Version : **2023.01**

Dernière mise-à-jour : 2023/11/12 11:41

LDF701 - Présentation de Progress Chef

Contenu du Module

Présentation

Chef est un logiciel libre de gestion de configuration sous la licence **Apache 2.0** qui permet l'automatisation de la plupart des tâches de la gestion d'une infrastructure IT. En tant que tel, c'est un outil **IAC** (Infrastructure as Code).

Chef a été créé par **Adam Jacob**. Une société appelée **OpsCode** a été créée en **2008** par **Adam Jacob**, **Jesse Robbins**, **Barry Steinglass**, **Nathan Haneysmith** et **Joshua Timberman** pour développer Chef avec la première version sortie en **janvier 2009**.

Les fondateurs étaient, pour la plupart, des individus des équipes des data centers d'Amazon™ et de Microsoft™.

Chef a été développé en utilisant **Ruby** et **Erlang**.

Chef était disponible en trois versions :

- **Chef Basic**,
  - disponible gratuitement, cette version offrait une liste de fonctionnalités limitée par rapport aux deux autres versions,
- **Hosted Chef**,
  - hébergé par chef dans leur propre cloud, cette version représentait la façon la plus simple pour démarrer avec Chef,
  - cette version était gratuite pour jusqu'à 5 nœuds. Au-delà, l'accès était conditionné au paiement d'un abonnement lié à un contrat de support,
- **Chef Automate**,
  - cette version, basée sur les produits OpenSource **Chef**, **InSpec** et intégrée avec **Habitat**, était destinée à être installée dans l'infrastructure du client et était gratuite pour pour jusqu'à 25 nœuds,
  - au-delà, l'accès était conditionné au paiement d'un abonnement lié à un contrat de support.

En avril 2019, OpsCode a annoncé que le code source des logiciels serait toujours sous la licence Apache 2.0 mais que les binaires seraient disponibles sous licence propriétaire. A cette date, le projet **Cinc** (**C**inc **I**s **N**ot **C**hef) a commencé à sortir des versions des binaires Chef sous la licence Apache 2.0.

En 2020, la société OpsCode a été achetée par la société **Progress** pour $220 millions. La société est devenue ansuite **Progress Chef**.

A faire - Pour plus d'information concernent **Progress Chef** et son offre actuelle, consultez le site **https://www.chef.io/**.

Actuellement, Chef est disponible en deux **modèles** :

- **Chef-Zero**,
  - utilisé pour le développement et le testing,
- **Client-Serveur**,
  - utilisé dans des cas de production.

Chef supporte de multiples OS en tant que clients :

- Red Hat,
- CentOS,
- Ubuntu,
- Oracle Linux,
- MacOS™,
- AIX™,
- Solaris™,
- Windows™.

L'Organisation de Chef

Chef organise les systèmes IT dans des **Organisations**. Une organisation peut être :

- une société,
- une division ou un département d'une grosse société.

Chef peut gérer de multiples organisations au sein du même Chef Infra Server.

Chef est **Multi-Tenancy** qui implique que l'infrastructure dans un Chef Infra Server peut être gérée par de multiples administrateurs.

Les différentes étapes dans le cycle de vie d'une application, par exemple **Développement**, **Test**, **User Accepting Testing (UAT)** et **Production**, sont représentées par des **Environnements** au sein de Chef. L'Environnement par défaut de Chef s'appelle **_default**.

Chaque serveur dans une infrastructure remplit une fonction spécifique. Chacune des ces fonctions est appelée un **Rôle** sous Chef. Les serveurs sont appelés des **Nœuds** de >Chef ou encore des **Clients**.

Chaque serveur dans l'infrastructure ne peut être associé à :

