Utilisation du logiciel d'administration automatique Cfengine

—

version étendue
ENSTBr/INFO/RR/2001-011

Ronan KERYELL
14 novembre 2001

Résumé

Cet article présente le système d'automatisation des tâches d'administration système Cfengine avec comme illustration la mise en place d'un environnement logiciel permettant une automatisation de l'installation et de la maintenance d'un système d'exploitation de type UNIX d'un réseau d'ordinateurs. Cfengine est utilisé pour la mise en place et la correction de l'état du système à partir d'un langage déclaratif.

Le système exposé est déployé au sein de deux laboratoires de recherche en informatique à l'ENSMP et à l'ENSTBr pour la gestion de machines génériques ou de machines dédiées à un projet de recherche sur les réseaux actifs.

1 Introduction

L'installation automatique des ordinateurs est un problème récurrent et naturel pour les informaticiens car, par essence, l'informatique est la science de l'automatisation et il ne s'agit là que de l'appliquer à elle-même : « *boot-straper* » l'outil lui-même en allant jusqu'à automatiser le déploiement du système d'automatisation. Quoi de plus frustrant en effet que d'installer une machine quand on sait le faire et que cela devient chronophage avec le nombre de machines à installer ?...

Le Centre de Recherche en Informatique de l'ENSMP et le Laboratoire Informatique et Télécommunication de l'ENSTBr sont spécialisés en parallélisation, compilation, optimisation de programme et en distribution. Il est donc logique que ces technologies soient appliquées à l'outil informatique lui-même, c'est à dire à l'organisation système, aussi bien des machines des chercheurs, des élèves, des ordinateurs parallèles ou des projets de recherches plus spécifiques comme les réseaux actifs, ou encore des machines personnelles, utilisées pour le télétravail ou autre, avec le déploiement d'accès personnels rapides à Internet tel que ADSL.

Il existe de nombreuses méthodes d'installation automatique pour différents systèmes d'exploitations mais, afin de réduire les efforts de développement, nous proposons une méthode en deux phases :

- utiliser la méthode d'installation spécifique à un système d'exploitation de manière minimale pour avoir un système tout juste fonctionnel ;
- utiliser une méthode générique portable pour terminer l'installation et maintenir le système dans un état fonctionnel, c'est à dire automatiser l'administration système.

L'idée est qu'on va factoriser les efforts dans la partie générique de l'installation. Cette partie peut être réalisée avec n'importe quel langage permettant d'interagir avec le système (typiquement des *shell-script*) mais nous utiliserons ici un langage déclaratif permettant de classifier les machines en se rapprochant d'une méthodologie objet : GNU/Cfengine.

Dans cette étude on écarte donc les systèmes qui n'ont pas (encore) de systèmes d'installation automatique, ne seraient-ce que basiques.

La suite de l'article présente le système Cfengine (§ 2) suivi d'un résumé de la manière dont il est utilisé au LIT (§ 3). Quelqu'un intéressé par les potentialités de ce type de système pourra regarder directement l'exemple donné dans cette dernière partie afin de se donner une idée.

2 Le logiciel Cfengine

Cfengine est un logiciel développé depuis 1993 principalement par Mark BURGESS [Bur01a, Bur01d] dans le cadre d'un projet de recherche concernant l'administration système distribuée et automatisée. Son usage actuel est estimé à une centaine de milliers de nœuds dans le monde Unix et Windows.

Le concept de base en est la gestion de la configuration des machines à base d'une politique établie sous formes de règles permettant de centraliser un comportement d'assez haut niveau plutôt que d'avoir à définir les tâches en détail pour tous les cas possibles de machines.

L'intérêt du modèle est de définir des actions à effectuer en fonction des états du système (tel service ne marche pas, un disque est plein) plutôt que d'avoir à définir toutes les transitions possibles entre les états (que faire pour passer de l'état (service ne marche pas, répertoire plein) à l'état (service en marche, disque avec de la place libre), ce qui voudrait dire prévoir le produit cartésien des états. Cela permet d'appréhender les évolutions futures de son architecture d'administration et de son parc de machines avec une complexité contrôlable.

L'initiateur de ce projet aime le comparer à un système immunitaire où le système réagit par une série d'actions lorsqu'il rencontre telle ou telle situation anormale et progressivement le système doit revenir vers l'état stable pleinement fonctionnel, si possible.

2.1 Composants constituant Cfengine

Cfengine est en fait constitué principalement de plusieurs composants [Bur01c, Bur01b] :

- le système de base :
  - l'ensemble des fichiers de configuration définissant le comportement à adopter, typiquement localisés dans le répertoire `/usr/local/share/cfengine` reflété par la valeur de la variable d'environnement `CFINPUTS`. La configuration y est stockée dans le fichier `cfengine.conf` ;
  - l'exécutant de Cfengine `cfagent` doit être lancé régulièrement pour effectuer le travail ;
- des systèmes optionnels :
  - un serveur sécurisé de configurations `cfservd` permettant à des `cfagents` distants de récupérer leur configuration s'ils ne peuvent pas y accéder par d'autres moyens. Il permet aussi des exécutions à distance de `cfagents` ;
  - un démon centralisant l'état global du système afin de faire des statistiques sur le fonctionnement du système ;
  - un outil de représentation graphique de l'état du système.

Les noms des exécutables dans la liste précédente concernent la version 2 de Cfengine et non l'ancienne version.

Il existe un mode `cfengine.el` pour EMACS qui permet de rajouter des couleurs et gérer l'indentation des fichiers de configuration pour faciliter encore plus la tâche de l'administratrice(-eur).

2.2 Classes et environnement

La configuration d'une machine va dépendre de nombreux aspects :
- son rôle dans l'entreprise ;
- sa localisation géographique ;
- le réseau qui la contient ;
- sa structure matérielle :
  - architecture ;
  - système d'exploitation ;
  - disques disponibles ;
  - type de réseau local ;
  - écran disponible ;
  - ...

