Orquestación de la Configuración con Puppet

Por:

José Luis Jaime Gonzalez

Junio de 2013

Índice de contenido

1 - Introducción

El proyecto tiene como objetivo aprender a utilizar puppet como herramienta para administrar la configuración de varios servidores de forma sencilla y automatizada.

Veremos como instalar y configurar puppet tanto en el lado del servidor como en el cliente.

Crearemos varios módulos para automatizar la instalación y configuración de apache, php y mysql y una aplicación php.

También veremos como instalar de forma automática un balanceador de carga (http), en este caso haproxy y mediante un sistemas de scripts automatizaremos el proceso de añadir un nuevo nodo al balanceador de forma automática cuando se llegue a un numero de peticiones concurrentes.

Por ultimo vemos como instalar y configurar un dashboard o herramienta web para monitorizar puppet de forma gráfica pudiendo comprobar el estado de los nodos, los cambios realizados, fallos etc..

Todo el código de este proyecto esta alojado en mi github:

https://github.com/joseluisjaime/puppet-project

Para ver el código mas claro se recomienda verlo en el repositorio.

2 - ¿Que es Puppet?

Puppet es una herramienta desarrollada por Puppetlabs para administrar la configuración de sistemas Unix y windows de forma **declarativa**, esto quiere decir, que que no le decimos a la maquina lo que tiene que ejecutar, si no, el estado final en el que queremos que se encuentre.

Para entenderlo de una forma clara:

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

Con esta simple declaración, puppet es capaz de instalar apache2, en ningún caso nosotros hemos indicado si tiene que usar apt en caso de debian, yum en caso de centos, solo indicamos que este seguro de que el paquete apache2 este instalado.

En resumen, puppet nos permite realizar la configuración de forma abstracta, especificando el estado en el que queremos que se encuentre la maquina y no las ordenes que tiene que ejecutar, esto nos permite que con la declaración anterior, instalar el paquete independientemente del gestor de paquetes que tenga el sistema.

3 - Componentes y Características

Puppet esta desarrollado en ruby, actualmente existen dos versiones una libre y otra enterprise :

Puppet open source, bajo licencia Apache 2.0 para las versiones 2.7 en adelante y GNU GPL para las anteriores.

Puppet enterprise, es la solución de pago que ofrece PuppetLabs, aunque puedes probar el producto con un limite de 10 nodos.

En el siguiente imagen podemos ver las diferencias:

FEATURES	Puppet Open Source	Puppet Enterprise
Graphical User Interface		✔
Provisioning – Amazon EC2	✔	✔
Provisioning – VMware VMs		✔
Configuration management – Discovery		✔
Configuration management – User accounts		✔
Configuration management – Operating systems & applications	✔	✔
1000+ pre-built configurations on Puppet Forge	✔	✔
Orchestration – Task automation		✔
Role-Based Access Control – Now with external authentication support		✔
Unified cross-platform installer of all components		✔
Support – Option for 24 x 7 x 365		✔
Support – Defined SLA		✔
Certified by Puppet Labs engineers		✔
Pre-packaged dependencies in one directory		✔
Smooth upgrade and maintenance path		✔

En este proyecto se usa Puppet open source.

Los requisitos básicos para Puppet 3 son:

Ruby: 1.8.7 o 1.9.3

Facter: Que es una herramienta que nos permite recolectar información de los nodos tales como ip, hostname, nombre del sistema, versión etc.

Hiera: Es una herramienta que nos permite organizar y externalizar las variables de nuestros módulos, haciéndolos muchos mas eficientes y permitiéndonos tener un mayor control de la configuración.

Mas adelante veremos algunos ejemplos de hiera y facter para entender su funcionamiento,

Puppet soporta una gran cantidad de sistemas lo que lo hace aun mas potente:

Linux:

- Debian, versión 5 o superior.
- Ubuntu 8.04 o superior.
- Red Hat Enterprise, versión 5 o superior.
- Fedora, version 15 o superior.
- SUSE Enterprise Server, versión 11 o supeior.
- Gentoo Linux
- Mandriva Corporate Server 4
- ArchLinux

Unix:

- Mac OS X, versión 10.5 o supeior.
- Oracle Solaris, versión 10 o superior.
- AIX, versión 5.3 o superior.
- FreeBSD 4.7 o superior.
- OpenBSD 4.1 o superior.
- HP-UX

Windows:

- Windows Server 2003 y 2008
- Windows 7

¿Que nos permite hacer puppet?

- Gestión de paquetes.
- Gestión de configuraciones/ficheros
- Gestión de scripts, cron, mount ...
- Gestión de servicios
- Gestión de usuarios, grupos

4 - Como Funciona.

Puppet tiene una estructura de Cliente-Servidor, por lo tanto debemos instalar software tanto en la parte de servidor, como en los clientes.

En el servidor estará configurado con Puppetmaster, que sera el que administre los demás nodos y un cliente puppet en los nodos.

La comunicación entre el cliente y el servidor es cifrada mediante SSL y uso de certificados, puppetmaster incluye una pequeña entidad autoridad certificadora que maneja certificados X509.

El proceso de comunicación básica seria el siguiente:

Los clientes, periódicamente solicitan al servidor el estado en el que tienen que estar.

El servidor coge todos los recursos* que están asignados al nodo que ha hecho la petición y compila un catalogo* que envía al nodo.

El cliente lee el catalogo y aplica las configuraciones necesarias para replicar el estado con el que se ha configurado previamente.

Recurso: Un recursos puede ser un fichero, la instalación de un paquete, el estado de un servicio, etc..

Catalogo: Se trata del estado exacto en el que tiene que estar la maquina. (paquete instalado, fichero con la configuración, servicio corriendo etc..)

5 - Instalación y Configuración básica de Puppet.

5 .1 - Instalación en el Servidor

En este proyecto vamos a trabajar con la versión 3.2 de puppet sobre debian wheezy.

En el lado del servidor el paquete principal que vamos a instalar se llama Puppetmaster.

