Automatisation de l'administration de 700 serveurs avec Chef

Alain Heinrich
Direction Informatique / Université de Strasbourg
4 rue Blaise Pascal
CS 90032
67081 STRASBOURG Cedex

Christophe Palanché
Direction Informatique / Université de Strasbourg
4 rue Blaise Pascal
CS 90032
67081 STRASBOURG Cedex

Résumé

Les logiciels de gestion de configurations et de déploiement automatisé de système sont maintenant devenus incontournables pour administrer un grand nombre de serveurs. Ces outils permettent de définir et déployer de façon centralisée les configurations des systèmes et des applications hébergées. La reproductibilité du déploiement simplifie l'administration et apporte une plus grande fiabilité du service.

À l'heure actuelle, plusieurs solutions existent. Nous détaillerons dans cette présentation les raisons pour lesquelles notre choix s'est porté sur le logiciel libre développé par la société Opscode : Chef. Nous présenterons aussi les différentes fonctionnalités qu'il apporte ainsi que l'infrastructure mise en œuvre. Nous décrirons ensuite la démarche que nous avons suivie pour intégrer nos serveurs à Chef en nous appuyant sur des exemples concrets tels que les serveurs de messagerie, les hôtes de virtualisation et la solution de groupware SOGo. Pour finir, nous aborderons les évolutions possibles et notamment l'utilisation de Chef dans les plates-formes d'orchestration et d'intégration continue.

Mots-clefs

gestion de configuration, administration système, automatisation, Chef

1 Introduction

L'automatisation des tâches d'administration et de déploiement des plates-formes a toujours été une des préoccupations majeures des administrateurs système. En effet, en plus de permettre un gain de temps, la reproductibilité du processus de déploiement est une garantie du résultat. Avec l'arrivée de la virtualisation et l'augmentation des services proposés aux utilisateurs, les parcs de serveurs sont en forte expansion. Pour répondre à l'accroissement du nombre de plates-formes gérées, de nouveaux outils d'administration sont apparus. Les logiciels de gestion de configuration et de déploiement automatisé sont maintenant une alternative aux installations manuelles ou réalisées à l'aide de « scripts maison ». De nombreuses solutions existent, alors que certaines se contentent d'envoyer des commandes en parallèle sur plusieurs serveurs, d'autres permettent de décrire les configurations et les processus d'installation des services dans un langage dédié (DSL[1]) qui sera interprété par les clients.

La Direction Informatique de l'Université de Strasbourg héberge à l'heure actuelle plus de 700 serveurs avec une grande diversité d'applications. Le coût humain et les erreurs dues à l'administration manuelle de ces systèmes nous ont amenés à évaluer différentes solutions d'automatisation, avec pour objectif d'en déployer une.

1. *Domain-Specific Language*

2 Solutions testées

Suite au recensement des différents produits existants, nous en avons testé trois : « CFEngine », maintenu par la société CFEngine, « Puppet » supporté par Puppetlabs et « Chef » développé par Opscode. Ces trois produits sont libres ou proposent une version communautaire.

CFEngine est un des premiers outils de gestion de configuration et il est souvent considéré comme la source d'inspiration des autres produits. Il se caractérise par une faible empreinte mémoire et de bonnes performances. En revanche son langage de description de très bas niveau rend complexe son utilisation.

Puppet et Chef sont des solutions très proches et proposent globalement les mêmes fonctionnalités. Néanmoins alors que le langage de description utilisé par Puppet est strict et figé, Chef utilise un langage spécifique basé sur Ruby, ce qui permet une plus grande souplesse dans l'écriture des recettes. La méthode d'ordonnancement des actions diffère également entre les deux produits, Puppet détermine l'ordre final d'exécution des actions en construisant un graphe orienté, alors que Chef exécute les actions dans l'ordre quand lequel elles ont été décrites.

La souplesse qu'apporte Ruby dans l'écriture des recettes et la simplicité dans l'ordonnancement d'exécution sont les principales raisons de notre choix d'utiliser Chef.