Il est clair que le nombre de possibilités est énorme et la manière de gérer cette complexité dans Cfengine est de raisonner par classes de machines suivant différents critères plus ou moins orthogonaux.

2.2.1 Classes « matérielles »

Comme Cfengine tourne sur chaque machine qui doit être administrée de cette manière, la classification d'une machine est faite au moment de l'exécution de `cfagent` et une liste d'attributs qui vont déclarer l'appartenance à autant de classes. Par exemple pour la machine `rodomouls.enst-bretagne.fr` différentes classes seront automatiquement définies :
- l'identité de la machine définit des classes de noms (`rodomouls_enst_bretagne_fr`, `enst_bretagne_fr`, `rodomouls`), d'adresses (`192_44_75_24`), réseau (`192_44_75`) ;
- le système d'exploitation et l'architecture matérielle définissent des classes : `solaris`, `32_bit`, `sunos_5_8`, `sunos_sun4u`, `sunos_sun4u_5_8`, `sunos_sun4u_5_8_Generic_108528_09`, `solaris2_8` d'une part et `sparc`, `sun4u` d'autre part ;
- le temps et la date définissent des classes temporelle (`Thursday`, `Hr17`, `Min46`, `Min45_50`, `Day20`, `September`, `Yr2001`). L'idée est de pouvoir éventuellement changer de comportement au cours du temps ;
- une classe `any` à laquelle toute machine appartient.

2.2.2 Classes utilisateur

Évidemment les classes précédentes sont de bas niveau et il est indispensable de définir de nouvelles classes à partir de ces dernières ou de définition de *netgroup* UNIX pour avoir une configuration propre et gérable.

2.2.2.1 Algèbre booléenne sur les classes

Partout où une classe peut être utilisée on peut en fait utiliser une expression booléenne de classes avec les opérateurs classiques et les parenthèses.

2.2.2.2 Déclaration explicite

La manière la plus simple est de définir l'appartenance à des classes en dur dans une action de `control` :

```
addclasses = ( mon-labo France )
```

L'intérêt peut être aussi qu'on pourra sortir exceptionnellement d'une classe via une option de ligne de commande (§ 2.2.2.3).

Une manière moins triviale de définir l'appartenance à une classe est d'utiliser l'action `classes` ou son synonyme `groups` qui permet de définir des classes à partir d'opérations d'union et d'intersection sur d'autres classes ou des `netgroup` UNIX comme dans :

```
classes:
  apache_hosts = ( +@n_web www.enstb.org -gavotte.enst-bretagne.fr)
  cri_sun = ( +@n_cri_sun )
  sun_sans_serveur_WWW = ( +@n_cri_sun -@n_web )
```

L'intérêt de récupérer les `netgroup` est d'éviter la redondance dans la classification, dans la mesure où on est déjà obligé de faire la part des choses pour gérer les droits d'exportation NFS, etc.

2.2.2.3 Contrôle des classes par ligne de commande

On peut définir explicitement l'appartenance à une classe en lançant `cfagent` avec une option `-D` ou au contraire en soustrayant l'appartenance à une classe avec l'option `-N`.

L'intérêt est de pouvoir changer le comportement de Cfengine ponctuellement, comme par exemple l'incantation que nous utilisons lors de la fin de l'installation automatique d'UNIX

```
cfagent -DInstallationTime
```

qui permet de spécialiser le comportement des règles pour tenir compte du fait qu'on est en pleine installation du système et que certaines choses sont superflues alors que d'autres sont à faire en plus.

2.2.2.4 Appartenance à une classe par succès d'une commande

Un autre moyen d'appartenir ou pas à une classe est de tester le résultat d'une ou plusieurs commande telle que :

```
classes:
   have_cc = ( "/bin/test -x /usr/ucb/cc"
               "/bin/test -x /local/gnu/cc" )
```

La classe `have_cc` sera vraie si au moins un des `test` a renvoyé vrai.

2.2.2.5 Classes de retour d'action

Afin d'avoir des comportements mettant en œuvre la causalité, une action peut définir un ensemble de classes si elle est exécutée, permettant ainsi l'exécution d'autres classes par la suite.

Pour ce faire il suffit de rajouter la liste des classes à définir dans une action avec :

```
define=liste-de-classes
```

Ce concept est le moteur principal de Cfengine.

2.2.2.6 Classes définies par des modules

L'exécution d'un module peut définir des classes parmi celles indiquées dans l'`actionsequence` (§ 2.5) en envoyant sur la sortie standard autant de lignes de type :

```
+ma-classe
```

que nécessaire.

2.3 Variables

Comme tout langage, Cfengine comporte la notion de variables, certaines prédéfinies par le système et évidemment celles définies par l'utilisateur.

Les variables s'utilisent avec la syntaxe `$(`*`variable`*`)`.

2.3.1 Variables prédéfinies

On peut ranger les variables prédéfinies en différentes catégories dont voici les plus importantes :

- des éléments de nommage de la machine :

  `host` : le nom de la machine ;

  `fqhost` : son nom complètement qualifié au sens du DNS ;

  `ipaddress` : son adresse IP numérique ;

  `domain` : son domaine ;

  `faculty` **ou** `site` : contiennent indifféremment le nom de l'entité de l'entreprise défini dans une action de `control` ;

- des indications temporelles :

  `date` : la date courante ;

  `year` : l'année courante ;

  `timezone` : le fuseau horaire tel qu'il a été défini dans une action de `control` ;

- l'adresse électronique de l'administrateur système `sysadm` ;
- la liste des classes :

  `AllClasses` : contient toutes les classes auquel appartient l'instance de Cfengine à un instant donnée sous la forme `CFALLCLASSES=`*`class1`*`:`*`class2`*`...`. Cela permet de l'exporter simplement à des *shell-scripts* par exemple ;

  `class` : contient la liste des classes prédéfinies ;

- des variables syntaxiques contenant des constantes textuelles de type caractère pour se simplifier la vie dans la définition de chaînes de caractères :
  - de caractères de contrôle `cr`, `lf`, `n` (nouvelle ligne), `tab` ;
  - des guillemets avec `quote` et `dblquote` ;
  - et d'autres comme `spc` et `dollar`.