Actualmente en los repositorios de debian esta puppet pero en su versión 2.7, en este caso como vamos a usar una mayor, vamos a instalar los repositorios de PuppetLabs, para ello:

Instalamos los repositorios:

```
wget http://apt.puppetlabs.com/puppetlabs-release-wheezy.deb
dpkg -i puppetlabs-release-wheezy.deb
```

Actualizamos los repositorios:

```
apt-get update
```

Ahora instalamos el paquete puppetmaster, también se nos instalara por dependencia, los demás paquetes necesarios como ruby, facter, hiera etc..

```
apt-get install puppetmaster
```

Ya tendríamos Puppetmaster funcionando correctamente.

5.2 - Instalación en el Cliente.

Para los clientes no es necesario tener la ultima versión por lo que podemos usar directamente los repositorios para instalar puppet.

En el caso del cliente el paquete que tenemos que instalar es puppet.

```
apt-get install puppet
```

Esto nos resolverá todas las dependencias necesarias.

Una vez instalado tenemos que realizar unas pequeñas configuraciones.

Cambiamos el siguiente parámetro a yes en “/etc/default/puppet” para que se inicie el servicio cuando se inicie el sistema:

```
START=yes
```

Agregamos las siguientes lineas a “/etc/puppet/puppet.conf” y reiniciamos el servicio.

```
[agent]
server=master.example.com
```

En mi caso tengo definido en el “/etc/hosts” de los clientes una entrada con el master:

```
192.168.122.2 master.example.com master
```

Para que el cliente pueda comunicarse con el servidor, debemos firmar su certificado, para ver la lista de nodos que tenemos pendiente de firmar el certificado, en el puppetmaster ejecutamos:

```
root@master:~# puppet cert --list

"prueba1.example.com"(MD5)A4:89:F2:3B:2A:89:FC:92:DC:47:88:32:45
:55:79:F8
```

Para firmar su certificado ejecutamos:

```
root@master:~# puppet cert --sign prueba1.example.com
Notice: Signed certificate request for prueba1.example.com
Notice:Removing file Puppet::SSL::CertificateRequest
prueba1.example.com at
'/var/lib/puppet/ssl/ca/requests/prueba1.example.com.pem'
```

Si volvemos a comprobar la lista, esta vez con “--all” porque sin esta opción solo salen los que no están firmados:

```
root@master:~# puppet cert --list –all
+"prueba1.example.com"(SHA256)4F:E3:CD:B4:80:19:70:A0:4C:CA:28:2
E:13:A5:76:7F:37:65:05:A7:D3:64:2A:36:0E:63:E4:61:05:FC:50:F7
```

El signo + al principio significa que esta firmado.

Y con esto ya tendríamos preparado tanto la parte del servidor como la parte del cliente.

6 - Creación de los Módulos

Un modulo es un conjunto de recursos agrupados, por ejemplo, el recurso de la instalación del paquete de Apache, el fichero de configuración, el servicio de apache, pueden formar el modulo Apache que podemos asignar a los nodos.

Es una forma de tener ordenados los distintos recursos para que sea de fácil comprensión.

Antes de empezar a ver como podemos crear un modulo vamos a ver la estructura de directorios.

6.1 - Estructura de los módulos.

Todos los ficheros de configuración donde vamos a definir los recursos tienen la terminación .pp

```
root@master:~# ls /etc/puppet
auth.conf  fileserver.conf  hiera.yaml    manifests  modules
puppet.conf  templates
```

El directorio donde vamos a definir nuestros nodos es manifests. Dentro de manifests debemos crear un fichero llamado **“nodes.pp”** donde vamos a definir nuestros nodos.

La sintaxis para definir un nodo es la siguiente:

```
root@master:~# cat /etc/puppet/manifests/nodes.pp
node 'prueba1.example.com' {


}
```

También tenemos que crear un fichero **“site.pp”** donde vamos a importar el fichero nodes.pp

```
root@master:~# cat /etc/puppet/manifests/site.pp
import "nodes"
```

Con esto ya tendríamos definido el nodo prueba1.example.com.

Ahora vamos a ver la estructura de un modulo. Los módulos los creamos dentro de la carpeta modules:

```
root@master:~# ls /etc/puppet/modules/
apache
root@master:~# ls /etc/puppet/modules/apache/
files  manifests  templates  tests
```

files: En este directorio van los ficheros estáticos tales como un index.html que se pasaran a los nodos y no necesitan cambios.

templates: Aquí van los ficheros dinámicos, donde podemos usar variables que se sustituirán a la hora de crearse en el cliente. Los templates terminan en la extensión .erb Mas adelante veremos un ejemplo.

tests: En este directorio podemos instanciar el modulo para testearlo sin aplicarlo a ninguna maquina, mas adelante veremos un ejemplo.

manifests: Este directorio es el mas importante y es donde crearemos los recursos, por defecto el fichero creado aquí tiene que llamarse init.pp.

Mas adelante quedara todo mas claro conforme se vayan viendo los ejemplos.

6.2 - Recursos

En este apartado vamos a ver los distintos recursos que podemos usar a la hora de crear un modulo, hay bastantes mas de los mencionados aquí, que son los que mas he usado y los principales:

Para instalar un paquete:

```
package { 'apache package':
 # Nombre que veremos en los logs
    ensure => installed,
 # Estado que queremos el paquete
    name => "apache2",
 # Nombre del paquete en si.
  }
```

Mas opciones para el parámetro **ensure :**

- latest: Si queremos que se instale siempre el ultimo
- purged: Si queremos eliminar el paquete.