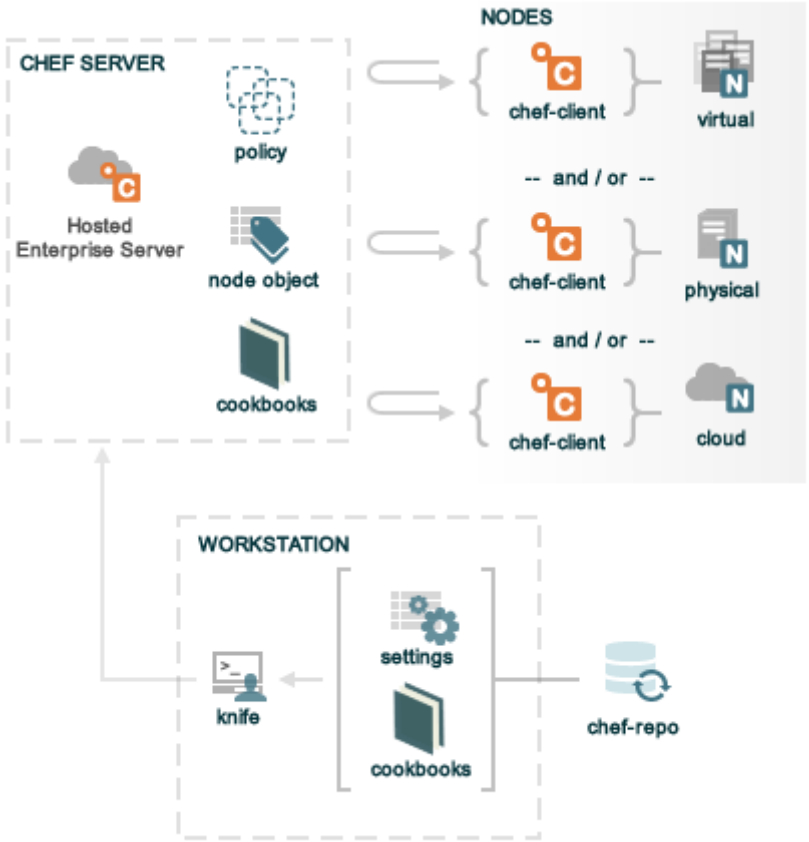
- **une seule** organisation,
- **un seul** environnement.

Par contre un serveur peut avoir de multiples **Rôles**.

L’Écosystème de Chef

L'écosystème de Chef comprend trois éléments :

- **Chef Infra Server**,
  - est le composant au cœur de Chef. Il agit comme un point centralisé de stockage des politiques de configuration ainsi que des informations concernant chaque nœud dans l'infrastructure. Ces informations sont stockées dans une base de données **PostgreSQL**,
- **Chef Workstation**,
  - permet aux administrateurs d'interagir avec le Chef Infra Server afin de gérer l'infrastructure.
- **Nœuds**,
  - sont des clients gérés par le Chef Infra Server. Chaque nœud est équipé d'un **Client Chef** ainsi que d'autres composants nécessaires au bon fonctionnement de Chef.



Chef Infra Server

Le **Chef Infra Server** nécessite une des distributions Linux suivantes :

- Red Hat Enterprise Server / CentOS versions 6 ou 7,
- SUSE Linux Enterprise Server versions 11 ou 12,
- Ubuntu Linux versions, 14.04, 16.04 ou 18.04.

Important - Notez que Chef Infra Server ne peut **pas** être installé sur

Windows™.

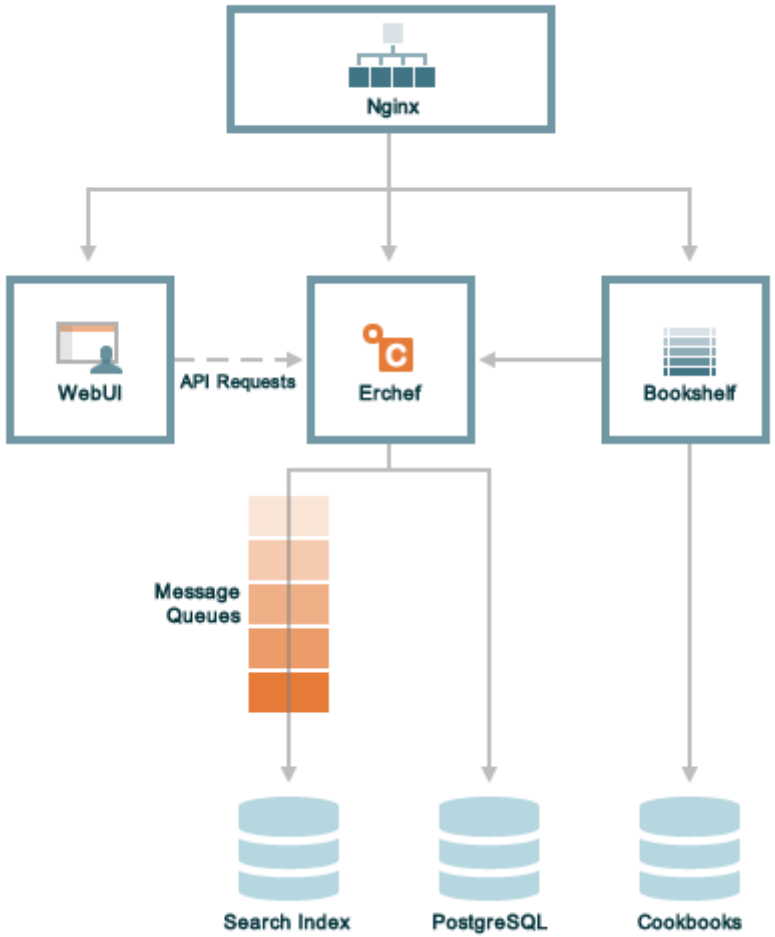
La configuration minimale de Chef Infra Server est :