2.3.2 Variables utilisateur

Elle sont déclarées dans une section de `control` avec la syntaxe suivante :

```
variable = ( contenu )
```

le *`contenu`* pouvant être ou non entre guillemets quelconques.

Un autre moyen de déclarer des variables est qu'un module (§ 2.5) renvoie des chaînes de type :

```
=variable=valeur
```

2.4 Actions

On peut découper les tâches de Cfengine en sous-tâches qui vont être rangées par action thématique. Une section d'actions a la structure générique :

```
thème:
  classe::
    action1
    action2
    ...
```

pour déclarer des actions *action1* et *action2* définies dans le thème d'action *thème* si la machine appartient à la classe *classe*.

Par exemple

```
links:
   solaris.apache_hosts::
      # To have apache starting at boot time:
      /etc/rc2.d/K15apachectl -> /etc/init.d/apachectl
      /etc/rc3.d/S85apachectl -> /etc/init.d/apachectl
```

définit 2 actions de type « créer un lien s'il n'existe pas » si on est sous SOLARIS et qu'on est une machine qui doit faire tourner APACHE qui ont pour effet de créer les liens sous `/etc/rc2.d` et sous `/etc/rc3.d` pour faire démarrer automatiquement le serveur WWW lorsque le système exporte ses ressources à l'extérieur.

Il y a beaucoup de types d'actions et chaque type a de nombreux paramètres et il ne saurait être question de les détailler tous ici [Bur01b]. Seuls les principaux seront montrés avec quelques exemples afin d'en dévoiler l'esprit.

En générale une action peut être configurée pour corriger un problème, définir une classe le cas échéant, ou tout simplement renvoyer un avertissement ou une erreur si telle ou telle condition n'est pas vérifiée.

2.4.1 Actions de contrôle

Il ne s'agit pas là à proprement parler d'actions concrètes mais du contrôle du comportement global de Cfengine.

L'action `import` est la plus simple et permet d'inclure d'autres fichiers de configuration pour mieux structurer la configuration :

```
import:
   $(cf_directory)/main.cf
   $(cf_directory)/accounting.cf
```

L'action `control` est l'auberge espagnole de Cfengine avec les déclarations de variables, des déclarations de classes, des motifs d'exclusion, etc.

```
control:
   cf_directory = ( /usr/local/share/cfengine/cf )
   LIT::
      site      = ( LIT )
      domain    = ( enst-bretagne.fr )
      sysadm    = ( Ronan.Keryell@enst-bretagne.fr )
   any::
      timezone  = ( MET )
```

2.4.2 Thème contrôlant les fichiers

Comme sous UNIX tout est fichier, il est normal que la majorité des actions de Cfengine soit tournée vers les fichiers.

Une première action est de vérifier et corriger la présence et les droits de fichiers et éventuellement de les créer avec `files` :

```
files:
   solaris::
      # Create this file to log the login failures:
      /var/adm/loginlog mode=600 owner=root group=sys action=touch
   any::
```

```
      home mode=o-w r=inf act=fixall
      home/public_html mode=a+r fixall
```

Un cas particulier d'action sur les répertoires est géré par `directories` comme dans :

```
directories:
   # Create the local mount points per machines:
   LIT.rodomouls::
      /export/burette
      /export/dinette
      /export/pipette
   LIT.gavotte::
      /export/interne
      /export/calice1
      /export/calice2
      /export/calice3
```

Une tâche fastidieuse lors de l'installation de systèmes est qu'il faut souvent aller faire des corrections dans de nombreux fichiers pour se trouver dans la configuration voulu. C'est automatisé avec les actions du thème `editfiles` comme suit :

```
editfiles:
   solaris::
      {
         # Log all the login failures:
         /etc/default/login
         AppendIfNoSuchLine "SYSLOG_FAILED_LOGINS=0"
      }
      {
         # Add accounting jobs to the adm crontab:
         /var/spool/cron/crontabs/adm
         AutoCreate
         AppendIfNoSuchLine "# Accounting stuff:"
         AppendIfNoSuchLine "0 * * * * /usr/lib/acct/ckpacct"
         AppendIfNoSuchLine
            "30 2 * * * /usr/lib/acct/runacct 2> /var/adm/acct/nite/fd2log"
         AppendIfNoSuchLine "30 7 1 * * /usr/lib/acct/monacct"
         DefineClasses "StartAccounting"
         DefineClasses "RestartCron"
      }
      # To mount the file systems with logging enabled to eradicate fsck.
      { /etc/vfstab
         # To use logging ufs file systems:
         ReplaceAll '^(/dev/.*[ $(tab)]ufs[ $(tab)].*)-$' With '\1logging'
      }
   solaris.LIT::
      # Use DNS and files :
      { /etc/nsswitch.conf
         SetCommentStart '#'
         CommentLinesStarting 'hosts:      nis [NOTFOUND=return] files'
         CommentLinesStarting 'hosts:      files'
         CommentLinesStarting 'hosts:      files dns'
         AppendIfNoSuchLine '# Put the DNS in first place to have FQHN. RK.'
         AppendIfNoSuchLine 'hosts:     dns files'
      }
```

Les directives d'édition sont nombreuses et plutôt spécialisées dans les éditions incrémentales les plus couramment utilisées en administration système.