Definir un fichero o carpeta:

```
  file {'index.html apache':
 # nombre del recurso.
    ensure => file,
 # Puppet se asegura de que sea un fichero.
    owner => www-data,
 # Propietario
    group => www-data,
 # Grupo
    mode => 0640,
 # Permisos
    source => “puppet:///modules/apache/index.html” # Buscara un
fichero estativo en el directorio files con nombre index.html
    path => "/var/www/index.html", # donde queremos colocarlo.
```

```
    }
```

Otros parámetros:

```
ensure => directory  #se asegura que es un directorio.
```

Si queremos coger un fichero dinámico, cambiamos **source** por:

```
content => template('apache/index.html.erb') # buscara en el
directorio templates, un fichero llamado index.html.erb
```

Definir un servicio:

```
service {'apache service': # nombre del recurso
    ensure => running, #El estado en el que queremos el servicio
    name => "apache2", # Nombre del servicio
    }
```

Otras opciones:

```
ensure => stopped # Si queremos que este parado.
```

Gestionar usuarios:

```
user {'www-data': # Nombre de usuario.
  ensure => present, # Estado en el que queremos el usuario
}
```

El siguiente recurso nos permite ejecutar comandos del sistema:

```
exec { "Extract tar.":
   path => "/bin:/usr/bin", #path donde esta el comando.
   unless => "find /tmp/prueba", # explicado abajo.
  command => "tar -xzf /tmp/prueba.tar.gz", #comando a ejecutar
 }
```

unless: Esto nos sirve como una regla para cuando queremos que se ejecute el recurso. Funciona con la variable $?, si el resultado del comando de unless, la variable $? vale 1, se ejecuta el command, si da 0, no se ejecuta.

Si vale 1 es porque no ha encontrado la carpeta extraída y puede extraerla y si da 0 es porque la ha encontrado por lo que no es necesario extraerlo.

Hay diversas opciones diferentes que no están mostradas aquí, se pueden consultar en la documentación oficial de puppet.

Ahora que ya tenemos claro los recursos y la estructura de un modulo, vamos a ver como seria crear un modulo que instale apache2.

6.3 - Modulo Apache

Vamos a ver como se crearía un modulo de apache, que instale apache2 y con la configuración por defecto y deje un simple index.html

Los recursos se definen dentro del directorio modules, a partir de ahora modulepath (/etc/puppet/modules por defecto pero se puede cambiar esta ubicación).

Lo primero que tenemos que crear es el init.pp, que es el fichero donde mira puppet los recursos del modulo. Iría dentro del directorio “modulepath/apache/manifests/init.pp” y tendría el siguiente contenido:

```
class apache {


     Package['apache package']  -> File['index.html'] ~>
Service['apache service']


  package { 'apache package':
    ensure => installed,
    name => "apache2",
  }


  file {'index.html file':
    ensure => file,
    owner => www-data,
    group => www-data,
    mode => 0640,
    source => “puppet:///modules/apache/index.html”,
    path => "/vaw/www/index.html",
    }
```

```
service {'apache service':
    ensure => running,
    name => "apache2",
    }

}
```

Puppet no interpreta los recursos en el orden que esta en el fichero por lo que tenemos que indicarle el orden en el que queremos que lo ejecute, porque no queremos que intente iniciar el servicio si no se ha instalado todavía el paquete. Para ello usamos la segunda linea:

```
Package['apache package']  -> File['index.html'] ~>
Service['apache service']
```

Cuando usamos la ~ significa que queremos que el recurso se reinicie si el recurso que lo precede cambia, o lo que es lo mismo si tuviéramos un fichero de configuración de apache y no un indexhtml, cuando el File cambia, el Service se reinicia. Ahora no tiene sentido ya que reiniciara el servicio cada vez que cambiemos el index.html

en el directorio files, creamos un simple index.html.

Con esto ya tendríamos los recursos para la instalación de apache.

Para comprobar que no nos hemos equivocado en la sintaxis podemos ejecutar la siguiente orden:

```
root@master:/etc/puppet/modules/apache# puppet parser validate
manifests/*
root@master:/etc/puppet/modules/apache#
```

Si no nos da ningún error, es que la sintaxis esta bien. (Cuidado con el tipo de comillas al copiar de este documento)

Ahora para probar que puppet ejecuta bien el modulo, creamos el siguien fichero que nos servira para hacer una simulación: “modulepath/apache/tests/init.pp”

```
root@master:/etc/puppet/modules/apache# cat tests/init.pp
include apache
```

Ejecutamos el siguiente comando:

```
root@master:/etc/puppet/modules/apache# puppet apply --noop
tests/init.pp
Notice: /Stage[main]/Apache::Install/Package[apache]/ensure:
current_value purged, should be present (noop)
Notice: Class[Apache::Install]: Would have triggered 'refresh'
from 1 events
Notice: /Stage[main]/Apache::Config/File[index.html]/content:
current_value {md5}21dde95d9d269cbb2fa6560309dca40c, should be
{md5}07dff4d92d217a5e5c4eacd7dfba3ea5 (noop)
Notice: /Stage[main]/Apache::Config/File[index.html]/owner:
current_value root, should be www-data (noop)
Notice: /Stage[main]/Apache::Config/File[index.html]/group:
current_value root, should be www-data (noop)
Notice: /Stage[main]/Apache::Config/File[index.html]/mode:
current_value 0644, should be 0640 (noop)
Notice: Class[Apache::Config]: Would have triggered 'refresh'
from 4 events
```

```
Notice: Class[Apache::Service]: Would have triggered 'refresh'
from 1 events
Notice: /Stage[main]/Apache::Service/Service[apache]: Would have
triggered 'refresh' from 1 events
Notice: Class[Apache::Service]: Would have triggered 'refresh'
from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 3
events
Notice: Finished catalog run in 0.89 seconds
```

Con la opción –noop indicamos que sea una simulación y no queremos que se apliquen los cambios, sin la opción, nos instalaría el modulo en la propia maquina.

Como vemos, se han ejecutado en el orden que pretendíamos, primero se instala el paquete, después se crea el fichero con sus respectivos permisos y mas tarde se intenta iniciar el servicio.

Ahora si queremos que este module se ejecute en el nodo que creamos prueba1.example.com, vamos al directorio /etc/puppet/manifests donde se definen los nodos, e incluimos la siguiente linea en nodes.pp:

```
node 'prueba1.example.com' {
  include apache #apache equivale al nombre de la clase.
}
```

Los clientes por defecto hacen una petición cada 30 min, si queremos comprobar los cambios ahora, ejecutamos en el cliente:

```
puppet agent –t
```

6.4 - Subclases

Un cambio en la configuración que podemos hacer en el puppetmaster es la ubicación de los directorios manifests (donde se definen los nodos), y modules (donde se definen los módulos). En mi caso como he trabajado en un repositorio en github, he cambiado la ubicación de estos. Para ello añadimos las siguientes lineas en la sección [Main] en "/etc/puppet/puppet.conf"

```
modulepath = /usr/share/puppet/puppet-project/modules
manifestdir = /usr/share/puppet/puppet-project/manifests
```

A partir de ahora todas las configuraciones de los scripts y módulos estarán reverenciadas a esa ruta.

