

Gestión de sistemas con Puppet

Imobach González Sosa
@imobachgs

Banot.net
<http://www.banot.net/>

Febrero de 2011



Contenidos

- 1 El problema y sus soluciones
- 2 Visión general de Puppet
- 3 Conceptos de modelado
- 4 Configuración
- 5 Extendiendo Puppet
- 6 Dashboard



Contenidos

1 El problema y sus soluciones

2 Visión general de Puppet

3 Conceptos de modelado

4 Configuración

5 Extendiendo Puppet

6 Dashboard



La repetitiva vida del sysadmin

- Tareas repetitivas
 - ▶ Propensas a errores
 - ▶ Tediosas
 - ▶ Aburridas
- La infraestructura cambia y crece
 - ▶ Más servicios que gestionar
 - ▶ Más «hardware» que administrar
- El trabajo en equipo no siempre es fácil
 - ▶ Cada uno hace las cosas a su modo
 - ▶ Necesidad de documentar estandarizar los procedimientos



Soluciones: «self-made» vs SCM

Necesidades

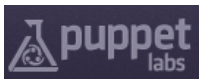
Automatizar y simplificar las tareas

- Soluciones «self-made»
 - ▶ Tienen a reinventar la rueda
 - ▶ Administradores no siempre son desarrolladores
 - ▶ No suele ser escalable
- Sistemas de gestión de configuración
 - ▶ El sistema se encarga de los detalles
 - ▶ Es fácil compartir el código

Contenidos

- 1 El problema y sus soluciones
- 2 Visión general de Puppet**
- 3 Conceptos de modelado
- 4 Configuración
- 5 Extendiendo Puppet
- 6 Dashboard

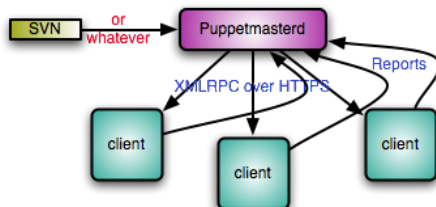
Puppet



- Es uno de los muchos sistemas de gestión de configuración existentes
 - ▶ Cfengine, Chef, Bcfg2, etc.
- Desarrollado por Puppet Labs
- Acaba de publicar Puppet Enterprise
- ¿Quién lo usa? Twitter, Sun/Oracle, Rackspace, Digg, match.com, SugarCRM...



Arquitectura



- Existe un servidor central que gestiona las configuraciones
- Los clientes se conectan periódicamente
- Los clientes pueden enviar informes al servidor
- La comunicación se realiza sobre REST (antes era XML-RPC)



Show me the code

```
file { '/etc/group' :  
  owner => root,  
  group => root,  
  mode => 644  
}
```

```
package { 'apache2' :  
  ensure => installed,  
  configfiles => keep,  
}
```

Algunas características

- Idempotencia
 - ▶ Aplicar la configuración varias veces no cambia el resultado
- Multiplataforma
 - ▶ Linux (múltiples distribuciones)
 - ▶ FreeBSD y OpenBSD
 - ▶ Mac OS X, Solaris, AIX y HP-UX
 - ▶ Microsoft Windows (limitado)
- Alto nivel de abstracción
 - ▶ Se piensa en términos de recursos, no de ficheros de configuración (como Cfengine)



Tipo frente implementación

- Tipo de recurso
 - ▶ Tipo e implementación están desacoplados
 - ▶ `file` o `package` son tipos
 - ▶ Más de 30 tipos
- Proveedor
 - ▶ Implementación de los tipos de recursos
 - ▶ Para paquetes: `apt`, `yum`, `gem`...
 - ▶ Puppet suele detectar el proveedor correcto



Factor

¿Qué es?

Biblioteca que recupera propiedades del sistema operativo («facts»)

- Arquitectura, dominio, sistema operativo...
- Puppet convierte los «facts» en variables
- *You can prove anything with facts!*

```
package { 'apache' :  
  name => $operatingsystem ? {  
    Debian => "apache2",  
    Redhat  => "httpd",  
    default => "apache",  
  },
```



Contenidos

- 1 El problema y sus soluciones
- 2 Visión general de Puppet
- 3 Conceptos de modelado**
- 4 Configuración
- 5 Extendiendo Puppet
- 6 Dashboard

Recursos

- Unidad básica de modelado
- Describe algún aspecto del sistema (servicios, ficheros, paquetes, usuarios...)

Tipo `package`, `file`, `user`...

Título Nombre para hacer referencia al recurso

Atributos Dependen del tipo de recurso (por ejemplo, `owner` para `file`)



Recursos

Ejemplo

```
host { 'puppetserver':  
  name => 'puppet.kvm.banot.net',  
  ip => '192.168.122.22',  
  host_aliases => [ 'puppet', 'dashboard' ],  
  ensure => 'present',  
}
```



Metaparámetros

- Atributos aplicables a todos los tipos de recursos
- Metaparámetros interesantes: `before/require` y `notify/subscribe`

```
file { "/etc/sshd_config":  
  source => template("sshd_config.erb"),  
  notify => Service[sshd]  
}
```



Colecciones de recursos: clases

¿Qué son?

