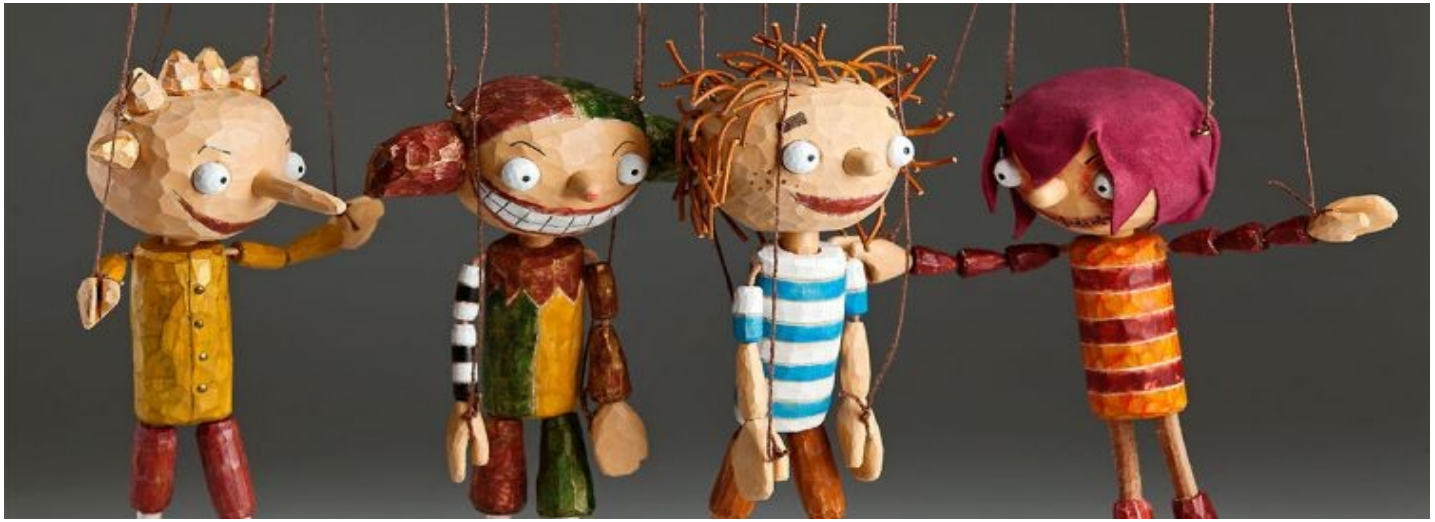


Puppet : C'est vous qui tirez les ficelles !

https://puppet.com

Journées Informatique IN2P3/IRFU, Le Lioran, 26-29 Septembre 2016
Jean-Michel BARBET, ASR , Laboratoire SUBATECH Nantes

Pourquoi cette présentation

- Début en « auto-formation » et l'aide de F.Schaer : Puppet et Foreman (été 2013)
  - Organisation « maison » mais efficace (à base de hostgroups)
- Formation DR17 (Jonathan Schaeffer)
  - Les bases (octobre 2014)
- Formation France-Grilles CCIN2P3 Novembre 2015
  - assez haut-niveau (la première marche est haute!)
- Proposition : réduire à un minimum vital simple et extensible pour installer sa première machine et évoluer par la suite : le « starting-kit »

Plan

- Introduction : Puppet en très bref
- Notion de modules, réutilisation du code, séparation code/données
- Eco-système autour de Puppet pour maîtriser les changements et installer automatiquement les machines
- Composition de modules
- Ce qui n'a pas été abordé mais mérite un coup d'oeil
- Vers un « starting-kit » ?
- Conclusion

Puppet (en très bref!) : principe

- Un langage déclaratif décrivant un état à obtenir. Un fichier contenant du code Puppet est appelé « manifest ».
- Compilation d'un ensemble de manifests et création de code exécutable appelé un « catalog » spécifique pour la machine cible.
- Un agent exécute le catalog sur la machine cible
- Ambition d'être indépendant de la plateforme cible (RedHat, Debian, MacOX, Windows (si!)
- Idempotence : ne pas refaire ce qui a déjà été fait, toujours converger vers l'état final décrit par les manifests.