Siendo puppet-project mi repositorio.

Ahora que hemos visto como se crea un modulo de apache vamos a ver como dividimos el modulo para tenerlo todo mas estructurado, ya que aunque ese modulo solo tiene 3 recursos, si tuviéramos que instalar 10 paquetes, 10 ficheros de configuración e iniciar 7 servicios, nos quedaría un fichero bastante largo y difícil de manejar.

Otro punto importante son las variables, que nos permiten tener la configuración mucho mas ordenada y clara, siendo fácil realizar cualquier cambio sin tener que modificar varios ficheros del modulo. Mas adelante con los ejemplos quedara mas claro.

El init.pp ahora lo dividimos en:

init.pp: Fichero principal donde hacemos los include, declaramos las variables, e indicamos el orden.

snstall.pp: Aquí ponemos todos los recursos relacionados con la instalación.

sonfig.pp: Aquí ponemos todos los recursos relacionados con la instalación.

service.pp: Aquí ponemos los recursos relacionados con el manejo de servicios.

El nuevo modulo tendría la siguiente estructura:

```
jose@josepc:~/puppet-project/modules/apache$ ls *
manifests:
config.pp  init.pp  install.pp  service.pp

templates:
index.html.erb

tests:
init.pp
```

init.pp

```
class apache (

  $apache_package = hiera('apache_package'),
  $indexhtml = hiera('indexhtml'),
  $apache_service = hiera('apache_service'),
  $machinename = $hostname,
  $machineip = $ipaddress,
  $osname = $operatingsystem,

) {

  Class['apache::install'] -> Class['apache::config'] ~>
Class['apache::service']

  include apache::install
  include apache::config
  include apache::service

}
```

Ahora para decidir el orden de ejecución usamos Class['nombreclase']

En el siguiente punto se explicaran las variables con detalle.

Install.pp

```
class apache::install {

  package { 'apache':
    ensure => installed,
    name => "${apache::apache_package}",
  }
}
```

config.pp

```
class apache::config {

  file {'index.html':
    ensure => file,
    owner => www-data,
    group => www-data,
    mode => 0640,
    content => template('apache/index.html.erb'),
    path => "${apache::indexhtml}",
    }
}
```

service.pp

```
class apache::service {

  service {'apache':
    ensure => running,
    name => "${apache::apache_service}",
    }

}
```

index.html.erb (fichero dinamico.)

```
<% $machinename =  scope.lookupvar('apache::machinename') -%>
<% $machineip =  scope.lookupvar('apache::machineip') -%>
<% $osname = scope.lookupvar('apache::osname') -%>
<html>
<body>
<h1>Apache funciona correctamente</h1>
<p>Nombre de la maquina: <%= $machinename %></p>
<p>Ip de la maquina: <%= $machineip %></p>
<p>Sistema operativo: <%= $osname %></p>
</body>
</html>
```

Como vemos, hemos separado en subclases, de forma que el código se entienda mucho mejor si tuviéramos que instalar 10 paquetes, iniciar 10 servicios, etc..

6.5 - Variables

Como hemos visto en el modulo de apache del punto anterior, ahora cualquier dato que estuviese sujeto a cambiar, lo hemos sacado a una variable, de forma que solo cambiando las variables sin tener que tocar la configuración del modulo, nos permitiría usar el modulo en otro entorno.

Respecto a las variables entran dos componentes en juego.

Facter

Facter es una herramienta que se instala (por dependencia en nuestro caso) en los clientes que nos permite recopilar toda clase de información y usarla como variables. Si ejecutamos en el cliente el comando:

```
root@prueba1:~# facter
/usr/lib/ruby/1.9.1/rubygems/custom_require.rb:36:in `require': iconv will be deprecated in the future, use String#encode instead.
architecture => amd64
domain => example.com
facterversion => 1.6.10
fqdn => prueba1.example.com
hardwareisa => unknown
hardwaremodel => x86_64
hostname => prueba1
id => root
interfaces => eth0,lo
ipaddress => 192.168.122.10
ipaddress_eth0 => 192.168.122.10
ipaddress_lo => 127.0.0.1
etc...
```

Veremos que nos aparece bastante información, todas estas variables son globales y las podemos usar como en el caso de apache en el init.pp:

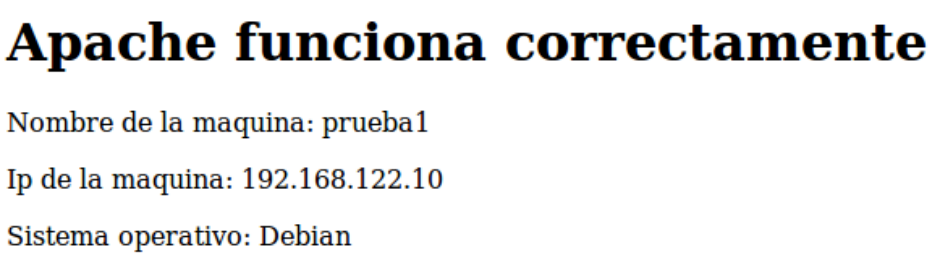
```
  $machinename = $hostname,
  $machineip = $ipaddress,
  $osname = $operatingsystem,
```

Como vemos las variables se llaman tal cual salen en el comando facter.

Estas variables las he usado en un template dinámico, que podemos ver en el apartado anterior.

De forma que al copiarse el fichero al nodo se sustituyen quedando la información:

Si instalamos el modulo en la maquina, y vemos el index.html:



Hiera

Hiera es el otro componente que nos permite mejorar nuestras configuraciones en puppet. Esta herramienta nos permite externalizar las variables en otro fichero, en formato .yaml, de forma que no tengamos que volver a tocar la configuración del modulo.