- architecture 64 bits,
- les ports 80 et 443 doivent être ouverts,
- serveur NTP installé,
- 4 cœurs physiques ou virtuels,
- 4 - 8 Go de RAM,
- 5 Go d'espace disque dans /var,
- 5 Go d'espace disque dans /opt.

Le Chef Infra Server contient :

- **ErChef**,
  - un serveur API écrit en Er lang qui s'occupe des requêtes API envoyées au Chef Infra Server,
- **Nginx**,
  - un front-end s'occupant de l'équilibrage de charge. Toute requête est d'abord passée par Nginx qui la route vers l'API appropriée,
- **Web UI**,
  - une interface Web pour la gestion du Chef Infra Server développée en **Ruby on Rails**,
- **Message Queues**,
  - envoie des messages au Search Index avec l'aide de :
    - **RabbitMQ**, un logiciel libre qui utilise l'**Advanced Message Queue (AMQ) Protocol**,
    - **Chef Expander** qui reçoit des messages de RabbitMQ et
    - **Chef-Solr**, un wrapper autour d'**Apache Solr**, est utilisé pour gérer l'API REST pour l'indexation en temps réel et la recherche **Full Text** dans les messages,
- **Search Index**,
  - stocke tous les messages ainsi que la liste, au format **Full Text**, de tous les objets dans le Chef Infra Server,
- **PostgreSQL**,
  - un logiciel de gestion des bases de données relationnelles utilisé par le Chef Infra Server pour stocker les informations concernant les :
    - configurations,
    - environnements,
    - rôles,

        - Data Bags,
        - attributs des nœuds.
- **Bookshelf**,
    - stocke tous les objets téléversés vers le Chef Infra Server à partir du Workstation, par exemple :
        - Cookbooks,
        - Recettes,
        - Templates
- **Cookbooks**,
    - contiennent des Recettes, Attributs, Versions, Gabarits, Ressources, et Bibliothèques,
- **Node Objects**,
    - contiennent la **Run-List** et les Attributs, sous forme de fichiers au format JSON, stockés sur le Chef Infra Server,
        - une Run-List contient des Recettes et Rôles à exécuter sur un nœud,
        - les Attributs sont, par exemple, des informations concernant :
            - des paquets à installer,
            - des services,
            - des utilisateurs,
            - des fichiers et répertoires à créer,
            - le réseau,
- **Policies**
    - contiennent des informations concernant :
        - rôles,
        - environnements.



Chef Workstation

Chef Workstation contient les exécutables suivantes :

- ChefDK
- Chef-run

- Chef Infra Client
- Chef-repo

La configuration minimale de Chef Workstation est :

- 4 Go de RAM
- 4 Go d'espace disque

La configuration recommandée de Chef Workstation est :

- 4 Go de RAM
- 8 Go d'espace disque

Le **Chef Workstation** contient aussi :

- **Knife**,
  - une outil **CLI** pour interagir avec le Chef Infra Server et le Chef Repository et utilisé pour gérer des :
    - Nœuds,
    - Rôles,
    - Cookbooks,
    - Recettes,
    - Environnements,
    - Données cryptées dans des Data Bags,
    - Search Index,
- **Chef Repository**,
  - contient tous les objets de données, sous forme de code, par exemple, des détails concernant les :
    - Environnements,
    - Rôles,
    - Data Bags,
    - Cookbooks.

