Puppet - l'infrastructure automatisée

Marc Fournier

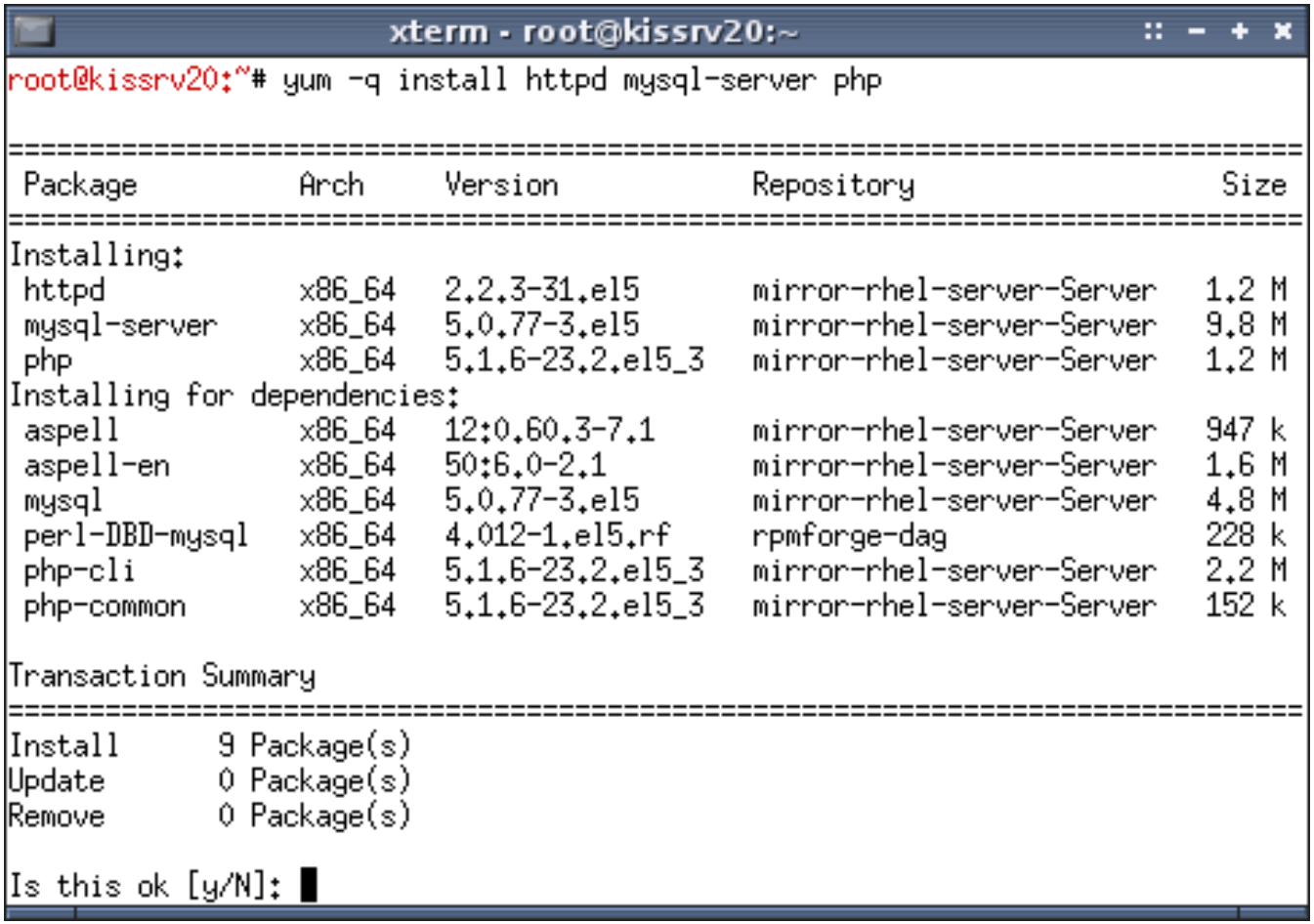
2 mars 2010

Cycle de vie d'une machine

. . . le quotidien du sysadmin . . .

1. Installation de l'OS

2. Configuration du système



3. Maintenance

mise à jour, dépannage, reconfiguration, dépannage, patch, debuggage, déconfiguration, modification, mise à jour, etc . . .

Configuration manuelle

Login and "Just Do It" !

⇒ inadapté à la croissance
⇒ répétitif, erreurs d'inattention
⇒ pas d'historique / documentation des interventions
⇒ pas toujours reproductible de façon fiable
⇒ grande rigueur requise (surtout en équipe) !