Puppet (en très bref!) : le langage

- Ressources : permet de spécifier un élement de configuration d'un système (un fichier, un package à installer, un service, une entrée cron, etc.)
- Classes : permet de regrouper des ressources en un ensemble cohérent pour gérer un service
- Variables
- Facts : variables collectées sur le système cible
- Instructions conditionnelles
- Composition de classes : instanciation (declare) ou include

Un exemple de manifest

```
class hostgroups::eosv12::nrpeconfig {

package { 'nrpe':
       ensure   => installed
   }
package { 'nagios-common':
       ensure   => installed
   }
package { 'nagios-plugins':
       ensure   => installed
   }
file { '/etc/nagios/nrpe.cfg':
      mode => "644",
      source => "puppet:///modules/hostgroups/eosv10-nagios-nrpe.conf",
      notify => Service['nrpe'],
      require => Package['nrpe']
   }
service { 'nrpe':
        ensure => running,
        enable => true,
        require => Package['nrpe'],
        hasstatus => true ,
        hasrestart => true
}
```

Modules et réutilisation du code

- La notion de module vise à permettre la réutilisabilité du code
- Un module doit être « auto-contenu » et faire le moins d'hypothèses possibles sur son utilisation
- Toutes les données spécifiques d'un site ou d'un contexte d'utilisation doivent être sorties et pouvoir être passées en paramètre ou demandées à une base de données comme Hiera [2].
- Il existe de nombreux sites proposant des modules Puppet prêts à l'emploi.
- Il existe des bonnes pratiques pour écrire des modules réellement réutilisables et fiables.

Mise en oeuvre

- Utilisation possible en « standalone » directement sur la machine cible

  puppet apply mon_manifest.pp

- Généralement en mode client/serveur :
  - Serveur : puppetmaster
  - Clients : puppet agent
  - Autorité de certification : PuppetCA
  - Bootstrap : manifest site.pp
  - Notion d'environnement : production ou test ou ...
  - Liste de classes (manifests, modules) à appliquer selon la cible

Maîtrise des changements

- Dans chaque environnement, le code sur le serveur Puppetmaster (classes, modules) va évoluer au cours du temps
- Il est souhaitable d'avoir recours à un logiciel de suivi des versions de code. GIT est préconisé
- Le Puppetmaster récupère (git clone) les dernières modifications depuis un dépôt GIT (local, remote, git-hub, git-lab)
- Généralement, on fait correspondre les différents « environnements » Puppet à des branches GIT
- R10K [3] permet d'automatiser, sur le puppetmaster, la création/suppression d'environnements correspondants aux branches GIT.

Déploiement automatisé

- Puppet intervient après l'installation initiale d'une machine intégrée au réseau
- C'est, par exemple, le complément d'une installation de type Kickstart pour Linux RedHat
- Il existe plusieurs solutions pour automatiser les installations Kickstart avec DHCP/PXE : Foreman, Cobbler, …
- Une nouvelle machine présente un certificat qu'il faut valider au niveau de Puppet-CA (service associé au puppetmaster mais qui peut être déployé sur une machine distincte). Cette validation peut être rendu automatique (mécanisme « autosign »)

Repositories

- Pour maîtriser parfaitement l'état d'une machine et produire des installations identiques dans le temps, on ne peut pas s'appuyer sur les repositories logiciels officiels car ils évoluent dans le temps.
- Il est préférable d'entretenir des copies locales de ces repositories et de les figer (système de liens hardware : tags)
- Les définitions de repositories (/etc/yum.repo.d) devront être modifiées très tôt dans l'application des manifests puppet (utilisation de contraintes d'ordonnancement)

Surveiller Puppet

- Lorsqu'un ensemble de machines sont gérées avec puppet, il faut pouvoir détecter les échecs d'application de catalogs et les machines qui ne sont plus synchronisées.
- Plusieurs solutions existent :
  - Analyse du répertoire « reports » sur le puppetmaster
  - Foreman (remontée de rapports d'exécution)
  - Nagios
  - [...]

Composition de modules

- Composition de modules par empilement de classes
- Pour chaque machine, il existe une liste de classes à appliquer
- Des modules paramétrables et des données séparées spécifiques de chaque machine devraient éviter la duplication de code
- Modèle « rôle/profil » :
  - Chaque machine a un rôle unique
  - Un rôle est une composition de profils utilisant des modules prêts à l'emploi (appelés « modules technologiques »)
- Hiera[2] est une base de données hiérarchique pour les données

Vers un « starting-kit »

- Installer un Puppetmaster
- Créer un repository GIT local
- Y déposer le contenu de départ (rôle/profil ultra simple)
- Installer et configurer R10K
  - Récupère les modules externes
  - Crée dynamiquement les environnements à partir des branches GIT
- Installer la première machine
  - Boot sur un CD et spécification d'un fichier kickstart
  - Par la suite : DHCP, PXE et fichier kickstart

Quattor/Puppet

- Beaucoup de points communs
  - Grandeurs et misères du « développement collaboratif »
  - Réutiliser du code écrit ailleurs
  - Nécessite de maîtriser les repositories (depuis quattor/yum)
- Quelques différences à noter
  - Puppet ne gère pas l'installation (quattor le fait avec aii-shelfe)
  - Puppet : pas à développer des modules d'exécution ncm-*
  - La notion de modules et les « guidelines » permet-elle de mieux encadrer le développement collaboratif ?