Chef Nœuds

Un **Chef Nœud** contient :

- **Chef Client**
  - un agent. L'agent doit être **inscrit** ou **référencé** chez le Chef Infra Server,
  - le travail de l'agent est de ramené le nœud à l'état spécifié par le Chef Infra Server. Ce processus est appelé un **Chef Run**,
- **OHAI**
  - un outil installé sur le nœud afin que le Chef Infra Server puisse être informé des valeurs de chaque Attribut concernant le nœud. Ces attributs sont appelés des **Attributs Automatiques**, par exemple :
    - la plateforme,
    - l'OS,
    - la version de l'OS,
    - l'architecture,
    - les détails du système de fichiers,
    - les informations concernant la mémoire,
    - les paquets installés,
    - la configuration du réseau,
    - e.t.c ...

Cookbooks, Recettes, Ressources et Attributs

Rappelez-vous que les **Cookbooks** contiennent des **Recettes**, **Ressources**, **Attributs**, **Versions**, **Gabarits** et **Bibliothèques**.

Recettes

Une **Recette** :

- est un fichier Ruby contenant tout ce qui est necéssaire afin de configurer un Noeud à un état précis,
- peut contenir d'autres Recettes,
- contient en règle générale des Ressources,
- est appelée par la Run-List du Noeud.

Les Recettes sont exécutées par la commande **chef-run** dans l'ordre qu'elles apparaissent dans la Run-List.

Ressources

Une **Ressource** :

- représente une utilisateur, un groupe, un paquet, un service etc ...,
- plusieurs Ressources peuvent être définies dans une Recette,
- peut contenir plusieurs Attributs.

Attributs

Chaque Ressource définie dans une Recette possède un **état** désiré ou une **valeur** associée.

Un **Attribut** :

- est utilisé pour définir un état ou une valeur,
- peut être utilisé pour contenir des informations concernant un composant présent sur le Nœud, par exemple, des informations concernant l'OS du Nœud.

Création d'un Cookbook et d'une Recette

Un Cookbook est créé sur un Chef Workstation dans un répertoire appelé **chef-repo/cookbooks**. La commande utilisée pour créer un Workbook est :

```
# chef generate cookbook <chemin>/<name>
```

Par exemple, dans le cas de la création d'un Cookbook nommé **apache**, la commande devient :

```
root@workstation:~/chef-repo# chef generate cookbook cookbooks/apache
Generating cookbook apache
- Ensuring correct cookbook content

Your cookbook is ready. Type `cd cookbooks/apache` to enter it.
```

```
There are several commands you can run to get started locally developing and testing your cookbook.
Type `delivery local --help` to see a full list of local testing commands.

Why not start by writing an InSpec test? Tests for the default recipe are stored at:

test/integration/default/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:

recipes/default.rb
```

A faire - Pour plus d'information concernant la commande **chef generate cookbook**, consultez **cette page**.

Lors de la création du Cookbook, une arborescence est créée dans le répertoire **chef-repo/cookbooks/apache** :

```
root@workstation:~/chef-repo# ls -l cookbooks/apache
total 40
-rw-r--r-- 1 root root  150 nov.   6 10:11 CHANGELOG.md
-rw-r--r-- 1 root root 1176 nov.   6 10:11 chefignore
-rw-r--r-- 1 root root  741 nov.   6 10:11 kitchen.yml
-rw-r--r-- 1 root root   70 nov.   6 10:11 LICENSE
-rw-r--r-- 1 root root  676 nov.   6 10:11 metadata.rb
-rw-r--r-- 1 root root  507 nov.   6 10:11 Policyfile.rb
-rw-r--r-- 1 root root   54 nov.   6 10:11 README.md
drwxr-xr-x 2 root root 4096 nov.   6 10:11 recipes
drwxr-xr-x 3 root root 4096 nov.   6 10:11 spec
drwxr-xr-x 3 root root 4096 nov.   6 10:11 test
```

Une Recette par défaut, appelé **default.rb** a été créée dans le sous-répertoire **recipes** :

```
root@workstation:~/chef-repo# ls -l cookbooks/apache/recipes/
total 4
-rw-r--r-- 1 root root 97 nov.   6 10:11 default.rb
```

Ce fichier prend la forme suivante :

```
root@workstation:~/chef-repo# cat cookbooks/apache/recipes/default.rb
#
# Cookbook:: apache
# Recipe:: default
#
# Copyright:: 2023, The Authors, All Rights Reserved.
```

Étudions maintenant une Recette destinée à installer, activer et démarrer le service **httpd** sur un système CentOS :

```
root@workstation:~/chef-repo# cat cookbooks/apache/recipes/default.rb
#
# Cookbook:: apache
# Recipe:: default
#
# Copyright:: 2023, The Authors, All Rights Reserved.
package "httpd" do
  action :install
end

service "httpd" do
  action [:start, :enable]
end
```

Dans ce fichier **deux** types de Ressources sont définies :

- **package**,
- **service**.