3 Présentation de Chef

Chef est composé de deux parties : chef-server et chef-client. Il met à la disposition de l'administrateur deux outils d'administration : knife et webui. Voici une rapide présentation de ces différents composants et des structures de données indispensables à l'utilisation et au fonctionnement de Chef.

3.1 Chef-server

Chef-server centralise l'ensemble des informations nécessaires au déploiement d'un nœud (nom donné aux hôtes gérés par Chef). Il est composé d'une application serveur (Erchef), de *backends* servant à stocker les données (Bookshelf et PostgreSQL), d'une interface Web (WebUI) et d'un moteur d'indexation. En plus de permettre l'accès aux données via des outils d'administration, Chef fournit une API.

3.2 Chef-client

Chef-client est l'agent installé sur tous les nœuds. À chaque exécution, il s'authentifie auprès du serveur, y récupère les informations nécessaires, et exécute les actions telles qu'elles sont décrites dans les recettes. La liste des recettes à exécuter est définie dans la run-list du nœud.

Chef-client fait également appel à ohai, un agent qui va collecter les informations qui servent à renseigner les attributs du nœud sur le serveur (hostname, adresse IP, etc.).

3.3 Les cookbooks

L'ensemble des informations servant à configurer un nœud sont stockées dans des cookbooks. Ils sont composés de :

- Recettes. Elles décrivent dans un langage spécifique (DSL) basé sur Ruby l'ensemble des actions à effectuer sur un nœud. L'utilisation de ressources fournies par Chef en simplifie l'écriture, par exemple l'installation d'application se fait grâce à la ressource package, qui fournit un niveau d'abstraction par rapport au système d'exploitation cible.

- Attributs. Ce sont des variables qui peuvent être définies soit par l'état du nœud dans les cookbooks, soit dans les rôles, soit dans les environnements. Les attributs sont utilisés par les recettes lors de l'exécution de chef-client.

- Fichiers à transférer sur les nœuds. Il peut s'agir de fichiers bruts ou de *templates* écrits en eRuby. L'utilisation des *templates* permet de personnaliser les fichiers de configuration en utilisant par exemple des attributs, des *databag* ou encore des sources de données externes.

Il est possible dans Chef de définir des versions pour un cookbook et ainsi de spécifier pour un nœud la version à utiliser. Ceci permet d'avoir une version stable pour la production et d'utiliser la version en développement pour les tests. Ce système ne remplace pas un logiciel de gestion de version des sources, il est donc recommandé d'en utiliser un en parallèle, tel que Git.

Opscode centralise un grand nombre de cookbooks directement utilisables pour le déploiement d'application dans son infrastructure. On trouvera par exemple des cookbooks pour déployer Apache, MySQL, Nginx, etc.

3.4 Databags, rôle et environnement

Les databags permettent de stocker des informations au format JSON, ils sont centralisés et accessibles depuis l'ensemble des cookbooks. Il est également possible de les chiffrer afin de stocker des informations sensibles comme des mots de passe.

Les rôles sont appliqués à des nœuds, qui héritent alors de la run-list et des attributs définis dans celui-ci. Cela permet de factoriser la configuration de la run-list et des attributs pour des serveurs de même type.

Les environnements sont appliqués à des nœuds. Ils contiennent des attributs qui sont appliqués aux nœuds, et permettent de définir des contraintes sur les versions des cookbooks à installer.

3.5 Outils d'administration

3.5.1 Knife

Knife est un outil en ligne de commande qui est utilisé pour gérer l'ensemble des composants de Chef. C'est l'outil principal d'administration de la plate-forme. On l'utilisera par exemple pour :

- déployer chef-client sur des nouveaux nœuds,

```
knife bootstrap Serveur
```

- gérer les cookbooks,

```
knife cookbook upload Cookbook
```

- créer ou modifier les databags, les rôles, les attributs et les run-list,

```
knife data bag edit Bag Item
```

- exécuter des commandes sur des nœuds via ssh,

```
knife ssh node:serveur "chef-client"
```

- Etc.

