SARI - Outils de gestion de configuration

Quelques exemples avec CFEngine-2 au LEGI

Gabriel Moreau

CNRS / UGA / Grenoble-INP - France

2 juillet 2018

Licence

Cette présentation est sous : Licence Art Libre

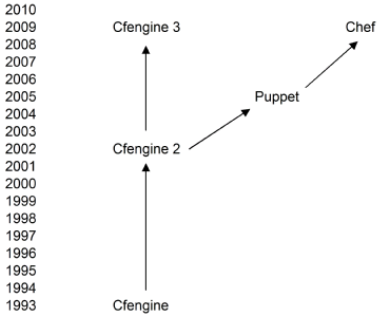
http://artlibre.org/

Historique

- 1993 CFEngine-1 - DSL
- 2002 **CFEngine-2** - DSL
- 2004 Bcfg2 - DSL
- 2004 Puppet - DSL
- 2009 Chef - DSL (Ruby base)
- 2009 CFEngine-3 - DSL
- 2011 Salt / Saltstack - YAML
- 2012 Ansible (sans agent) - YAML

Declarative Domain Specific Language

Relative Origin of Cfengine, Puppet and Chef



Source `http://verticalsysadmin.com`

Comparaison `https://en.wikipedia.org/wiki/Comparison_of_open-source_configuration_management_software`

Historique

Grandes batailles historiques

Même genre de problématique au niveau des outils de sauvegarde

- Vieux débat : **Push / Pull**
- Grande question : **DSL or not DSL** (Declarative Domain Specific Language)
- Longue causerie : **with or without Agent**

CFEngine-2

- Principe **Client / Serveur** dédié en mode **Pull** (Push possible) avec langage **DSL**
- Le client est autonome
- Le client décide de lui même (il vit sa propre vie)
- C'est bien pour les portables !

Historique

À l'origine de toute cette aventure (**1993**) de la gestion de la configuration, un Homme...

- **Mark Burgess** - Emeritus Professor of Network and System Administration, Oslo
- `https://en.wikipedia.org/wiki/Mark_Burgess_(computer_scientist)`
- Chercheur qui a essayé de théoriser et a mis en pratique ses recherches avec du code libre
- 2004 - **Promise Theory** base de CFEngine-3
  `https://en.wikipedia.org/wiki/Promise_theory`

Comment cela fonctionne

Client / Serveur

- Système classique client serveur (multi-clients / multi-serveurs)
- Client se synchronise toutes les heures avec le(s) serveur(s) (phase **update**)
- Client fait sur lui-même un état des lieux et décide d'actions à faire

- Système **SplayTime** pour gérer l'engorgement sur le(s) serveur(s) (décalage aléatoire décidé par le client)
- Actions de base : copie de fichiers, édition de fichiers, création de liens, exécution de programmes, suppression de fichiers...
- Actions spécialisées : point de montage (`fstab`), gestion des paquetages...

Aucune action CFEngine spécialisée au LEGI, que des choses très basiques !

Comment cela fonctionne

Au LEGI

- CFEngine-2 depuis 2006 - **très grande stabilité** (du logiciel et des API), **compatibilité** dans le temps
- CFEngine-2 tourne sur Debian : Sarge (2005), Etch, Lenny, Squeeze, Wheezy, Jessie et Stretch (2017) (et les Ubuntu) à l'identique
- Toutes les variantes GNU/Linux : IA32, AMD64 et ARM64
- Même système de configuration pour tout le monde !