Una clase define una serie de recursos que pueden aplicarse a un nodo

- Se ejecutan una sola vez por nodo (*singleton*)
- Soportan herencia simple
- Son parametrizables
- Pueden anidarse
- Es típico agrupar `service`, `package` y `file`
- Ejemplo: el servicio `apache`

Ejemplo de clase

```
class unix {  
  file { '/etc/password':  
    owner => 'root',  
    group => 'root',  
    mode => 644;  
  }  
  
  file { '/etc/group':  
    owner => 'root',  
    group => 'root',  
    mode => 640;  
  }  
}
```



Classes

Herencia

```
class freebsd inherits unix {  
  File['/etc/passwd'] { group => wheel }  
  File['/etc/shadow'] { group => wheel }  
}
```

Colecciones de recursos: definiciones

- También definen una serie de recursos pero...
- ... se pueden ejecutar varias veces por nodo
- No soportan herencia
- También son parametrizables
- No se pueden anidar
- Soportan metaparámetros
- Ejemplo: un `virtualhost` de Apache

Definiciones

Ejemplo

```
define svn_repo($path) {
  exec { "/usr/bin/svnadmin create $path/$title":
    unless => "/bin/test -d $path",
  }
}

svn_repo { puppet_repo: path => '/var/svn_puppet' }
svn_repo { other_repo:   path => '/var/svn_other' }
```



Nodos

- Representa a las máquinas gestionadas por Puppet
- Los nodos se identifican por nombre
- Importante que el DNS funcione bien
- Se usa el nodo «default» si no coincide ninguno

```
node 'www.banot.net', 'www2.banot.net' {  
    include base  
    include webserver  
}
```



Nodos

Expresiones regulares

- Se pueden usar expresiones regulares para identificar las máquinas

```
node /^smtp\d+.banot.net$/ {  
  include common  
  include smtpserver  
}
```



Nodos

Herencia

- Como las clases, soporta herencia

```
node 'smtp.banot.net' {  
  include base  
  include smtpserver  
  ...  
}
```

```
node 'lists.banot.net' inherits 'smtp.banot.net' {  
  include mailman  
  ...  
}
```



¡Sólo tengo 45 minutos!

- Variables, *arrays*, diccionarios...
- Operadores
- Estructuras de control (*if/else*, selectores...)
- Funciones
- Dependencias entre recursos
- Stages
- Nodos externos
- Recursos virtuales
- ...



Contenidos

- 1 El problema y sus soluciones
- 2 Visión general de Puppet
- 3 Conceptos de modelado
- 4 Configuración**
- 5 Extendiendo Puppet
- 6 Dashboard

Organización de la configuración

```
puppet/  
  auth.conf  
  fileserver.conf  
  puppet.conf  
  manifests/  
    site.pp  
  modules/  
  templates/
```

Ejemplo

```
puppet/  
  auth.conf  
  fileserver.conf  
  puppet.conf  
  manifests/  
    classes/  
      base.pp  
      webserver.pp  
      site.pp  
  modules/  
  files/  
    samhainrc  
  templates/  
    apache2-site.erb
```



Contenidos

- 1 El problema y sus soluciones
- 2 Visión general de Puppet
- 3 Conceptos de modelado
- 4 Configuración
- 5 Extendiendo Puppet**
- 6 Dashboard

Módulos

- Colección de clases, definiciones y recursos
- Facilitan la reutilización y la redistribución
- En la web de Puppet existe un repositorio (*Module Forge*)
- Muchos de esos módulos están alojados en Github

Morfología de un módulo

```
mi_modulo/  
  manifests/  
    init.pp  
  lib/  
    facter/  
    puppet/  
      parser/  
        functions  
      provider/  
      type/  
  files/  
  templates/  
  README
```



Añadiendo «facts»

- Es posible añadir «facts» de forma sencilla

```
Facter.add("samba_sid") do
  setcode do
    %x{/usr/bin/net getlocalsid}.chomp
  end
end
```



Contenidos

- 1 El problema y sus soluciones
- 2 Visión general de Puppet
- 3 Conceptos de modelado
- 4 Configuración
- 5 Extendiendo Puppet
- 6 Dashboard**

Dashboard

- Aplicación web que permite ver el estado de los nodos
- Se basa en los informes que le envían los clientes
- Desarrollada con Ruby on Rails 2.3

Dashboard

puppet dashboard v1.0.4 » Home » Nodes » Groups » Classes » Reports

Nodes

- Currently successful 1
- Currently failing 0
- Ever succeeded 1
- Ever failed 1
- Never reported 0
- Not currently reporting 1

Add node

Class

Add class

Group

Add group

Dashboard

Nodes no longer reporting

1 node has not reported in the last about 1 hour: puppet-client.kvm.banot.net.

Daily run status

Number and status of runs during the last 30 days:

Date	Failed (Red)	Successful (Green)	Total
2011-01-30	0	12	12
2011-01-31	0	8	8
2011-02-01	3	18	21
2011-02-02	0	5	5
2011-02-03	0	1	1



Referencias



[Puppet Labs](#)

<http://www.puppetlabs.com>



[Puppet](#)

<http://projects.puppetlabs.com/projects/puppet/>



[Chef](#)

<http://www.opscode.com/chef/>



[Cfengine](#)

<http://www.cfengine.org/>



[Bcfg2](#)

<http://trac.mcs.anl.gov/projects/bcfg2>