Le contenu d'un fichier ou d'une arborescence plus ou moins complète peuvent aussi être récupérés depuis une autre arborescence ou un serveur `cfservd` distant :

```
copy:
   UserDiskServer::
      /local/masterfiles/.cshrc  dest=home/.cshrc mode=0600
   solaris.OpenSSH::
      $(shared_conf)/OpenSSH/etc/openssh/ssh_config
         dest=/etc/openssh/ssh_config
         type=byte
         define=InstallSsh
```

qui copiera régulièrement des `.cshrc` chez tous les utilisateurs sur les machines faisant office de serveurs de disques et installera un morceau de la base

de données anti-*spam* de `sendmail` sur toutes les machines depuis un répertoire de référence.

Les liens symboliques ou durs peuvent être créés s'ils n'existent pas avec l'action `links` :

```
links:
   # Add the entry to launch pppd at boot time:
   solaris.pppd_hosts::
      /etc/rc3.d/S90pppd ->! /etc/init.d/pppd
```

ou encore, de manière plus intéressante, on peut faire des liens vers tous les fichiers de plusieurs arborescences, par exemple pour fournir à l'utilisateur un simple `/usr/local/bin` qui serait une union de nombreux répertoires.

Même grands, les disques ne le sont jamais assez et il peut être nécessaire de faire du ménage avec la primitive `tidy` de manière plus ou moins profonde :

```
tidy:
   AllHomeServers::
      home     pattern=core   R=inf age=0
      home     pattern=*~     R=inf age=7
      home     pattern=#*     R=inf age=30
   any::
      /tmp/    pat=*            R=inf   age=1
      /        pat=core         R=2     age=0
      /etc     pat=hosts.equiv  r=0     age=0
```

Certains fichiers peuvent être dangereux mais on ne veut pas les effacer pour autant. L'action `disable` permet de changer de nom au fichier voire de gérer la notion de versions :

```
disable:
   any::
      etc/hosts.equiv
   WWW.(Sunday|Wednesday)::
      /usr/local/httpd/logs/access_log  rotate=10
```

Certains fichiers ou répertoires doivent échapper aux copies ou aux nettoyages. Cela est précisé avec l'action `ignore` :

```
ignore:
   any::
      # Prevent tidying .X11 directories in /tmp
      .X11
      # Don't tidy emacs locks
      !*
      /local/lib/gnu/emacs/lock/
```

2.4.3 Thèmes contrôlant les processus

Les actions de type `shellcommands` permettent d'exécuter des commandes depuis Cfengine :

```
shellcommands:
   AllBinaryServers.sun4.Saturday::
      "/usr/etc/catman -w -M /usr/local/man"
      "/usr/etc/catman -w -M /usr/local/X11R5/man"
   solaris.LaunchExportfs::
      "/usr/sbin/shareall"
```

Mais un contrôle des processus en train de tourner est faisable grâce aux actions `processes` :

```
processes:
   # At least one httpd process should run,
   # except during installation of course...
   solaris.apache_hosts.!InstallationTime::
      "/usr/local/apache/bin/httpd"
         matches=>0
         restart "/etc/init.d/apachectl start"
   RestartSyslogd::
      # Tell syslogd to read again its configuration:
      "/usr/sbin/syslogd" signal=hup
```

2.4.4 Thèmes de type macro

Les thèmes de type macro permettent de déclarer des objets qui sont utilisables par d'autres actions :

- les thèmes `classes` et `groups` qui sont synomymes et permettent de déclarer de nouvelles classes et sont discutés en § 2.2.2.2 ;
- `acl` permet la déclaration d'ACL (*Access Control List*) de manière portable et de les réutiliser dans les actions comme droits d'accès plus précis ;
- `filters` va déclarer des contraintes, réutilisables ailleurs pour modulariser le code, telles que des contraintes sur le temps, des tailles, des expressions régulières sur les noms, des modes et des droits d'accès, des aspects concernant des processus.

2.4.5 Thèmes de configuration réseau

La configuration réseau d'une machine est contrôlée par un certain nombre d'actions.

L'action `broadcast` spécifie si les adresses de diffusion se terminent par des « 0 » ou des « 1 » binaires.

L'action `interfaces` permet de spécifier par interface réseau le masque et le type de diffusion de manière plus spécifique que dans l'action de `control`.

L'action `defaultroute` permet de vérifier que la route par défaut est correcte. Cfengine n'est pas capable de la corriger, seul un avertissement signalera le problème.

Enfin l'action `resolve` permet de configurer automatiquement l'usage du DNS.

2.4.6 Thèmes de configuration des disques

Dans un système d'exploitation la persistance des données est assurée par des disques durs et il est primordial qu'ils soient bien gérés. De ce fait Cfengine propose de nombreuses actions pour aider leur gestion.

Les synonymes `disks` et `required` permettent de déclencher des actions s'il n'y a plus assez de place ou que des points de montage ont disparu.

Ensuite il y a plusieurs classes qui n'ont de sens que si on utilise l'organisation des montages de disques de l'université de l'auteur de Cfengine, ce qui est discutable. `mountables` déclare les ressources disques réseau utilisables, `binservers` et `homeservers` permettent de choisir parmi ces disques lesquels seront montés comme disque contenant respectivement des binaires et des répertoires d'utilisateurs. `mailserver` spécifie le disque réseau à utiliser pour lire le courrier. `unmount` permet d'effectuer des démontages.

En général une solution à base d'automonteur remplacera avantageusement ce système.

On peut cependant faire des montages explicites généraux en utilisant les actions `miscmounts` :

```
miscmounts:
   physics::
      libraryserver:/$(site)/libraryserver/data2
         /$(site)/libraryserver/data2 mode=ro
```

2.5 Séquence d'actions

Pour que le découpage en action garde une certaine causalité, il est nécessaire d'ordonner les actions suivant une sertaine séquence. Par exemple une fois qu'on a modifié le fichier de configuration d'un démon il peut être nécessaire de notifier ce dernier avec un signal quelconque ou de le relancer. Il est clair que l'inverse empêchera la prise en compte de la nouvelle configuration.

