbook_versions  'cookbook' OR 'cookbook' =>
'cookbook_version'
default_attributes 'node' => { 'attribute' => %w(value value etc.) }
override_attributes 'node' => { 'attribute' => %w(value value etc.) }
```

Commande chef-solo

Cette commande a la syntaxe suivante :

```
chef-solo VALEUR DE L'OPTION VALEUR DE L'OPTION ...
```

Options

Option	Libellé
-c CONFIG, **–config CONFIG**	Le fichier de configuration à utiliser.
-d, –daemonize	Exécuter en tant que démon. Cette option ne peut pas être utilisée dans la même commande avec l'option **–[no-]fork**. Cette option n'est disponible que sur les machines qui s'exécutent dans des environnements UNIX ou Linux.
-E ENVIRONMENT*NAME*, *–environnement ENVIRONMENT*NAME	Le nom de l'environnement.
-f, -[no-]fork	Contient **Chef Infra Client** s'exécute dans un processus secondaire avec une RAM dédiée. Lorsqu'une exécution de **Chef Infra Client** est terminée, la RAM est renvoyée au processus maître. Cette option permet de garantir qu'un client Chef Infra utilise une quantité constante de RAM au fil du temps, car le processus principal n'exécute pas de recettes. Cette option permet également d'éviter les fuites de mémoire telles que celles qui peuvent être introduites par le code contenu dans un cookbooks mal conçu. Utiliser **–no-fork** pour désactiver l'exécution de **Chef Infra Client** dans le nœud de fourche. Valeur par défaut : **–fork**. Cette option ne peut pas être utilisée dans la même commande avec les options **–daemonize** et **–interval**.

Option	Libellé
-F FORMAT, –format FORMAT	Le format de sortie : **doc**(par défaut) ou **min**. Utiliser **doc** pour imprimer la progression d'une exécution de **Chef Infra Client** à l'aide de chaînes complètes qui affichent un résumé des mises à jour au fur et à mesure qu'elles se produisent. Utiliser **min** pour imprimer la progression d'une exécution de **Chef Infra Client** à l'aide de caractères uniques. Un résumé des mises à jour est imprimé à la fin d'une exécution de **Chef Infra Client**, un point (.) est imprimé pour les événements qui n'ont pas d'informations d'état significatives, comme le chargement d'un fichier ou la synchronisation d'un cookbooks. Pour les ressources - un point (.) est imprimé lorsque la ressource est à jour - un **S** est imprimé lorsque la ressource est ignorée par **not_if** ou **only_if** - un **U** est imprimé lorsque la ressource est mise à jour. D'autres options de formatage sont disponibles lorsque ces formateurs sont configurés dans le fichier client.rb à l'aide de l'option **add_formatter**.
–force-formatter	Afficher la sortie du formateur au lieu de la sortie de l'enregistreur.
–force-logger	Afficher la sortie de l'enregistreur au lieu de la sortie du formateur.
-g GROUPE, –group GROUPE	Nom du groupe propriétaire d'un processus. Ceci est requis lors du démarrage de tout exécutable en tant que démon.
-h, –help	Afficher l'aide de la commande.
-i SECONDES, –intervalle SECONDES	La fréquence (en secondes) à laquelle **Chef Infra Client** s'exécute. Lors de l'exécution de **Chef Infra Client** à intervalles réguliers, appliquer les valeurs **–splay** et **–interval** avant l'exécution de **Chef Infra Client**. Cette option ne peut pas être utilisée dans la même commande avec l'option **–[no-]fork**.

-j CHEMIN, –json-attributs CHEMIN	Chemin d'accès à un fichier contenant des données JSON. Utiliser cette option pour définir un objet **run_list**. Par exemple, un fichier JSON similaire à :
	
	`"run_list": [\"recipe[base]",`
	`"recipe[foo]",`
	`"recipe[bar]",`
	`"role[webserver]"`
	`],`
	
	peut être utilisé en exécutant `chef-client -j path/to/file.json`
	
	Dans certaines situations, cette option peut être utilisée pour mettre à jour les attributs normaux.
	
	Tout autre type d'attribut contenu dans ce fichier JSON sera traité comme un attribut normal. La définition d'attributs à d'autres niveaux de priorité n'est pas possible.

Option	Libellé
-l NIVEAU, –log_level NIVEAU	Niveau de journalisation à stocker dans un fichier journal. Niveaux possibles : **auto** (par défaut), **debug**, **error**, **fatal**, **info**, **trace** ou **warn**. Valeur par défaut : **warn** (lorsqu'un terminal est disponible) ou **info** (lorsqu'un terminal n'est pas disponible).
-L LOGLOCATION, –logfile c	Emplacement du fichier journal. Ceci est recommandé lors du démarrage de tout exécutable en tant que démon.
–legacy-mode	Faire en sorte que Chef Infra Client utilise le mode **chef-solo** d'origine au lieu du mode chef local. Ceci n'est pas recommandé.
–minimal-ohai	Exécuter les plugins **Ohai** pour la détection de nom et la sélection de ressources/fournisseurs et aucun autre plugin **Ohai**. Définir sur vrai pendant les tests d'intégration pour accélérer les cycles de test.
–[no-]color	Afficher la sortie en couleur. Paramètre par défaut : **–color**.
-N NODE*NAME, –node-name NODE*NAME	L'identifiant unique du nœud.
-o RUN*LIST*ITEM, –override-runlist RUN*LIST*ITEM	Remplacer la liste d'exécution actuelle par les éléments spécifiés.
-r RECIPE*URL, –recipe-url RECIPE*URL	L'URL du fichier tar.gz du livre de recettes distant que l'on souhaite télécharger. Dans Chef Infra Client 14, la forme courte -r sera supprimée, car elle est en conflit avec la possibilité de spécifier une liste d'exécution.
–run-lock-timeout SECONDES	La durée (en secondes) d'attente pour la suppression d'un fichier de verrouillage de Chef Infra Client. Valeur par défaut : non définie (indéfinie). Définir sur 0 pour provoquer la fermeture immédiate d'un deuxième client Chef Infra.
-s SECONDES, –splay SECONDES	Un nombre aléatoire entre zéro et **splay** qui est ajouté à l'intervalle. Utiliser **splay** pour aider à équilibrer la charge sur le serveur Chef Infra en s'assurant que de nombreuses exécutions du client Chef Infra ne se produisent pas au même intervalle. Lors de l'exécution de **Chef Infra Client** à intervalles réguliers, appliquer les valeurs **–splay** et **–interval** avant l'exécution de **Chef Infra Client**.
-u UTILISATEUR, –user UTILISATEUR	L'utilisateur qui possède un processus. Ceci est requis lors du démarrage de tout exécutable en tant que démon.
-v, –version	La version du client Chef Infra.
-W, –why-run	Exécuter l'exécutable en mode pourquoi, qui est un type d'exécution de **Chef Infra Client** qui fait tout sauf modifier le système. Utiliser le mode d'exécution pour comprendre les décisions prises par **Chef Infra Client** lors d'une exécution et pour en savoir plus sur l'état actuel et proposé du système.

Exécution en tant qu'utilisateur non root

chef-solo peut être exécuté en tant qu'utilisateur non root. Par exemple, on peut mettre à jour le fichier sudoers :