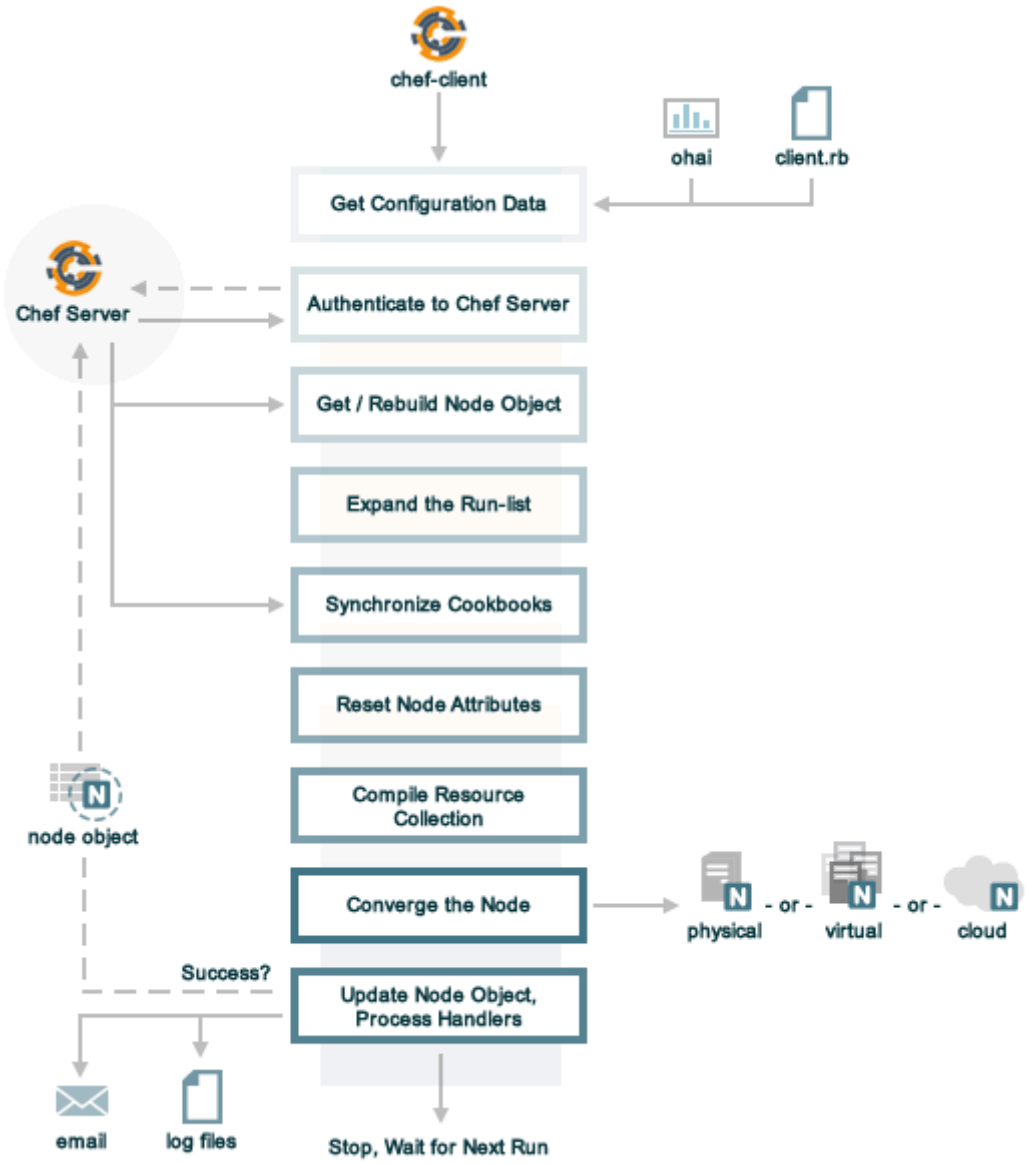
Dans le cas de la Ressources **package**, le nom de la Ressource est le nom du paquet à installer tandis que dans le cas de la Ressources **service**, le nom de la Ressource est le nom du service.