Otra mejora que nos proporciona es una estructura herarquica a la hora de mirar en los ficheros para elegir variables. Con el ejemplo mas adelante se entenderá mejor:

Hiera viene de forma nativa a partir de puppet 3.

Para poder activar hiera debemos crear el siguiente fichero: “/etc/puppet/hiera.yaml” (Hay que reiniciar puppetmaster una vez creado el fichero)

```
---
:hierarchy:
    - %{::osfamily}
    - common
:backends:
    - yaml
:yaml:
    :datadir: '/usr/share/puppet/puppet-project/hieradata'
```

Hierarchy: Es la jerarquía por la que va a buscar las variables, primero va a buscar un fichero Debian.yaml (osfamily es una variable de facter por lo que en nuestro caso el nodo es un Debian, buscara primero en Debian.yaml, si el nodo fuese centos, buscaria un fichero llamado centos.yaml, si no lo encuentra, buscara en el fichero common.yaml.

Backends: Indicamos que vamos a usar ficheros con formato yaml, pero tambien podemos usar json, mysql etc.

Yaml: Aqui indicamos donde queremos que busque los ficheros .yaml

```
jose@josepc:~/puppet-project$ ls hieradata/
common.yaml  Debian.yaml
```

```
jose@josepc:~/puppet-project$ cat hieradata/common.yaml
---
apache_package: 'apache2'
indexhtml: '/var/www/index.html'
apache_service: 'apache2'
```

Como vemos en el init.pp del modulo de apache para indicar que una variable busque en hiera, lo realizamos con:

```
$apache_package = hiera('apache_package')
```

Con esta forma de hacer los módulos, los hacemos mas flexibles, reutilizables, y compatibles con varios sistemas. Un ejemplo claro seria el paquete snmp.

El paquete snmp en debian se llama snmp y en centos net-snmp, por lo que con la primera forma de hacer el modulo, al indicar el nombre del paquete, en este caso snmp, no lo podría instalar en centos puesto que no se llama así.

Sin embargo si el nombre del paquete es una variable, y en este caso usamos hiera, podríamos crear un fichero llamado Debian.yaml

```
---
snmp_package: 'snmp'
```

y un fichero Centos.yaml

```
---
snmp_package: 'net-snmp'
```

La variable es la misma, pero gracias a facter que nos dice que sistema es, y a hiera que lo tenemos configurado para que mire primero en el osfamily, ya que esta primero en la jerarquía en el caso de Debian mirara en Debian.yaml y centos en el Centos.yaml

De este modo, este modulo de apache nos vale tanto para centos, como para debian, como para cualquier sistema. Y si lo tuviéramos que reutilizar, podríamos cambiar las variables para elegir nombres de paquetes, ficheros, rutas. Etc.

7 - Modulo LAMP

Ahora vamos a ver los módulos que he creado para este proyecto.

El modulo de apache ya lo vimos en el punto anterior, para completar una infraestructura LAMP necesitamos PHP y MySQL.

PHP

```
jose@josepc:~/puppet-project/modules/php$ ls *
files:
info.php  php.ini

manifests:
config.pp  init.pp  install.pp

templates:

tests:
init.pp
```

El modulo de php no tiene servicio por lo que no tiene una subclase service.pp

A demás tiene dos ficheros estáticos, que son un info.php para comprobar el funcionamiento y un php.ini como fichero de configuración para apache.

Init.pp

```
class php (

  $php_package = hiera('php_package'),
  $php_mysql_package = hiera('php_mysql_package'),
  $phpinfo = hiera('phpinfo'),
  $phpini = hiera('phpini'),

)  {

  Class['php::install'] -> Class['php::config']

  include php::install
  include php::config
}
```

install.pp

```
class php::install {

  Package['php'] -> Package['php_mysql']

  package { 'php':
    ensure => installed,
    name => "${php::php_package}",
  }
  package { 'php_mysql':
    ensure => installed,
    name => "${php::php_mysql_package}",
  }
}
```

config.pp

```
class php::config {

  file {'phpinfo':
    ensure => file,
    owner => www-data,
    group => www-data,
    mode => 0644,
    source => 'puppet:///modules/php/info.php',
    path => "${php::phpinfo}",
    }

    file {'phpini':
      ensure => file,
      owner => root,
      group => root,
      mode => 0644,
      source => 'puppet:///modules/php/php.ini',
      path => "${php::phpini}",
    }

}
```

El fichero estático info.php

```
<?php phpinfo(); ?>
```

El fichero php.ini es uno estándar al instalar el sistema, se puede ver en el repositorio de github.

MYSQL

```
jose@josepc:~/puppet-project/modules/mysql$ ls *
files:

manifests:
config.pp  init.pp  install.pp  other.pp  service.pp

templates:

tests:
init.pp
```

init.pp

```
class mysql (

  $mysql_package = hiera('mysql_package'),
  $mysql_service = hiera('mysql_service'),
  $mysql_root_password = hiera('mysql_root_password'),

)  {

  Class['mysql::install'] ~> Class['mysql::service'] ->
Class['mysql::other']

  include mysql::install
  include mysql::service
  include mysql::other

}
```

install.pp

```
class mysql::install {


  package { 'mysql':
    ensure => installed,
    name => "${mysql::mysql_package}",
  }
}
```

En este caso, config.pp esta vacío ya que no necesito configurar nada.

Service.pp

```
class mysql::service {


  service {'mysql':
    ensure => running,
    name => "${mysql::mysql_service}",
  }
}
```

other.pp

```
class mysql::other {


  exec { "Set Mysql Root Password":
    subscribe => [ Package["mysql-server"]],
    refreshonly => true,
    unless => "mysqladmin -uroot -p${mysql::mysql_root_password}
status",
    path => "/bin:/usr/bin",
    command => "mysqladmin -uroot password $
{mysql::mysql_root_password}",
  }
}
```

En este caso al instalar mysql-server en debian, te pide la contraseña de root, al instalarlo puppet la contraseña se queda vacía por lo que he creado este recurso que comprueba si la contraseña esta vacía, y si lo esta le asigna una.