```
# spécification du privilège chef-solo
chef ALL=(ALL) NOPASSWD: /usr/bin/chef-solo
```

où **chef** est le nom de l'utilisateur non root. Cela permettrait à chef-solo d'exécuter n'importe quelle commande sur le nœud sans nécessiter de mot de passe.

Exemples d'utilisation

Exécuter chef-solo en utilisant les paramètres `~/chef/solo.rb`

```
chef-solo -c ~/chef/solo.rb
```

Utiliser une URL

```
chef-solo -c ~/solo.rb -j ~/node.json -r
http://www.example.com/chef-solo.tar.gz
```

Le fichier tar.gz est archivé dans **file_cache_path**, puis extrait dans **cookbooks_path**.

Utiliser un répertoire

```
chef-solo -c ~/solo.rb -j ~/node.json
```

chef-solo cherchera dans le fichier `solo.rb` pour déterminer le répertoire dans lequel se trouvent les cookbooks.

Utiliser une URL pour le cookbooks et les données JSON

```
chef-solo -c ~/solo.rb -j http://www.example.com/node.json --recipe-url
http://www.example.com/chef-solo.tar.gz
```

où **–recipe-url** correspond à **recette_url** et **-j** correspond à **json_attribs**, qui sont tous deux des options de configuration dans `solo.rb`.