Gestion centralisée

- scripts shell / perl / Makefile
- réplication de fichiers avec rsync
- centralisation du stockage / authentification
- `for i in $SERVER_LIST ; do ssh root@${i} ... ; done`

⇒ gestion des erreurs ?
⇒ exceptions ?
⇒ logs ?
⇒ undo ?
⇒ différentes versions ? différents OS ? différentes architectures ?
⇒ solutions "fait maison", évoluant mal, intransposables ailleurs.

Procédures d'installation

- Documentation des recettes d'installation et configuration (howtos, wikis, etc)
- procédures pas-à-pas, checklists

⇒ rapidement périmés
⇒ constitués à 90% de commandes exécutables
⇒ copy-n-paste man
⇒ fix faits dans l'urgence pas remontés dans la doc.

Automatisation, rationalisation

- Installation automatisée (FAI, kickstart, jumpstart, etc)
- duplicata de modèles, partimage, Norton Ghost, etc.

$\Rightarrow$ pas de maintenance / maintenance manuelle
$\Rightarrow$ maintenance du template
$\Rightarrow$ et lorsque les besoins évoluent avec le temps ?

Croissance

- OS / serveurs devenus des consommables
- augmentation du nombre de serveurs (virtualisation).

$\Rightarrow$ augmentation du nombre de sysadmins ?
$\Rightarrow$ augmentation du nombre d'heures sup ?

Inventaire, documentation

- nécessité d'inventorier les machines
- documentation du fonctionnement de l'infra.

⇒ laborieux, ennuyeux, jamais prioritaire
⇒ rapidement périmé / désynchronisé.

Conclusion

Besoin de :

- gérer la configuration du système
- automatiser la phase de maintenance
- formaliser les "recettes"
- ne pas résoudre 2x le même problème
- favoriser la réutilisation du savoir-faire
- . . . contenir l'entropie et le chaos dans le parc de serveurs !

James White Infrastructure Manifesto (extraits)

- There is one system, not a collection of systems
- The actual state of the system must self-correct to the desired state
- The only authoritative source for the actual state of the system is the system
- The entire system must be deployable using source media and text files
- Do not use any product with configurations that are not machine parsable and machine writeable
- Do not improve manual processes if you can automate them instead

⇒ `http://loki.websages.com/ws/`


Puppet

Puppet

Puppet, c'est :

- un framework client-serveur
- un langage de "programmation"
- une boîte à outils pour le sysadmin
- un logiciel libre écrit en Ruby
- supporté par la majorité des UNIX modernes.

Puppet

Puppet, ce n'est pas :

- un outil d'inventaire
- un service de distribution de logiciels
- un remplaçant pour FAI / kickstart / jumpstart
- (obligatoirement) pour gérer l'ensemble du système
- une fin en soi
- une bonne raison d'être moins rigoureux !

Caractéristiques du langage

- déclaratif, idempotent (Makefile)
- **what** not **how**
- permet de **décrire** des collections de ressources
- syntaxe facile et abordable.

Ressources

Décrit les propriétés de différents types d'objets :

- user
- file
- package
- cron
- clé ssh
- etc.

⇒ crée si absent, corrige si différent
⇒ pas seulement des fichiers ou des paquets !
⇒ abstraction des spécificités propres à l'OS
⇒ déclarés dans des "manifests".

Exemple de ressource

```
file { "/etc/resolv.conf":
  ensure  => present,
  owner   => "root",
  group   => "root",
  mode    => 0644,
  content => "
    search camptocamp.com
    nameserver 10.27.21.1
    nameserver 10.26.21.1
  ",
}
```

Ressources : inter-dépendances

```
package { ["apache", "tomcat"]: ensure => installed }

service { "apache":
  ensure  => running,
  require => Package["apache"],
}

service { "tomcat":
  ensure  => running,
  require => Package["tomcat"],
  before  => Service["apache"],
}
```

Ressources : notification

```
file { "/etc/apache/httpd.conf":
  ensure  => present,
  content => template("example.com/httpd.conf.erb"),
  notify  => Service["apache"],
}

service { "apache":
  ensure  => running,
  restart => "apachectl configtest && apachectl graceful",
}
```

Attributs, conditions

```
if ( $use_nagios == "true" ) {

  $warn = $processorcount * 3
  $crit = $processorcount * 6

  monitoring::check { "Load Average":
    command => "check_load",
    options => "-w ${warn} -c ${crit}",
  }

}
```

Classes

Regroupement de ressources, héritage :

```
class apache {
  package { "apache": ensure => present }

  service { "apache":
    ensure  => running,
    require => Package["apache"],
  }

  file { "/var/www":
    ensure => directory,
  }
}
```