Cet ordonnancement est capital dans Cfengine et est spécifié via la primitive de `control actionsequence` comme par exemple :

```
control:
  any::
    actionsequence = (
      netconfig
      resolve
      checktimezone
      directories
      files
      copy
      editfiles
      links
      tidy
      module:mytests.class1.class2.class3 arg1 arg2
      shellcommands
      processes
      # At least for the Minitel access installation:
      copy.minitel
      editfiles.minitel
    )
```

On peut voir que certaines actions doivent être répétées (comme ici `copy`) pour atteindre l'effet voulu à cause des dépendances. On peut définir des classes supplémentaires à la volée. En particulier `editfiles.minitel` signifie que l'on va réexécuter toutes les actions du thème `editfiles` mais en appartenant en plus à la classe `minitel`.

On peut étendre les actions disponibles avec la notion de `module`. Dans l'exemple précédent on a rajouté l'exécution d'un programme externe qui aura 2 arguments et pourra définir les 3 classes indiquées. Le module hérite des variables d'environnement de `cfagent`.

3 Mise en œuvre

3.1 Hypothèses

Les hypothèses de travail pour une telle architecture sont :

- la simplicité afin d'avoir un système utilisable par un maximum de personnes, éventuellement sans compétence système sans faire peur outre mesure ;
- la portabilité d'un point de vue :
  - du système d'exploitation utilisé (SOLARIS, GNU/LINUX/DEBIAN, GNU/LINUX/REDHAT,...) et du modèle d'ordinateur (SUN, PC,...) ;
  - localisation et entité administrative, laboratoire, entreprise, maison ;
  - utilisation de machines : que ce soit en réseau ou pas, une machine utilisateur ou plutôt de type serveur, spécialisée comme des nœuds d'une machine parallèle, embarquée comme pour un pare-feu ou un nœud de réseau actif, des serveurs WWW stockés chez un hébergeur et sur lesquels il est difficile d'intervenir physiquement,...

- un référentiel centralisé pour avoir une vision cohérente du système et faciliter l'installation du système au départ ou en cas de gros dégât. La mise à jour à plusieurs de ce référentiel permet de partager et mettre en commun les expériences des administrateurs ;
- la tolérance aux pannes en ayant un nombre arbitraire d'instances du système d'installation disponible ;
- on suppose une confiance mutuelle entre les administrateurs des différents sites (car il est clair qu'il est facile pour un site mal-intentionné de rajouter des trous de sécurité, de détruire le système,...). Cela peut se concevoir par exemple si les sites travaillent sur des projets communs ou partages des membres (c'est par exemple le cas de l'auteur de cet article actuellement mis à disponibilité par l'ENSMP à l'ENSTBr) et que l'intérêt à factoriser les efforts est supérieur aux risques encourus.

D'autres hypothèses ne sont pas directement liées à l'infrastructure d'installation présentée mais à l'utilisation du système qui est faite et donc permet de mieux comprendre les exemples qui sont donnés dans la suite :

- avoir un espace de nommage unique quelle que soit la machine dans un domaine administratif : un chercheur ou un élève doit pouvoir accéder à son compte et travailler depuis n'importe quelle machine quelle que soit la localisation physique du compte ;
- exploiter toutes les ressources matérielles disponibles, en particulier les disques durs, pour améliorer les capacités et augmenter la tolérance aux pannes ;
- supprimer le maximum de points uniques de panne possibles par exemple en ayant plusieurs routeurs de courrier électronique, quitte à utiliser de la place disque et des machines en plus pour le faire ;
- distribuer le plus possible les services sur les différentes machines pour une meilleure tolérance aux pannes même pour des services n'existant qu'à un exemplaire : si une machine tombe en panne, seul le compte d'un ou de quelques utilisateurs sont indisponibles jusqu'au transfert ou restauration sur une autre machine par exemple ;
- préférer des approches logicielles pour assurer la tolérance aux pannes (ne pas utiliser par exemple de RAID matériel si on peut mettre plusieurs machines pour faire un serveur NFS redondant) car cela permet de tirer profit du seul matériel existant ;
- utiliser un système de sauvegarde en réseau pour sauvegarder chaque nuit toutes les informations sensibles (en utilisant ici le logiciel libre AMANDA pour éviter entre autre de payer des licences par machine sauvegardées).

Toutes ces dernières hypothèses ont poussé la mise en place de l'automatisation du système.

Le système d'installation est utilisé pour le déploiement automatique d'un réseau actif pour la distribution de vidéo et de contenu multimédia à l'ENSTBr dans le cadre du projet RÉACTIVE. Chaque routeur est en fait un ordinateur sous GNU/LINUX qui, outre le rôle de routeur IP classique, sert de cache, de stockage et de serveur de contenu réparti sur tous les disques des nœuds du réseau, chacun à chaque étage de chaque bâtiment de la résidence des élèves de l'ENSTBr. L'installation automatique est faite par vague à partir d'une référence accessible au premier nœud du réseau. Une fois ce premier nœud installé automatiquement avec sur celui-ci un serveur d'installation, ses voisins vont être installés à partir de celui-ci. Ces derniers serviront à installer eux-mêmes leurs

voisins et ainsi de suite jusqu'à avoir un réseau fonctionnel complet.

3.2 Architecture

3.2.1 Structure du référentiel

Afin de suivre l'évolution du système, tous les fichiers sont gérés par le système de gestion de version RCS, très simple à utiliser par exemple sous EMACS. L'intérêt supplémentaire est qu'en utilisant l'entête RCS, on sait où se trouve le fichier d'origine à modifier, car il n'est pas toujours simple de traquer l'origine d'un fichier installé automatiquement dans `/etc` par exemple.