Puppet : Quelques écueils

- Gestion de la configuration réseau
- Gestion des repositories et des updates
- Modules pour la grille ? Où en est Glite-worker-node ?
- Le recours à des contraintes d'ordonnancement
- Où décide-t'on de l'environnement ?
  - /etc/puppet/puppet.conf sur la machine cible ?
  - ENC (External Node Classifier comme avec Foreman)
  - Autre mécanisme ?

Conclusion

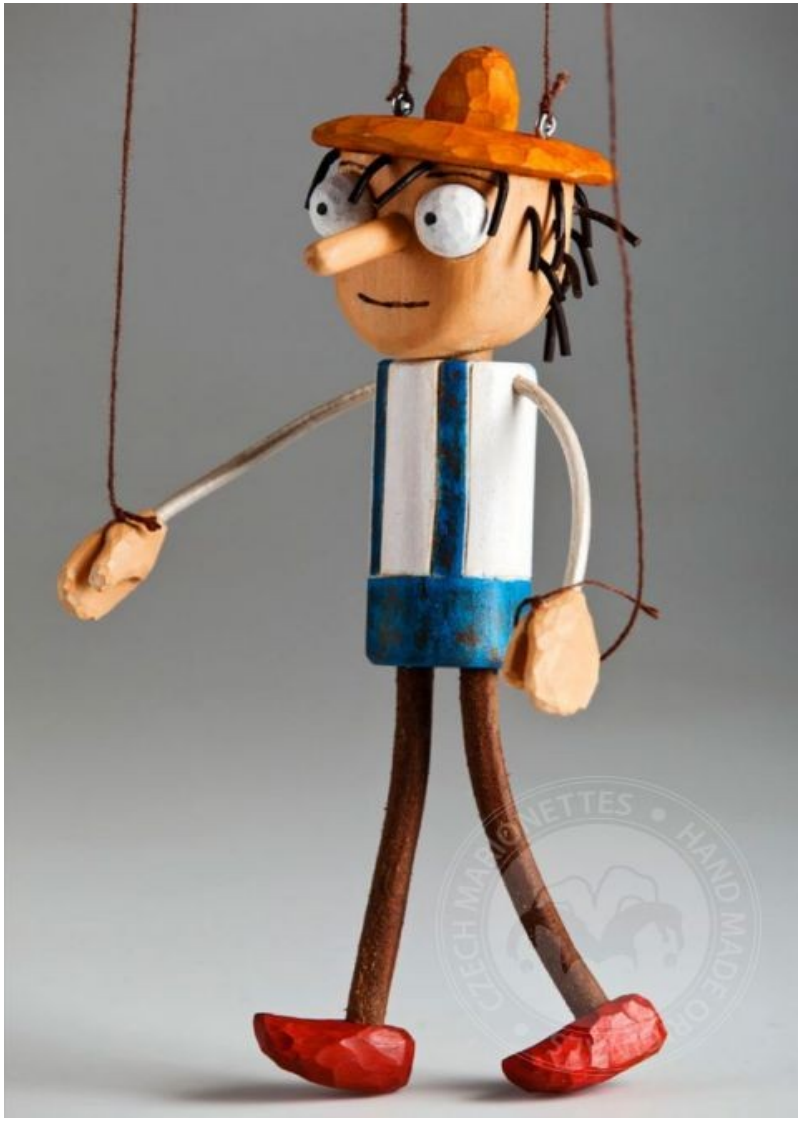
- Sentiment de mieux maîtriser la configuration des machines qu'avec Quattor
- Le recours à des modules externes reste un défi avec l'évolution de ces modules (mises à jour, prise en compte de nouveaux OS, systemd,…)
- Puppet/Quattor : quel avenir avec le Cloud et Docker ?

Remerciements

- Fredéric Shaer pour son entousiasme et son assistance
- Rémi Ferrand, Fabien Wernli et Jérôme Pansanel pour la formation au CC et les conseils précieux
- Jonathan Shaeffer pour la formation puppet de la DR17
- Les membres du forum de discussion configsys@renater
- Mes collègues de Subatech pour avoir fait l'effort de s 'adapter à ces nouveaux outils : puppet, git, foreman

Questions

- (et les réponses en essayant de ne pas mentir, sinon ...)



Liens et références

- [1] Puppet : https://puppet.com
- [2] Hiera : https://docs.puppet.com/hiera/
- [3] R10K : https://github.com/puppetlabs/r10k