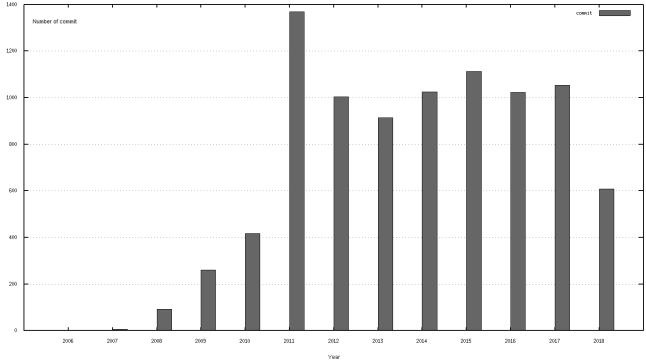
  Un anneau pour les contrôler tous...
- Portables, postes fixes, machines d'acquisition, serveurs, hyperviseurs, machines de calcul...
- Installation minimale via `net-install`, `preseed`...

  CFEngine-2 fait (presque) tout le reste

Comment cela fonctionne

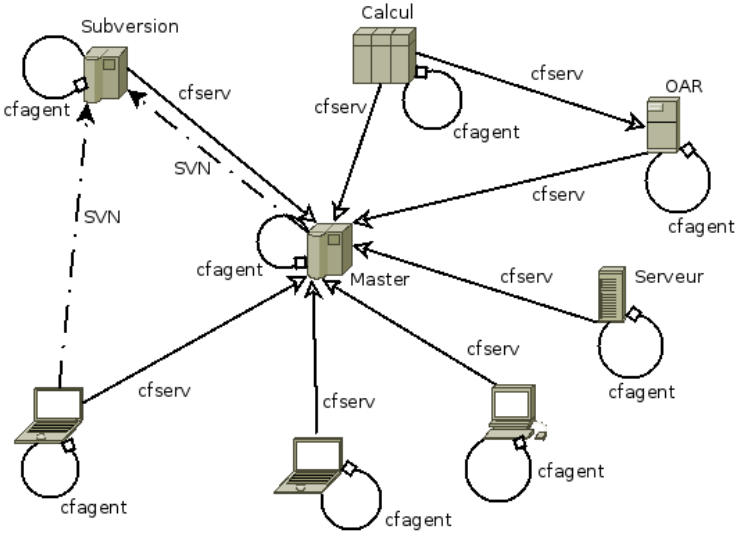
Au LEGI

- 12 ans de configuration Subversion (SVN)
- **8886 commits**
- Environ 1000 commit/an
- 1(+1) Serveurs CFEngine
- 177 Clients CFEngine



- Au début : modification directe sur le serveur CFEngine (erreur)
- Aujourd'hui : modification sur son poste de travail puis **commit**
- Le **serveur CFEngine** est un **client Subversion**
- Bonne organisation (travail en équipe), très **bonne tracabilité**
- Investissement dans l'apprentissage de l'outil rentabilisé depuis longtemps !

Comment cela fonctionne



Exemple simple de commande

Étape 1 : Copie de fichiers du serveur vers le client

```
copy:
   /srv/cfengine
      server=srv-cfengine.legi.grenoble-inp.fr
      dest=/var/lib/cfengine2/inputs
      recurse=inf
      mode=600
      owner=root
      group=root
      type=checksum
      ignore=.svn
```

Synchronisation du client avec le serveur si celui-ci est en ligne (portable)

Exemple simple de commande

Étape 2 : Copie locale de fichiers

```
copy:
   /var/lib/cfengine2/inputs/pub/usr/local/bin/hostname-in-group
      dest=/usr/local/bin/hostname-in-group
      mode=0755
      owner=root
      group=root
      type=byte
```

- Installation des fichiers aux bons endroits
- Gère les droits et la non modification des fichiers

Exemple simple de commande

Étape 3 : Édition et modification en ligne d'un fichier

```
editfiles:
   { /etc/fstab
   ResetSearch "1"
   DeleteLinesContaining "tobedeleted"
   }

   { /etc/inittab
   ResetSearch "1"
   CommentLinesMatching "5.23:respawn./sbin/getty$(spc)38400$(spc)tty5"
   CommentLinesMatching "6:23:respawn:/sbin/getty$(spc)38400$(spc)tty6"
   }
```