Toute la configuration du système se trouve dans 2 répertoires de référence dans `/usr/local/share` :

`cfengine` : contient la configuration de base de Cfengine

- `cfengine.conf` : il s'agit du fichier de configuration de Cfengine qui se contente en fait d'inclure tous les fichiers du répertoire suivant ;
- `cfengine/cf` : pour plus de clarté les différents fichiers de configurations sont dans ce sous-répertoire, rangés par fonction ou thème plutôt que d'être dans quelques fichiers monolithiques comme le sont les exemples fournis avec la distribution de Cfengine ;

`conf` : contient tous les fichiers de référence nécessaire à configurer correctement le système, tels que description des imprimantes, fichiers réseaux, tables gérant le courrier, etc. La philosophie choisie a été de ranger les choses par ordre d'application importante, puis par site, par système d'exploitation et enfin par répertoire de destination des fichiers à installer. Ce qui donne de manière non exhaustive :

- `CRI` : répertoire contenant les fichiers généraux spécifiques au CRI ;
- `etc` répertoire contenant ce qui va génériquement dans `/etc` ;
- `linux` : hiérarchie spécifique à l'installation de LINUX, autre que les configurations d'application importante
  - `etc` : hiérarchie `/etc` spécifique à LINUX ;
  - `fai` : rassemble les fichiers impliqués dans la procédure d'installation automatique FAI de DEBIAN ;
- `LIT` : répertoire contenant les fichiers généraux spécifiques au LIT ;
- `log` : contient les informations sur les transferts multi-site du référentiel de configuration ou d'autres hiérarchies. Outre le fait d'améliorer la traçabilité de manière évidente, cela permet aux administrateurs des différents sites de se tenir au courant des choses modifiées par les administrateurs distants ou encore d'informer les utilisateurs ou autres administrateurs d'un site local des modifications apportées à leur système, les nouveaux logiciels disponibles,...

  Ce répertoire est classé par relations inter-site. Par exemple :

  `CRI-LIT` contient la liste des fichiers mis à jour depuis le CRI vers le LIT. ainsi le fichier `2001.07.20-usr-local-src` contiendra les fichiers synchronisés dans la partition `/usr/local/src` le `20/7/2001` et `2001.07.26-usr-local` ceux synchronisés dans la partition `/usr/local` le `26/7/2001` ;

`LIT-CRI` contient les mêmes fichiers en ce qui concerne la mise à jour depuis le LIT vers le CRI ;

`mail` : ce qui est nécessaire au bon fonctionnement du courrier électronique, à savoir une partie commune et quelques définitions plus spécifiques qu'on va trouver dans les répertoires `generic`, `CRI` et `LIT` ;

`ppp` : fichiers spécifique à un accès PPP ;

`ssh` : fichiers spécifiques à l'installation de `ssh` ;

`SunOS5` : fichiers spécifiques à l'installation des machines sous SOLARIS hormis les configurations d'application importante ;

`etc` : hiérarchie `/etc` spécifique à SOLARIS ;

`jumpstart` : rassemble les fichiers impliqués dans la procédure d'installation automatique de SOLARIS. Les fichiers contenant les images du système lui-même sont ailleurs, hors de la hiérarchie `/usr/local/share` pour des raisons de place.

La synchronisation des référentiels est actuellement lancée à la main de manière régulière d'abord en mode seulement verbeux étant donné les risques inhérents en cas d'impondérable. Cela permet de suivre de près l'évolution des fichiers installés sur les différents site et de détecter les erreurs grossières, des répertoires temporaires oubliés lors d'une compilation qu'il faudrait nettoyer... Une fois que les différences sont inspectées, la synchronisation réelle est lancée.

3.2.2 Extrait de la configuration Cfengine

Contrairement à la méthode utilisée par l'auteur de Cfengine le fichier de configuration a été de le hiérarchiser par thèmes fonctionnels plutôt que par architectures de machines.

```
control:
   # Where all the configuration files for cfengine are:
   # Will be /usr/local/share/cfengine/cf soon
   cf_directory = ( /usr/local/share/cfengine/cf )
import:
   # Split things up to keep things tidy:
   # The main file...
   $(cf_directory)/main.cf
   # and all the other files sorted by function:
   $(cf_directory)/accounting.cf
   $(cf_directory)/accounts.cf
   $(cf_directory)/apache.cf
   $(cf_directory)/automount.cf
   $(cf_directory)/cfengine.cf
   $(cf_directory)/cron.cf
   $(cf_directory)/exportfs.cf
   $(cf_directory)/holidays.cf
   $(cf_directory)/localmounts.cf
   $(cf_directory)/logging.cf
   $(cf_directory)/minitel_access.cf
   $(cf_directory)/naming.cf
   $(cf_directory)/network.cf
   $(cf_directory)/ntp.cf
   $(cf_directory)/patch.cf
   $(cf_directory)/ppp_access.cf
   $(cf_directory)/printing.cf
   $(cf_directory)/sendmail.cf
   $(cf_directory)/solaris.cf
   $(cf_directory)/ssh.cf
```