Important - Notez que, comme les noms son variables, ceux-ci doivent être entourés des caractères **“**.

Chaque Ressource contient une ou plusieurs **actions**. Chaque action est précédée par un caractère **:**. Dans le cas de plusieurs actions, celles-ci doivent être entourées des caractères **[]** et séparées par des virgules.

Après avoir téléversé le Workbook **apache** vers le Chef Infra Server pour l'inclure dans la Run-List du Nœud concerné, la commande **chef-run** peut être exécutée sur les Nœud.

Le Processus chef-run



Quand la commande **chef-run** est exécutée sur le Nœud, **OHAI** est exécutée et les **Attributs Automatiques** sont collectés. L'information collectée est retournée au format JSON.

Ensuite chef-run s'authentifie auprès du Chef Infra Server en utilisant la cléf **/etc/chef/client.pem**. Si ce fichier n'est pas présent, chef-run recherche le fichier **/etc/chef/validation.pem**.

Une fois la connexion authentifiée, chef-client demande au Chef Infra Server les Cookbooks à appliquer au Nœud.

Le Chef Infra Server consulte les objets concernant le Nœud. Un de ces objets contient la Run-List applicable au Nœud et contient des objets Recettes et Rôles. Le Chef Infra Server envoie ensuite la Run-List et ces objets au Nœud sous forme d'objets JSON.

A la réception des objets, le Nœud identifie les Recettes et les Rôles à appliquer. Une fois l'identification terminée, le Nœud télécharge du Chef Infra Server tous les Cookbooks contenants des références à ces mêmes Recettes et Rôles, sauf dans le cas où des Cookbooks **identiques** existent déjà sur le Nœud.

Si dans les Recettes téléchargées, il existe des dépendances, le Nœud télécharge les Recettes dépendantes. Les Recettes sont stockées dans le répertoire **/var/chef/cache** du Nœud.

La commande **chef-run** charge ensuite toutes les bibliothèques requises par Ruby et les autres langages invoqués.

Une fois la commande chef-run terminée sans erreurs, un message est envoyée au **Message Queue**. Dans le cas d'erreurs, celles-ci sont envoyées à l'administrateur.

Types de Ressources Chef

La déclaration d'une Ressource dans Chef prend la forme suivante :

```
type "name" do
  attribute "value"
  attribute "value"
  action :type_action
  action :type_action
end
```

Important - Notez qu'un **Attribut** est aussi connu sous le nom *Propriété*.

Les Ressources les plus courantes sous Chef sont :

- La Ressource Package
- La Ressource Service
- La Ressource Directory
- La Ressource File
- La Ressource Bash
- La Ressource Execute
- La Ressource Cron
- La Ressource Cookbook_File

La Ressource Package

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:install**,
  - installe le paquet indiqué. Cette action est l'action par défaut,
- **:upgrade**,
  - met à jour le paquet indiqué. Dans le cas où le paquet n'est pas présent, celui-ci est installé,
- **:remove**,
  - supprime le paquet indiqué,
- **:purge**,
  - supprime le paquet indiqué. Ensuite le(s) fichier(s) de configuration du paquet est(sont) supprimé(s),
- **:reconfig**,
  - reconfigure le paquet indiqué,
- **:nothing**,
  - informe Chef de ne pas installer le paquet.

Les **attributs** disponibles pour cette Ressource sont :

- **allow_downgrade**,
  - permet de rétrograder le paquet indiqué pour satisfaire des dépendances. La valeur par défaut de cet attribut est **faux**,
- **arch**,
  - permet de spécifier une autre architecture du paquet indiqué. Par défaut Chef installe l'architecture du paquet correspondant à

l'architecture du Nœud,
- **flush_cache**,
  - permet de supprimer le cache du gestionnaire des paquets. Ceci peut être fait avant ou après l'installation du paquet. L'attribut prend donc deux options; **before** ou **after**,
- **options**,
  - permet de spécifier des options lors de l'action sur le paquet concerné,
- **source**,
  - permet de spécifier un paquet stocké sur disque en indiquant le chemin complet,
- **version**,
  - permet de spécifier une version spécifique du paquet indiqué.

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.

La Ressource Service

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:enable**,
  - active le service indiqué,
- **:disable**,
  - désactive le service indiqué,
- **:start**,
  - démarre le service indiqué,
- **:stop**,
  - arrête le service indiqué,
- **:restart**,
  - redémarre le service indiqué,
- **:reload**,
  - recharge la configuration du service indiqué,

- **:nothing**,
  - informe Chef de ne rien faire.