3.5.2 WebUI

WebUI est une interface web d'administration de Chef, elle est intéressante pour effectuer des opérations simples, comme la modification d'une run-list, d'un rôle ou d'un databag, mais ne permet pas l'édition de cookbooks.

4 Retour d'expérience

Nous avons déployé la version 11 du logiciel Chef et utilisons conjointement Git pour gérer la version des cookbooks. Bien qu'il soit possible de stocker tout dans un seul dépôt, nous avons fait le choix de créer un dépôt par cookbook. Cela simplifie la gestion des accès concurrents aux cookbooks. À l'heure actuelle, cinq plates-formes sont totalement déployées en utilisant Chef. Il s'agit du groupware SOGo, du gestionnaire de liste Sympa, des services OSIRIS de relayage de messagerie et de DNS récursifs, et des serveurs hôtes de virtualisation. Ces déploiements se décomposent en deux familles :

- des cookbooks simples impactant un grand nombre de nœuds,
- des cookbooks complexes impactant peu de nœuds.

4.1 Contraintes et bénéfices

L'intégration de plates-formes dans Chef n'est pas immédiate car elle impose de formaliser le processus de déploiement des services dans un langage de haut niveau. Ainsi toutes les étapes de celui-ci doivent être décrites dans des recettes et testées avant leur mise en production ce qui peut être coûteux en temps. De plus après leur intégration, toutes les modifications doivent être faites via Chef, ce qui impose l'adoption du produit par l'ensemble des équipes.

Malgré ces contraintes, les bénéfices de Chef sont importants, il permet notamment :

- De déployer plus rapidement de nouveaux serveurs.
- De gérer de façon centralisée les paramètres de configuration des applications déployées et d'en faciliter la modification, par exemple depuis le déploiement des hôtes de virtualisation avec Chef, il est simple d'ajouter un réseau à ces serveurs.
- De déployer facilement dans divers environnements des plates-formes complexes. Il est maintenant, par exemple, immédiat de déployer un environnement de développement Sympa ou une plate-forme de test SOGo.
- De garantir le résultat d'un déploiement grâce à la reproductibilité du processus, on peut effectivement tester et valider les cookbooks sur les plates-formes de tests et de pré-production avant de les appliquer en production.

4.2 Illustration de fonctionnalités de Chef

4.2.1 SOGo : paramétrage de template avec des databags

Pour l'écriture de la recette « sogo.rb » qui déploie les packages utiles au fonctionnement de SOGo et génère sa configuration à partir du fichier template « GNUstepDefaults.erb », nous utilisons un item du databag chiffré « sogo-conf » qui contient tous les éléments de paramétrage y compris les mots de passe utilisés par le template.

```
#Recette sogo.rb
…
sogo_conf = Chef::EncryptedDataBagItem.load("sogo-conf",node.chef_environment)
…
template "/home/sGNUstep/Defaults/.GNUstepDefaults" do
  source "GNUstepDefaults.erb"
  variables(
    :bdd_login => sogo_conf['bdd_login'],
    :bdd_password => sogo_conf['bdd_password'],
    :bdd_host => sogo_conf['bdd_host'],
    …
  )
…
```

```
#Template GNUstepDefaults.erb
…
<key>OCSEMailAlarmsFolderURL</key>
<string>postgresql://<%= @bdd_login %>:<%= @bdd_password %>@<%= @bdd_host %>:<%=
@bdd_port %>/<%= @bdd_name %>/sogo_alarms_folder</string>
<key>OCSFolderInfoURL</key>
<string>postgresql://<%= @bdd_login %>:<%= @bdd_password %>@<%= @bdd_host %>:<%=
@bdd_port %>/<%= @bdd_name %>/sogo_folder_info</string>
<key>OCSSessionsFolderURL</key>
<string>postgresql://<%= @bdd_login %>:<%= @bdd_password %>@<%= @bdd_host %>:<%=
@bdd_port %>/<%= @bdd_name %>/sogo_sessions_folder</string>
…
```

```
#Item prod du databag chiffré sogo-conf
{
  …
  "bdd_host": "bddsogo.u-strasbg.fr",
  "bdd_password": "mot de passe en clair",
  …
}
```