También podemos crear roles o grupos para agrupar varios modulos, por ejemplo, crearemos un rol, role_lamp donde juntaremos apache, php y mysql.

Para ello en el directorio manifests donde definimos los nodos, creamos un nuevo fichero llamado roles.pp

```
jose@josepc:~/puppet-project/manifests$ cat roles.pp
class aptupdate {
  exec { 'aptupdate':
    path => "/bin:/usr/bin",
    command => "apt-get -y update",
    timeout => 600,
  }
}

class role_lamp {

  stage { 'update': before => Stage['main'] }

  class {'aptupdate':
    stage => update,
  }

  class {'apache':}
  class {'php':}
  class {'mysql':}
}
```

Stage: Con la stage podemos indicar el orden en que queremos que se apliquen los módulos. Por defecto la stage principal es main, por lo que podemos definir que se ejecute una antes, y asignar una clase a esa stage. Como por ejemplo la clase para actualizar apt-get update.

Una vez hemos creado el fichero roles.pp lo importamos en el fichero site.pp

```
jose@josepc:~/puppet-project/manifests$ cat site.pp
import "roles"
import "nodes"
```

Ahora en el fichero nodes.pp podemos incluir role_lamp

```
node 'prueba1.example.com' {
  include role_lamp
}
```

Probamos la infraestructura LAMP en el nodo:

```
root@prueba1:~# puppet agent -t
info: Caching catalog for prueba1.example.com
info: Applying configuration version '1371760797'
notice: /Stage[update]/Aptupdate/Exec[aptupdate]/returns:
executed successfully
notice: /Stage[main]/Php::Install/Package[php]/ensure: ensure
changed 'purged' to 'present'
notice: /Stage[main]/Php::Install/Package[php_mysql]/ensure:
ensure changed 'purged' to 'present'
notice: /Stage[main]/Php::Config/File[phpinfo]/ensure: defined
content as '{md5}767d787126c09cf4a3bc19218b47de4c'
notice: /Stage[main]/Mysql::Install/Package[mysql]/ensure:
ensure changed 'purged' to 'present'
info: /Stage[main]/Mysql::Install/Package[mysql]: Scheduling
refresh of Exec[Set Mysql Root Password]
```

```
info: Class[Mysql::Install]: Scheduling refresh of
Class[Mysql::Service]

info: Class[Mysql::Service]: Scheduling refresh of
Service[mysql]

notice: /Stage[main]/Mysql::Service/Service[mysql]: Triggered
'refresh' from 1 events

notice: /Stage[main]/Php::Config/File[phpini]/content:

--- /etc/php5/apache2/php.ini   2013-06-05 10:14:27.000000000
+0200

+++ /tmp/puppet-file20130620-4955-1ae9ium-0     2013-06-20
22:46:29.693943001 +0200

@@ -873,7 +873,7 @@

 [Date]

 ; Defines the default timezone used by the date functions

 ; http://php.net/date.timezone

-;date.timezone =

+date.timezone = "Europe/Madrid"


 ; http://php.net/date.default-latitude

 ;date.default_latitude = 31.7667


info: FileBucket adding {md5}29f7729de178b69c113a697bbc0035ce

info: /Stage[main]/Php::Config/File[phpini]: Filebucketed
/etc/php5/apache2/php.ini to puppet with sum
29f7729de178b69c113a697bbc0035ce

notice: /Stage[main]/Php::Config/File[phpini]/content: content
changed '{md5}29f7729de178b69c113a697bbc0035ce' to
'{md5}ad41dd02198bdc6971d3b376ef114ed6'

notice: /Stage[main]/Mysql::Other/Exec[Set Mysql Root Password]:
Triggered 'refresh' from 1 events

notice: Finished catalog run in 392.67 seconds
```

Como vemos, el modulo de apache no se ha aplicado ya que ya estaba instalado.

7.1 - Modulo Aplicación TinytinyRSS

Para completar la infraestructura LAMP, he hecho un modulo que instala una aplicación php, en este caso Tinytinyrss que es un lector de noticias RSS.

```
jose@josepc:~/puppet-project/modules/tinyrss$ ls *
files:
Tiny-Tiny-RSS-1.8.tar.gz

manifests:
config.pp  init.pp  install.pp

templates:
config.php.erb

tests:
init.pp
```

init.pp

```
class tinyrss (

  $tinyrss_directory = hiera('tinyrss_directory'),
  $tinyrss_tarball = hiera('tinyrss_tarball'),
  $tinyrss_user = hiera('tinyrss_user'),
  $tinyrss_password = hiera('tinyrss_password'),
  $mysql_root_password = hiera('mysql_root_password'),
  $tinyrss_database = hiera('tinyrss_database'),
  $machineip = $ipaddress,

)  {
```

```
  Class['tinyrss::install'] -> Class['tinyrss::config']

  include tinyrss::install
  include tinyrss::config

}
```

install.pp

```
class tinyrss::install {
# Recurso para crear la carpeta en /var/www
  file {'tinyrss_directory':
    ensure => directory,
    owner => www-data,
    group => www-data,
    mode => 0640,
    path => "${tinyrss::tinyrss_directory}",
  }
# Recurso para copiar el instalador al nodo
  file {'tinyrss_tarball':
    ensure => present,
    owner => www-data,
    group => www-data,
    mode => 0775,
    source => "puppet:///modules/tinyrss/$
{tinyrss::tinyrss_tarball}",
    path => "/tmp/${tinyrss::tinyrss_tarball}",
  }
# Recurso para extraer el tar en la carpeta
  exec { "Extract tinyrss":
    subscribe => [ File['tinyrss_directory']],
    path => "/bin:/usr/bin",
    unless => "find ${tinyrss::tinyrss_directory}/themes",
    command => "tar -xzf /tmp/${tinyrss::tinyrss_tarball}
--strip=1 -C ${tinyrss::tinyrss_directory}",
```

```
  }
# Recurso para crear la base de datos
  exec { "Create tinyrss database":
    path => "/bin:/usr/bin",
    unless => "mysql -u${tinyrss::tinyrss_user} -p$
{tinyrss::tinyrss_password} ${tinyrss::tinyrss_database}",
    command => "mysql -uroot -p${tinyrss::mysql_root_password}
-e \"create database ${tinyrss::tinyrss_database}; grant all on
${tinyrss::tinyrss_database}.* to $
{tinyrss::tinyrss_user}@localhost identified by '$
{tinyrss::tinyrss_password}';\"",
  }
}
```