- Commande `editfiles` complexe mais très puissante
- Possibilité d'utiliser des variables "`$(ThisVariable)`" (systèmes ou personnelles)

Sous le capot

Actions de base

- `editfiles:` édition de fichiers
- `copy:` copie de fichiers
- `control:` définition de variables
- `links:` création de liens
- `groups:` **définir des classes**
- `shellcommands:` commande à exécuter
- `directories:` droit sur les dossiers
- `files:` droit sur les fichiers
- `import:` inclure un sous fichier de configuration
- `strategies:` permet de basculer aléatoirement entre plusieurs possibilités
- `tidy:` suppression de fichiers

Ordre des actions au LEGI : `actionsequence = ( directories files copy editfiles links shellcommands tidy )`

Sous le capot

Automate fini - Machine à états

- **De l'importance des classes** = état
- On ne **cherche jamais à atteindre un état** final F
- Philosophie générale : **si état A alors action X** et on boucle
- CFEngine se relance toutes les heures
- Magie globale : le **système converge** assez vite tout seul vers un état F **stable**

- Nulle part n'est écrite la configuration finale souhaitée
- Chaque machine se stabilise sur un état qui lui est propre (plus rien à faire)
- Pas forcément 2 machines identiques dans son parc !
- À chaque **modification** dans CFEngine, les **machines se stabilisent de nouveau** (si besoin)...

Promise Theory : `https://en.wikipedia.org/wiki/Promise_theory`

Sous le capot

Automate fini - Machine à états

- Attention à ne pas générer d'actions incompatibles - le système oscille alors entre deux (ou plus) états stables
- 787 classes au LEGI - Autant d'états de systèmes possibles !

Exemples de classes très simples

```
groups:
   MyClsServerGDM3  = ( FileExists(/etc/init.d/gdm3) )

   MyClsOSStretch   = ( "/bin/grep -q ^9. /etc/debian_version" )

   MyClsHostTeamNrj = ( "/usr/local/bin/hostname-in-group nrj" )
```

Par sécurité, CFEngine impose par défaut des **chemins absolus** pour les **exécutables**

Un exemple un peu plus complet : déactiver IPv6

```
groups:
   MyClsOSStretch   = ( "/bin/grep -q ^9. /etc/debian_version" )

editfiles:
   (MyClsOSSqueeze|MyClsOSWheezy|MyClsOSJessie|MyClsOSStretch)::
      { /etc/default/grub
      ResetSearch "1"
      LocateLineMatching  "^GRUB_CMDLINE_LINUX=.*"
      BeginGroupIfNoMatch  "GRUB_CMDLINE_LINUX=$(dblquote)ipv6.disable=1$(dblquote)"
         ReplaceLineWith   "GRUB_CMDLINE_LINUX=$(dblquote)ipv6.disable=1$(dblquote)"
      EndGroup
      DefineClasses "MyShlUpdateGrub"
      }

shellcommands:
   MyShlUpdateGrub::
      "/usr/sbin/update-grub"
```

- Les **classes** permettent de mettre des **conditions** sur les actions
- Une **action** permet de définir une **nouvelle classe** (exemple `MyShlUpdateGrub`)
- Le système **évolue** ainsi par **petites touches**...

Un exemple un peu plus complet : résolution DNS

- Gestion par fichier `resolv.conf`
- Gestion par édition du fichier `resolv.conf`
- Gestion par serveur DHCP

Gestion par fichier `resolv.conf`

```
groups:
   MyClsIpResolvByNM       = ( "/bin/grep -q 'Generated by NetworkManager' /etc/resolv.conf" )
   !MyClsIpResolvByNM::
      MyClsIpResolvByFile  = ( "/bin/egrep -q '^iface .* inet (dhcp|static)' /etc/network/interfaces" )

copy:
   (MyClsOSSarge|MyClsOSEtch|...|MyClsOSWheezy|MyClsOSJessie|MyClsOSStretch).MyClsIpResolvByFile::
      $(MyVarCFPub)/etc/resolv.conf
         dest=/etc/resolv.conf
         mode=0644
         owner=root
         group=root
         type=byte
```