Les **attributs** disponibles pour cette Ressource sont :

- **init_command**,
  - permet de spécifier le chemin vers le script associé avec le service indiqué,
- **priority**,
  - permet de spécifier la priorité du service indiqué lors du démarrage ou de l'arrêt du système,
- **start_command**,
  - permet de spécifier la commande utilisée pour démarrer le service indiqué,
- **stop_command**,
  - permet de spécifier la commande utilisée pour arrêter le service indiqué,
- **restart_command**,
  - permet de spécifier la commande utilisée pour redémarrer le service indiqué,
- **status_command**,
  - permet de spécifier la commande utilisée pour indiquer le statut du service indiqué,
- **reload_command**,
  - permet de spécifier la commande utilisée pour recharger le service indiqué,
- **supports**,
  - permet de spécifier les actions compatibles avec le service indiqué.

Dans le cas de l'utilisation du dernier Attribut, l'exemple suivant est un exemple :

```
service "httpd" do
  supports :status => true, :restart => true, :reload=> true
  action [ :enable, :start ]
end
```

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.

La Ressource Directory

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:create**,
  - crée le répertoire indiqué. Cette action est l'action par défaut,
- **:delete**,
  - supprime le répertoire indiqué,
- **:nothing**,
  - informe Chef de ne rien faire.

Les **attributs** disponibles pour cette Ressource sont :

- **path**,
  - permet de spécifier le chemin du répertoire indiqué,
- **owner**,
  - permet de spécifier l'UID du répertoire indiqué,
- **group**,
  - permet de spécifier le GID du répertoire indiqué,
- **mode**,
  - permet de spécifier les permissions du répertoire indiqué sous forme octale,
- **recursive**,
  - permet de créer ou de supprimer le répertoire indiqué d'une manière récursive.

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.

La Ressource File

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:create**,
  - crée le fichier indiqué. Cette action est l'action par défaut,
- **:delete**,
  - supprime le fichier indiqué,
- **:create_if_missing**,
  - crée le fichier indiqué uniquement si celui-ci est manquant,
- **:touch**,
  - met à jour la date de dernière modification du contenu (**atime**) et la date du dernier accès (**mtime**) du fichier indiqué,
- **:nothing**,
  - informe Chef de ne rien faire.

Les **attributs** disponibles pour cette Ressource sont :

- **path**,
  - permet de spécifier le chemin du fichier indiqué,
- **content**,
  - permet de spécifier le contenu du fichier indiqué,
- **owner**,
  - permet de spécifier l'UID du fichier indiqué,
- **group**,
  - permet de spécifier le GID du fichier indiqué,
- **mode**,
  - permet de spécifier les permissions du fichier indiqué sous forme octale,
- **backup**,
  - permet de spécifier le nombre de copies de sauvegarde du fichier indiqué. La valeur par défaut et **5**.

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.

La Ressource Bash

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:run**,
  - exécute le script indiqué. Cette action est l'action par défaut,
- **:nothing**,
  - informe BASH de ne pas exécuter le script.

Les **attributs** disponibles pour cette Ressource sont :

- **code**,
  - permet de spécifier le code à exécuter. Le code est entouré de caractères **“**,
- **creates**,
  - permet d'empecher la création d'un fichier si celui-ci existe déjà,
- **command**,
  - permet de spécifier le nom de la commande à exécuter,
- **cwd**,
  - permet de spécifier répertoire de travail de l'exécution du code,
- **user**,
  - permet de spécifier l'UID de l'utilisateur qui doit exécuter le code,
- **group**,
  - permet de spécifier le GID du groupe qui doit exécuter le code,
- **umask**,
  - permet de spécifier le masque pour la création de fichiers,
- **environment**,
  - permet de spécifier le valeurs des variables d'environnement,
- **timeout**,
  - permet de spécifier la durée du timeout de la commande. La valeur est spécifiée en secondes et la valeur par défaut est 3 600s,
- **path**,
  - permet de spécifier le PATH pour rechercher la commande,
- **return**,
  - permet de spécifier le statut du code retour de la commande,
- **flags**,

- permet de spécifier les options à passer à BASH.

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.

La Ressource Execute

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:run**,
  - exécute la commande indiquée. Cette action est l'action par défaut,
- **:nothing**,
  - informe Chef de ne pas exécuter la commande.