config.pp

```
class tinyrss::config {
#Copio el fichero de configuración de tinyrss
  file {'tinyrss config file':
    ensure => file,
    owner => www-data,
    group => www-data,
    mode => 0755,
    content => template('tinyrss/config.php.erb'),
    path => "${tinyrss::tinyrss_directory}/config.php"
  }
# Ejecuto el script que rellena la base de datos
  exec { "Populate tinyrss database":
    path => "/bin:/usr/bin",
    unless => "mysql -u${tinyrss::tinyrss_user} -p$
{tinyrss::tinyrss_password} ${tinyrss::tinyrss_database} -e \"
select * from ttrss_users;\" ",
    command => "mysql -u${tinyrss::tinyrss_user} -p$
{tinyrss::tinyrss_password} ${tinyrss::tinyrss_database} < $
{tinyrss::tinyrss_directory}/schema/ttrss_schema_mysql.sql",
  }
```

```
  # Cambio los permisos a www-data:www-data
  exec {'set owner and group tinyrss':
    subscribe => [ Exec["Populate tinyrss database"] ],
    refreshonly => true,
    path => "/bin:/usr/bin",
    command => "chown -R www-data:www-data $
{tinyrss::tinyrss_directory}"
  }

}
```

Aquí podemos ver parte del fichero de configuración dinámico: config.php.erb

```
<?php
     // ****************************************
     // *** Database configuration (important!) ***
     // ****************************************

     define('DB_TYPE', 'mysql');
     define('DB_HOST', 'localhost');
     define('DB_USER', '<%= $tinyrss_user %>');
     define('DB_NAME', '<%= $tinyrss_database %>');
     define('DB_PASS', '<%= $tinyrss_password %>');
     define('DB_PORT', '3306');

     define('MYSQL_CHARSET', 'UTF8');
     // ***********************************
     // *** Basic settings (important!) ***
     // ***********************************

     define('SELF_URL_PATH', 'http://<%= $machineip
%>/tinytinyrss/');
```

También he creado un grupo en roles.pp que incluye role_lamp y la aplicación.

roles.pp

```
class app_tinyrss {
  stage { 'first': before => Stage['main'] }

  class {'role_lamp':
    stage => first,
  }
  class {'tinyrss':
    stage => main,
  }
}
```

Cambiamos el include a app_tinyrss

```
node 'prueba1.example.com' {
  include app_tinyrss
}
```

Al tener ya en el nodo la infraestructura LAMP, ese estado ya lo tiene la maquina, lo que le falta es la aplicación, que es lo que instalara:

```
root@prueba1:~# puppet agent -t
info: Caching catalog for prueba1.example.com
info: Applying configuration version '1371762326'
notice: /Stage[update]/Aptupdate/Exec[aptupdate]/returns:
executed successfully
notice: /Stage[main]/Tinyrss::Install/Exec[Create tinyrss
database]/returns: executed successfully
notice:
/Stage[main]/Tinyrss::Install/File[tinyrss_tarball]/ensure:
defined content as '{md5}d6d9957acb2442dc7ef8083796ead048'
```

```
notice:
/Stage[main]/Tinyrss::Install/File[tinyrss_directory]/ensure:
created

info: /Stage[main]/Tinyrss::Install/File[tinyrss_directory]:
Scheduling refresh of Exec[Extract tinyrss]

notice: /Stage[main]/Tinyrss::Install/Exec[Extract
tinyrss]/returns: executed successfully

notice: /Stage[main]/Tinyrss::Install/Exec[Extract tinyrss]:
Triggered 'refresh' from 1 events

notice: /Stage[main]/Tinyrss::Config/Exec[Populate tinyrss
database]/returns: executed successfully

info: /Stage[main]/Tinyrss::Config/Exec[Populate tinyrss
database]: Scheduling refresh of Exec[set owner and group
tinyrss]

notice: /Stage[main]/Tinyrss::Config/File[tinyrss config
file]/ensure: defined content as
'{md5}d1abfa7c58bdf8dcdfa4cf96694e809c'

notice: /Stage[main]/Tinyrss::Config/Exec[set owner and group
tinyrss]: Triggered 'refresh' from 1 events

notice: Finished catalog run in 6.56 seconds
```

Aquí esta el fichero common.yaml donde estan las variables de los módulos:

```
---


apache_package: 'apache2'

indexhtml: '/var/www/index.html'

apache_service: 'apache2'


php_package: 'php5'

php_mysql_package: 'php5-mysql'
```

```
phpinfo: '/var/www/info.php'
phpini: '/etc/php5/apache2/php.ini'

mysql_root_password: 'puppet'
mysql_package: 'mysql-server'
mysql_service: 'mysql'

tinyrss_directory: '/var/www/tinytinyrss'
tinyrss_tarball: 'Tiny-Tiny-RSS-1.8.tar.gz'
tinyrss_user: 'tinyrss_user'
tinyrss_password: 'tinytinyrss'
tinyrss_database: 'db_tinyrss'
```

8 - Modulo Haproxy

También voy a incluir en el proyecto un modulo que va a instalar un balanceador de carga haproxy.

En debian wheezy no esta en los repositorios por lo que el modulo, copiara el tar.gz y lo compilara.