Alors qu'au CRI on utilise encore les NIS, pour des raisons de tolérance aux pannes au LIT, on n'utilise plus de système de nommage global si ce n'est le DNS, ou plutôt on utilise Cfengine pour propager les fichiers de nommage via `naming.cf` et faire notre propre système de nommage robuste :

```
 editfiles:
   solaris.CRI::
      # Use DNS and NIS :
      { /etc/nsswitch.conf
         SetCommentStart '#'
         CommentLinesStarting 'hosts:      nis [NOTFOUND=return] files'
         CommentLinesStarting 'hosts:      files'
         CommentLinesStarting 'hosts:      files dns'
         AppendIfNoSuchLine '# Put the DNS in first place to have FQHN. RK.'
         AppendIfNoSuchLine 'hosts:     dns nis files'
      }

   solaris.LIT::
      # Use DNS and files :
      { /etc/nsswitch.conf
         SetCommentStart '#'
         CommentLinesStarting 'hosts:      nis [NOTFOUND=return] files'
         CommentLinesStarting 'hosts:      files'
         CommentLinesStarting 'hosts:      files dns'
         AppendIfNoSuchLine '# Put the DNS in first place to have FQHN. RK.'
         AppendIfNoSuchLine 'hosts:     dns files'
      }

copy:
   solaris::
      # Set up the host file :
      $(shared_conf)/$(site)/etc/hosts
         dest=/etc/inet/hosts
         type=byte

      # Set up the netgroup file :
      $(shared_conf)/$(site)/etc/netgroup
         dest=/etc/netgroup
         type=byte

resolve:
   # Declare the DNS servers to use :
   CRI::
      193.48.171.40
      193.48.171.215
      193.48.180.100

   LIT::
      192.44.75.10
      192.108.115.2
      192.44.77.1
```

Du coup, les fichiers déclarants les comptes doivent être propagés, ce qui est simplement fait avec `accounts.cf` :

```
editfiles:
   solaris::
      # Authorized root to remote login as root.
      # Useful only for desesperated case...
      # That means that in normal circumstance, this should never be
      # used except with ciphered methods (ssh...).
      { /etc/default/login
         CommentLinesStarting "CONSOLE=/dev/console"
      }

copy:
   any::
      $(shared_conf)/$(site)/etc/passwd
         dest=/etc/passwd
         type=byte

      $(shared_conf)/$(site)/etc/shadow
         dest=/etc/shadow
         type=byte

      $(shared_conf)/$(site)/etc/group
         dest=/etc/group
         type=byte

      # Authorized log in without password from machines of the group:
```

```
      $(shared_conf)/$(site)/etc/hosts.equiv
         dest=/etc/hosts.equiv
         type=byte

      # Root can log in from administration machines:
      $(shared_conf)/$(site)/rhosts
         dest=/.rhosts
         type=byte
```

Il en est de même pour toutes les ressources gérées par le système de nommage, comme les tables d'automontage, etc.

Le cas de `sendmail` est intéressant car son installation personnalisée est réputée compliquée mais est automatisée par le `sendmail.cf` :

```
# Sendmail installation and configuration.
groups:
   mail_servers = ( gavotte.enstb.org smtp-cri.ensmp.fr )

copy:
   any::
      # Install directly the files from the server since
      # sendmail is launched before automount.
      # This adds also better fault tolerance.
      /usr/local/sbin/sendmail
         dest=/usr/lib/sendmail
         type=byte
         define=RelaunchSendmail


      /usr/local/sbin/editmap
         dest=/usr/sbin/editmap
         type=byte

      /usr/local/sbin/makemap
         dest=/usr/sbin/makemap
         type=byte

      /usr/local/sbin/mailstats
         dest=/usr/sbin/mailstats
         type=byte

      /usr/local/sbin/praliases
         dest=/usr/sbin/praliases
         type=byte

      $(shared_conf)/mail/generic/helpfile
         dest=/etc/mail/helpfile
         owner=root group=root
         type=byte

      $(shared_conf)/mail/generic/cf/cf/submit.cf
         dest=/etc/mail/submit.cf
         owner=root group=root
         type=byte

      # The anti-spam database:
      $(shared_conf)/mail/$(site)/access.dir
         dest=/etc/mail/access.dir
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/access.pag
         dest=/etc/mail/access.pag
         owner=root group=root
         type=byte

mail_servers::
      $(shared_conf)/mail/$(site)/local-host-names
         dest=/etc/mail/local-host-names
         type=byte
         define=RelaunchSendmail

      $(shared_conf)/mail/$(site)/server-$(os).cf
         dest=/etc/mail/sendmail.cf
         type=byte
```

```
         define=RelaunchSendmail

      # Specific database to a full-fledged mail router:
      $(shared_conf)/mail/$(site)/domaintable.dir
         dest=/etc/mail/domaintable.dir
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/domaintable.pag
         dest=/etc/mail/domaintable.pag
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/genericstable.dir
         dest=/etc/mail/genericstable.dir
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/genericstable.pag
         dest=/etc/mail/genericstable.pag
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/mailertable.dir
         dest=/etc/mail/mailertable.dir
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/mailertable.pag
         dest=/etc/mail/mailertable.pag
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/virtusertable.dir
         dest=/etc/mail/virtusertable.dir
         owner=root group=root
         type=byte

      $(shared_conf)/mail/$(site)/virtusertable.pag
         dest=/etc/mail/virtusertable.pag
         owner=root group=root
         type=byte

!mail_servers::
      $(shared_conf)/mail/$(site)/leaf-$(os).cf
         dest=/etc/mail/sendmail.cf
         type=byte
         define=RelaunchSendmail

files:
   solaris::
      # The statistics file for sendmail:
      /etc/mail/statistics
         mode=644 owner=root group=bin
         action=touch

links:
   solaris::
      # Links for sendmail pseudo-commands:
      /usr/bin/hoststat ->! /usr/lib/sendmail
      /usr/bin/mailq ->! /usr/lib/sendmail
      /usr/bin/newaliases ->! /usr/lib/sendmail
      /usr/bin/purgestat ->! /usr/lib/sendmail


directories:
   # Set sendmail mode and owner properly for security:
   any::
      / owner=root mode=go-w
      /etc owner=root mode=go-w
      /etc/mail owner=root mode=go-w
      /usr owner=root mode=go-w
      /var owner=root mode=go-w
      /var/spool owner=root mode=go-w
      /var/spool/mqueue owner=root mode=go-w
```

```
shellcommands:
   RelaunchSendmail.solaris::
      "/etc/init.d/sendmail stop"
      "/etc/init.d/sendmail start"

processes:
   solaris::
      # Start it anyway. More than 0 instance should run:
      "sendmail"
         matches=>0
         restart "/etc/init.d/sendmail start"
```

Cette configuration multi-site gère à la fois les machines serveurs et clientes.