Les **attributs** disponibles pour cette Ressource sont :

- **command**,
  - permet de spécifier le nom la commande à exécuter. Dans le cas où **command** n'est pas spécifié, Chef considère que le nom de la commande est identique au nom de la Ressource,
- **creates**,
  - permet d'empecher la création d'un fichier si celui-ci existe déjà,
- **cwd**,
  - permet de spécifier répertoire de travail de l'exécution de la commande,
- **user**,
  - permet de spécifier l'UID de l'utilisateur qui doit exécuter la commande,
- **group**,
  - permet de spécifier le GID du groupe qui doit exécuter la commande,
- **umask**,
  - permet de spécifier le masque pour la création de fichiers,
- **environment**,

    - permet de spécifier le valeurs des variables d'environnement,
- **timeout**,
    - permet de spécifier la durée du timeout de la commande. La valeur est spécifiée en secondes et la valeur par défaut est 3 600s,
- **path**,
    - permet de spécifier le PATH pour rechercher la commande,
- **return**,
    - permet de spécifier le statut du code retour de la commande,

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.

La Ressource Cron

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:create**,
    - crée un job cron. Si le job existe déjà, celui-ci est mis à jour. Cette action est l'action par défaut,
- **:delete**,
    - supprime un job cron,
- **:nothing**,
    - informe Chef de ne rien faire.

Les **attributs** disponibles pour cette Ressource sont :

- **command**,
    - permet de spécifier le nom la commande ou le script à exécuter,
- **day**,
    - permet de spécifier le jour du mois de l'exécution du job. La valeur est entre **1** et **31**. La valeur par défaut est *,
- **hour**,
    - permet de spécifier l'heure du jour de l'exécution du job. La valeur est entre **0** et **23**. La valeur par défaut est *,

- **minute**,
  - permet de spécifier la minute de l'heure de l'exécution du job. La valeur est entre **0** et **59**. La valeur par défaut est *,
- **month**,
  - permet de spécifier le mois de l'année de l'exécution du job. La valeur est entre **1** et **12**. La valeur par défaut est *,
- **week**,
  - permet de spécifier le jour de la semaine de l'exécution du job. La valeur est entre **0** et **6** où **0** est dimanche. La valeur par défaut est *,
- **user**,
  - permet de spécifier l'UID de l'utilisateur à qui appartient le job,
- **group**,
  - permet de spécifier le GID du groupe à qui appartient le job,
- **path**,
  - permet de spécifier la valeur de $PATH pour le job,
- **shell**,
  - permet de spécifier la valeur de $SHELL pour le job,
- **home**,
  - permet de spécifier la valeur de $HOME pour le job,
- **mailto**,
  - permet de spécifier la valeur de la variable **mailto** pour le job.

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.

La Ressource Cookbook_File

Cette Ressource est utilisée pour transférer des fichiers se trouvant dans le sous-répertoire **files** du Cookbook du Chef Workstation vers les Nœuds.

Les **actions** les plus courantes disponibles pour cette Ressource sont :

- **:create**,
  - crée un fichier sur le nœud indiqué. Cette action est l'action par défaut,

- **:delete**,
  - supprime un fichier du nœud indiqué,
- **:create_if_missing**,
  - crée le fichier indiqué uniquement si celui-ci est manquant,
- **:touch**,
  - met à jour la date de dernière modification du contenu (**atime**) et la date du dernier accès (**mtime**) du fichier indiqué,
- **:nothing**,
  - informe Chef de ne rien faire.

Les **attributs** disponibles pour cette Ressource sont :

- **path**,
  - permet de spécifier le chemin de destination pour le fichier. Dans le cas où **path** n'est pas spécifié, Chef essaie de déterminer la valeur à partir du nom de la Ressource,,
- **content**,
  - permet de spécifier le contenu du fichier indiqué,
- **owner**,
  - permet de spécifier l'UID du fichier indiqué,
- **group**,
  - permet de spécifier le GID du fichier indiqué,
- **mode**,
  - permet de spécifier les permissions du fichier indiqué sous forme octale,
- **backup**,
  - permet de spécifier le nombre de copies de sauvegarde du fichier indiqué. La valeur par défaut et **5**,
- **source**,
  - permet de spécifier où se trouve le fichier dans le sous-répertoire files du Cookbook concerné,

A faire - Pour plus d'informations concernant cette Ressource, consultez **cette page**.