4.2.2 Hôte de virtualisation : utilisation des attributs du rôle avec surcharge au niveau du nœud

Les 90 hôtes de virtualisation gérés par chef ont une configuration identique, mis à part le réseau. En effet, la liste des VLANs utilisés doit être personnalisée sur certains hôtes. Par défaut cette liste sera définie dans un attribut de rôle, qui pourra être surchargé au niveau d'un nœud.

```
# Définition de l'attribut dans le rôle hôte de virtualisation
…
"default_attributes": {
"kvm": { "vlans": [ "208", "752", "760", "896", … ] }
}
…
```

```
# Surcharge de l'attribut dans un nœud
…
"kvm": { "vlans": [ "120", "250", … ] }
…
```

```
# Utilisation de l'attribut dans le template interfaces.erb
<% node[:kvm][:vlans].each do |vlan| -%>
    auto br<%= vlan %>
    iface br<%= vlan %> inet manual
    bridge_ports bond0.<%= vlan %>
    bridge_stp off
    bridge_fd 2
    bridge_maxage 12
    bridge_maxwait 0
<% end -%>
```

4.2.3 Configuration de base : rôle base-serveur

La plupart des serveurs que nous gérons ont une configuration de base commune, notamment pour les services suivants : ssh, fusion-inventory, snmp, nagios et postfix. L'ensemble de ces services a été intégré dans des recettes, elles-même regroupées dans la run-list du rôle « base-serveur ». Ce rôle est appliqué à tous les nouveaux nœuds pour uniformiser la configuration des services de base et faciliter la modification en masse d'un paramètre, comme par exemple l'ajout d'une sonde Nagios.

5 Perspectives

Chef nous permet maintenant d'automatiser la configuration des systèmes et l'installation des services pour les plates-formes intégrées. Cependant, la création des machines virtuelles étant encore réalisée manuellement à partir de modèles, nous étudions la mise en place d'un outil de gestion centralisée de l'infrastructure de virtualisation qui permettrait le « provisionning » de nouveaux serveurs. Nous aimerions également que ces différentes briques puissent interagir via un outil d'orchestration de plus haut niveau, pour permettre de créer une chaîne de « provisionning » de bout en bout : allocation des ressources (disques, réseaux, bases de données…), création de la machine virtuelle, déploiement des environnements applicatifs et intégration à la supervision et à la sauvegarde. Pour finir nous travaillons avec l'équipe de développement et d'intégration d'applications de la direction informatique pour étudier la faisabilité d'utiliser Chef avec l'outil d'intégration continue qu'ils mettent en place.

6 Conclusions

Aujourd'hui cinq services critiques (environ 130 serveurs) sont déployés de façon totalement automatique avec Chef. Même si l'intégration d'une plate-forme dans Chef est contraignante, les gains en terme d'exploitation (facilité d'extension d'un cluster de serveurs, déploiement des mises à jour, modification de paramétrage, etc.) sont importants. Pour les plates-formes intégrées, les modifications doivent obligatoirement être faîtes via Chef, pour éviter d'être perdues lors des prochains déploiements. Une formation des équipes d'exploitation a donc été réalisée pour les familiariser avec ce nouvel outil de déploiement. Une autre formation des équipes d'ingénierie est également prévue dans le cadre de la généralisation du déploiement automatisé avec Chef.

Suite à ces formations nous souhaitons étendre l'utilisation de Chef à l'ensemble des serveurs gérés par la Direction Informatique.