Enfin, pour amorcer le système, on profite du premier appel de `cfagent` pour installer confortablement son exécution régulière avec `cfengine.cf` :

```
  editfiles:
   solaris::
      {
         # Add accounting jobs to the root crontab:
         /var/spool/cron/crontabs/root
         AutoCreate
         AppendIfNoSuchLine "# cfengine incantation stuff."
         AppendIfNoSuchLine "# Send the result as a mail to the cfengine local administrator:"
         AppendIfNoSuchLine "19 * * * * /usr/local/sbin/cfagent \
            -f /usr/local/share/cfengine/cfengine.conf 2>&1 \
            | /usr/bin/mailx -s 'Cfengine report' \
            `/usr/local/sbin/cfagent -a  \
            -f /usr/local/share/cfengine/cfengine.conf`"
         DefineClasses "RestartCron"
      }
```

3.2.3 Inclusion dans le Solaris JumpStart

Cfengine est à mettre en place à la fin de l'installation automatique du système d'exploitation, JumpStart pour Solaris, fai pour Debian,...

Dans le cas de JumpStart, l'appel est fait dans le script de terminaison de l'installation :

```
# Run in verbose mode:
set -v -x
# At the end of the install, the future / is indeed in /a,
# and / is only temporary during the installation.
UsrLocal=/usr/local
TempUsrLocal=/a$UsrLocal
# Since there is no running automount yet, no NIS, no DNS,...:
/bin/mkdir -p $TempUsrLocal
/usr/sbin/mount -F nfs 192.44.75.87:/export/calice1/local $TempUsrLocal
# Disable the autoshutdown:
/usr/bin/touch /a/noautoshutdown
# Make cfengine believe it is really in / intead of /a:
/usr/sbin/chroot /a $UsrLocal/sbin/cfagent -v -DInstallationTime -f $UsrLocal/share/cfengine/cfengine.conf
# Clean up the mounting point since it will be recreated by autofs anyway:
/usr/sbin/umount $TempUsrLocal
rmdir $TempUsrLocal
```

C'est l'exécution finale de `cfagent` qui va personnaliser l'installation du système et le rendre pleinement fonctionnel en mettant en place une exécution régulière de `cfagent`, l'agent exécutant de Cfengine.

4 Conclusion

Cfengine est un outil d'assez haut niveau d'automatisation des tâches d'administration système qui permet de concevoir des organisations systèmes complexes multi-sites, multi-architectures avec plusieurs systèmes d'exploitation différents tout en gardant un contrôle gérable de la complexité ce qui explique qu'il soit utilisé par de nombreux sites dans le monde.

Actuellement le logiciel est en pleine évolution avec la version 2 qui va vers la maturité, même si le portage sous NT n'a pas récemment évolué. On peut néanmoins regretter quelques lourdeurs de syntaxe et le fait finalement que Cfengine ne soit pas inclut dans un vrai langage de programmation classique. Cela pourrait automatiser encore plus l'administration avec la possibilité de générer au vol des règles Cfengine. C'est un axe de recherche actuellement regardé au LIT avec le projet PCFengine qui consiste à avoir les concepts d'un Cfengine d'ordre supérieur (pour pouvoir manipuler directement les concepts de Cfengine) directement dans le langage PERL.

Il reste encore à faire de notre côté la mutualisation des efforts pour la mise au point de l'infrastructure générique et automatiser la spécialisation de l'infrastructure pour une intégration dans un nouveau contexte et site. On aurait ainsi l'équivalent des *patterns* et *frameworks* du monde du génie logiciel qu'il suffirait d'instancier pour avoir une configuration système complète correspondant à ses besoins.

Des sites ne voulant pas tout partager dans ce pot commun pourraient rajouter des règles d'exclusion dans `rsync` (du style `-exclude conf/**/CEA-DAM/**` si par exemple le CEA/DAM voulait utiliser un tel système...)

Les exemples complets sont disponibles depuis la page WWW de l'auteur de cet article et dans son cours d'administration système. De même, on peut y récupérer la version d'origine étendue de cet article en LaTeX avant sa traduction via TeX4ht en HTML et sa digestion par Word.

Coordonnées de l'auteur

Laboratoire Informatique et Télécommunications
École Nationale Supérieure des Télécommunications de Bretagne
BP832, 29285 BREST CEDEX, FRANCE.
Tél. : (+33|0) 2.29.00.14.15, fax. : (+33|0) 2.29.00.12.82.
`Ronan.Keryell@enst-bretagne.fr`, http://www-info.enst-bretagne.fr/~keryell

Références

[Bur01a] Mark BURGESS. « Cfengine : A configuration engine », 2001. http://www.cfengine.org.

[Bur01b] Mark BURGESS. « Cfengine Reference », 2001. http://www.cfengine.org/docs/cfengine-Reference.html.

[Bur01c] Mark BURGESS. « Cfengine Tutorial : Automated System Administration », 2001. http://www.cfengine.org/docs/cfengine-Tutorial.html.

[Bur01d] Mark BURGESS. « Recent Developments in Cfengine ». http://www.iu.hio.no/~mark/papers/UnixNLBurgess.ps, 2001.

[Ker01] Ronan KERYELL. « Utilisation du logiciel d'administration automatique Cfengine — version étendue ». Rapport Technique ENSTBr/INFO/RR/2001-011, Laboratoire Informatique & Télécommunications, École Nationale Supérieure des Télécommunications de Bretagne, France, octobre 2001. http://www.cri.ensmp.fr/~keryell/publications/conf/2001/JRES2001/cfengine.