Définitions

```
define apache::vhost ($ensure=present, $source) {

  file { "/etc/apache2/sites-enabled/${name}.conf":
    ensure  => $ensure,
    content => template("apache/vhost.conf.erb"),
    notify  => Service["apache"],
  }

  file { "/var/www/$name":
    ensure  => directory,
    source  => $source,
    recurse => true,
  }

}
```

Nodes

Déclaration des classes/définitions à appliquer sur une machine :

```
node "websrv1.example.com" {
  include apache
  include mysql
  include php

  apache::vhost { "www.example.com":
     ensure => present,
     source => "puppet:///web/example.com/htdocs/",
  }
}
```

Caractéristiques du client (puppetd)

- interpréteur de code puppet
- agent ou cron (par défaut 2x/h)
- envoie les "facts" de l'OS au serveur
- reçoit la description de sa config depuis le serveur (catalogue).

Facter

- données caractérisant le système
- accessible sous forme de variable dans le langage puppet
- facilement extensible, Ruby.

Facter : exemple

```
root@example:~# facter | grep operating
operatingsystem => RedHat
operatingsystemrelease => 5
```

Utilisation facts

```
case $operatingsystem {
  RedHat: { include apache-redhat }
  Debian: { include apache-debian }
}
```

Caractéristiques du serveur (puppetmasterd)

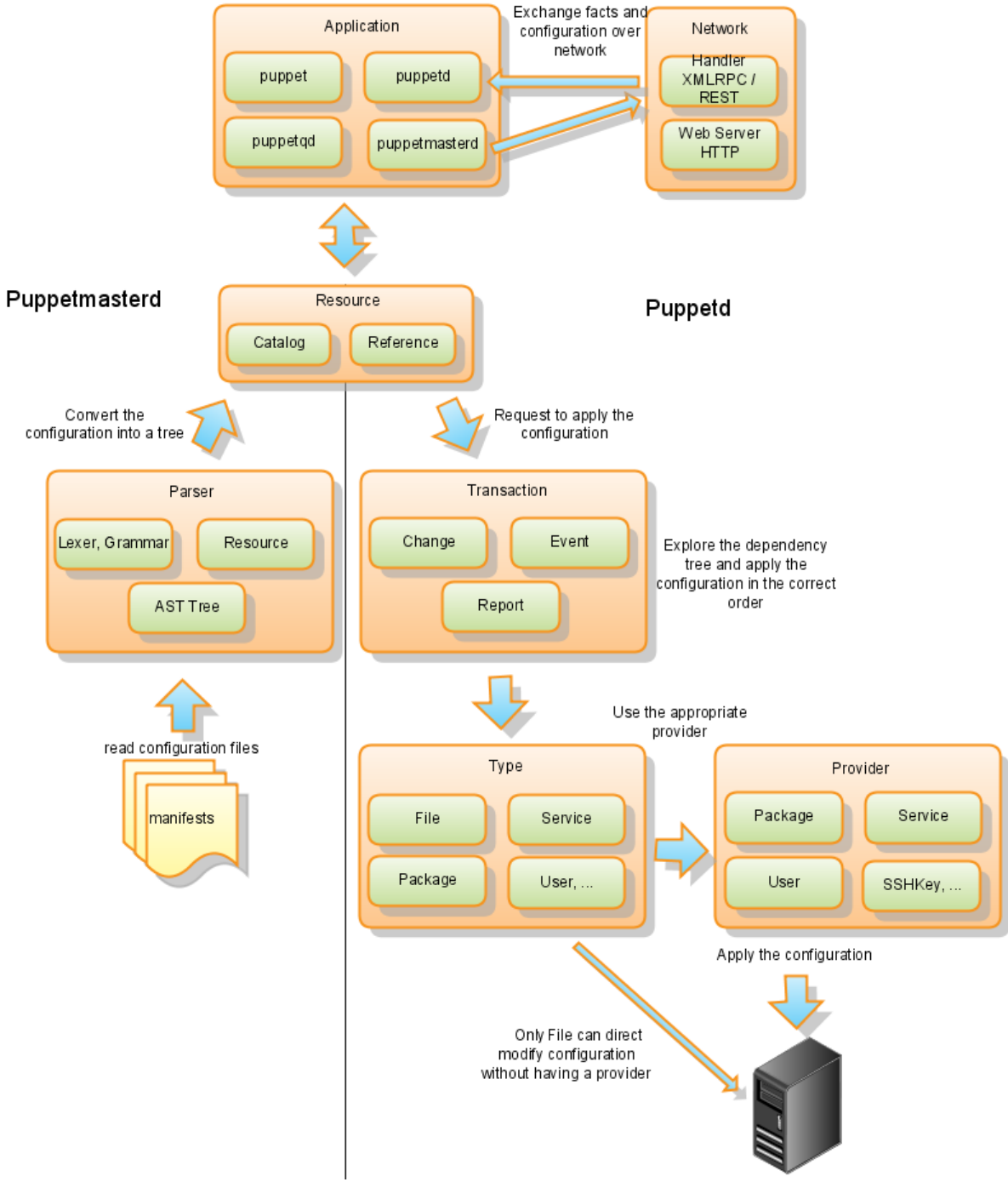
- authentifie les clients par certificat SSL
- compile les ressources de chaque client
- définition des nodes (fichiers texte, ldap ou script)
- log de toutes les opérations
- fileserver
- moteur de template.