Variable : `MyVarCFPub = /var/lib/cfengine2/inputs/pub`

Un exemple un peu plus complet : résolution DNS

Gestion par édition du fichier `resolv.conf`

```
groups:
   MyClsIpResolvStatic  = ( "/bin/egrep -q '^iface .* inet static' /etc/network/interfaces" )

editfiles:
   MyClsIpResolvStatic::
      { /etc/resolv.conf
      LocateLineMatching  "^nameserver$(spc).*"
      BeginGroupIfNoMatch  "nameserver 8.8.8.8"
         ReplaceLineWith   "nameserver 8.8.8.8"
      EndGroup
      }
```

Dans l'heure qui suit (plus le `SplayTime`),

toutes les machines qui ne sont pas en DHCP ont leur résolution DNS à jour !

Un exemple un peu plus complet : résolution DNS

Gestion par serveur DHCP

```
groups:
      MyClsConfigInCF = ( "/usr/bin/test -e /var/lib/cfengine2/inputs/etc/server/$(host).cfg" )

import:
   MyClsConfigInCF::
      $(MyVarCFInputs)/etc/server/$(host).cfg

------- file /srv/cfengine/etc/server/srv-dhcp.cfg

copy:
      $(MyVarCFPub)/etc/dhcp/dhcpd..host.$(host).conf
         dest=/etc/dhcp/dhcpd.conf
         mode=0644
         owner=root
         group=root
         type=byte
         define=MySrvRestartServer

shellcommands:
   MySrvRestartServer::
      "/etc/init.d/isc-dhcp-server restart"
```

Chaque serveur a son propre fichier de configuration (en complément des autres)

Un bref aperçu de la sécurité

- **Échange des clefs publiques** entre le client et le serveur (protocole symétrique) à la première connexion ou manuel (via SSH)
- Vérification à chaque connexion basée sur l'IP (problème avec les portables qui changent d'IP)
- **Copie serveur / client chiffrée**
- Possibilité de **limiter l'accès des dossiers** du serveur par IP
  - Accès de certaines arborescences accessibles à un seul serveur
  - Permet la distribution sécurisée des certificats SSL
  - Permet de limiter la diffusion de fichiers de configuration comportant un mot de passe
  - ...

Conclusion / Points positifs

- CFEngine-2 n'est pas forcément le bon outil pour commencer en 2018
  - Il existe CFEngine-3, Chef, Puppet, Ansible...
- Outil très stable avec une philosophie simple mais originale
- Outil s'adapte très bien à un parc très hétérogène (portable, serveur...)
- Point positif : pas de module avancé (sauf l'édition, tout ce que l'on fait est quasiment du copier coller de sa documentation Shell)
- Pas d'ordonnancement des actions (État A $\longrightarrow$ Actions)
  - On ne cherche pas à faire les actions menant de l'état A à l'état final F
  - Les systèmes convergent par eux-mêmes dans un état stable au bout de quelques itérations !

Conclusion / Problèmes - Difficultés

- Configuration en mode SQL dans le logiciel et non en mode fichier
  - Malheureusement mode actuelle suivant bêtement l'AD...)
  - Pas forcément facile à automatiser (cf OAR, LDAP, Samba...) avec CFEngine-2 (module SQL dans CFEngine-3)
  - En général, on a une configuration fichier et si celui-ci change, on pousse une action Shell sur le serveur
  - Pas toujours bien intégrée
- Version 2 périmée, plus d'évolution depuis des années
- CFEngine semble assez dépendant de son créateur (pas de grosse communauté des développeurs autour)

Il est inimaginable de gérer un parc (homogène ou non) sans outil centralisé et versionné de gestion des configurations en 2018 !

La configuration de l'OS de vos postes évolue très régulièrement au cours de leur vie