Mais surtout :

- centralise les facts de chaque client dans une DB
- centralise et réexporte les ressources de chaque client.

"Companion tools"

- foreman / dashboard – interface web
- puppetdoc – génération de documentation
- puppetrun – forcer un run à partir du serveur
- puppetca – gestion des certificats SSL
- puppet – CLI parser
- ralsh – ressources $\Rightarrow$ puppet
- pi – référence du langage.


Application
puppet
puppetd
puppetqd
puppetmasterd
Exchange facts and configuration over network
Network
Handler XMLRPC / REST
Web Server HTTP
Puppetmasterd
Puppetd
Resource
Catalog
Reference
Convert the configuration into a tree
Request to apply the configuration
Parser
Lexer, Grammar
Resource
AST Tree
Transaction
Change
Event
Report
Explore the dependency tree and apply the configuration in the correct order
read configuration files
manifests
Use the appropriate provider
Type
File
Service
Package
User, ...
Provider
Package
Service
User
SSHKey, ...
Apply the configuration
Only File can direct modify configuration without having a provider

Centralisation de la gestion

Tout converge vers le serveur puppet :

- inventaire / annuaire
- outils de gestion (monitoring, métriques, audit, backups, etc)
- savoir-faire
- documentation / logs d'intervention

Infrastructure programmable

Paradigme propre au développement :

- séparation code générique/spécifique, librairies de fonctions
- code open-source
- discipline imposée par un VCS : diff, branch, revert, blame, etc
- workflow : dev $\Rightarrow$ test $\Rightarrow$ prod
- code review
- inline documentation
- méthodologies dev (XP, Agile, Scrum)
- tests unitaires, continuous integration.

Bénéfices

- configurations homogènes
- pas d'intervention manuelle en production
- recettes reproductibles de façon fiable et précise
- totalement automatisable (cloud computing, diskless servers)
- mutualisation et partage du savoir-faire sysadmin.

$\Rightarrow$ Manifesto de James White devenu concret !

Dans le monde

- Uni Stanford, Harvard
- Google, Ebay
- Fedora, RedHat
- Alfresco, Oracle
- digg.com, guardian.co.uk, sans.org
- hyves.nl (3000 serveurs)
- ReductiveLabs.

En Suisse

- PSI
- EPF*
- Postfinance
- local.ch, swisstopo.ch
- Puzzle ITC
- Nimag Networks
- Camptocamp

Camptocamp : retour d'expérience

- env. 45 postes de travail Ubuntu en Suisse et en France
- 2-3 minutes pour l'installation d'un nouveau poste.

- chaque projet ⇒ 3 serveurs (dev/pré-prod/prod)
- déploiement en production sans intervention manuelle
- plus de 450 serveurs et postes de travail sous Debian/Ubuntu/Red Hat
- pics de charge délégués à Amazon EC2.

Autres outils

- chef
- cfengine
- bcfg2
- etc.

⇒ `http://en.wikipedia.org/wiki/Comparison_of_open_source_configuration_management_software`

Futur

- version "1.0"
- nouveaux types de ressources
- common modules repository
- windows, autre périphériques
- punc (puppet + func)
- jruby
- DSL alternatifs.

Ressources

- `http://docs.reductivelabs.com/`
- `puppet-users@googlegroups.com`
- `#puppet` sur irc.freenode.net
- "Pulling Strings with Puppet" chez Apress
- PuppetCamp 27-28 Mai, Ghent, BE
- `http://reductivelabs.com/training/`
- `http://spug.ch/`

Résumé

⇒ besoin d'industrialisation et de gestion automatisée de l'infra
⇒ qualité, cohérence, intégrité : pas réalisable manuellement
⇒ puppet décrit, distribue et réalise les configurations
⇒ déclare l'état du système dans son ensemble, sous forme de code
⇒ mutualisation et partage du savoir-faire.

Questions ?