El modulo esta preparado con una configuración simple que me permita balancear la carga entre dos nodos.

```
jose@josepc:~/puppet-project/modules/haproxy$ ls *
files:
haproxy-1.4.24.tar.gz  haproxy-default  haproxy-initd

manifests:
config.pp  init.pp  install.pp  service.pp

templates:
haproxy.cfg.erb
```

```
tests:
init.pp
```

init.pp

```
class haproxy (

  $haproxy_tarball = hiera('haproxy_tarball'),
  $haproxy_version = hiera('haproxy_version'),
  $haproxy_service = hiera('haproxy_service'),
  $haproxy_socat_package = hiera('haproxy_socat_package'),
  $machinename = $hostname,
#Variables para los distintos backends
#Esta configurado para soportar hasta 3.
  $config_server1 = hiera('config_server1'),
  $config_name1 = hiera('config_name1'),
  $config_ip1 = hiera('config_ip1'),

  $config_server2 = hiera('config_server2'),
  $config_name2 = hiera('config_name2'),
  $config_ip2 = hiera('config_ip2'),

  $config_server3 = hiera('config_server3'),
  $config_name3 = hiera('config_name3'),
  $config_ip3 = hiera('config_ip3'),


)  {
  Class['haproxy::install'] -> Class['haproxy::config'] ~>
Class['haproxy::service']

  include haproxy::install
  include haproxy::config
  include haproxy::service
}
```

install.pp

```
class haproxy::install {


  Package['socat'] -> File['haproxy_tarball'] -> Exec["Extract
haproxy"] -> Exec["install haproxy make"] -> Exec["install
haproxy make install"]
# El paquete socat nos servira para monitorizar haproxy
  package {'socat':
    ensure => installed,
    name => "${haproxy::haproxy_socat_package}",
  }
#Copio el instalador.
  file {'haproxy_tarball':
    owner => root,
    group => root,
    mode => 0775,
    source => "puppet:///modules/haproxy/$
{haproxy::haproxy_tarball}",
    path => "/opt/${haproxy::haproxy_tarball}",
  }
#Lo descomprimo
  exec {"Extract haproxy":
    path => "/bin:/usr/bin",
    subscribe => [ File['haproxy_tarball']],
    unless => "find /opt/${haproxy::haproxy_version}/doc",
    command => "tar -xzf /opt/${haproxy::haproxy_tarball} -C
/opt",
  }
#Lo compilo
  exec {"install haproxy make":
    path => "/bin:/usr/bin",
    subscribe => [ Exec['Extract haproxy']],
    unless => "find /usr/local/sbin/haproxy",
    command => "make TARGET=linux2628 -C /opt/$
{haproxy::haproxy_version}",
  }
```

```
  exec {"install haproxy make install":
    path => "/bin:/usr/bin",
    subscribe => [ Exec['install haproxy make']],
    unless => "find /usr/local/sbin/haproxy",
    command => "make install -C /opt/$
{haproxy::haproxy_version}",
  }
}
```

config.pp

```
class haproxy::config {
#Establezco un orden de ejecución
  File['haproxy init.d'] ->  File['haproxy default'] ->
File['haproxy config directory'] ->  File['haproxy config'] ->
User['haproxy']
# Copio el script del servicio en init.d
  file {'haproxy init.d':
    ensure => file,
    owner => root,
    group => root,
    mode => 0755,
    path => "/etc/init.d/${haproxy::haproxy_service}",
    source => 'puppet:///modules/haproxy/haproxy-initd'
  }
# Copio el fichero default
    file {'haproxy default':
    ensure => file,
    owner => root,
    group => root,
    mode => 0644,
    path => '/etc/default/haproxy',
    source => 'puppet:///modules/haproxy/haproxy-default'
  }
```

```
#Creo el directorio haproxy en /etc
file {'haproxy config directory':
    ensure => directory,
    owner => root,
    group => root,
    mode => 0755,
    path => "/etc/haproxy",
  }

#Copio el fichero de configuración dinámico.
  file {'haproxy config':
    ensure => file,
    owner => root,
    group => root,
    mode => 0644,
    path => '/etc/haproxy/haproxy.cfg',
    content => template('haproxy/haproxy.cfg.erb'),
  }

# Creo el usuario haproxy
  user {'haproxy':
    ensure => present,
  }
}
```

service.pp

```
class haproxy::service {
  service {'haproxy service':
    ensure => running,
    enable => true,
    name => "${haproxy::haproxy_service}"
  }
}
```

Y este seria el fichero de configuración dinámico.

Haproxy.cfg.erb

```
<% $machinename =  scope.lookupvar('haproxy::machinename') -%>
<% $server1 =
[scope.lookupvar('haproxy::config_server1'),scope.lookupvar('hap
roxy::config_name1'),scope.lookupvar('haproxy::config_ip1')]  -
%>
<% $server2 =
[scope.lookupvar('haproxy::config_server2'),scope.lookupvar('hap
roxy::config_name2'),scope.lookupvar('haproxy::config_ip2')]  -
%>
<% $server3 =
[scope.lookupvar('haproxy::config_server3'),scope.lookupvar('hap
roxy::config_name3'),scope.lookupvar('haproxy::config_ip3')]  -
%>
global
log 127.0.0.1 local0
log 127.0.0.1 local1 notice
maxconn 4096
user haproxy
group haproxy
daemon
stats socket /tmp/haproxy level admin
defaults
log global
mode http
option httplog
option dontlognull
retries 3
option redispatch
maxconn 50000
contimeout 5000
clitimeout 50000
srvtimeout 50000
```

```
listen <%= $machinename %> 0.0.0.0:80
mode http
balance roundrobin
<%= $server1[0] %> <%= $server1[1] %> <%= $server1[2] %>
<%= $server2[0] %> <%= $server2[1] %> <%= $server2[2] %>
<%= $server3[0] %> <%= $server3[1] %> <%= $server3[2] %>


listen stats 0.0.0.0:9000       #Listen on all IP's on port 9000
    mode http
    balance
    timeout client 5000
    timeout connect 4000
    timeout server 30000

    stats uri /haproxy_stats

    stats realm HAProxy\ Statistics
    stats auth admin:password
    stats admin if TRUE
```

Para este caso he creado un haproxy.yaml, y he añadido una opción a la jerarquía:

```
:hierarchy:
    - %{::osfamily}
    - common
    - haproxy
```

De forma que voy a almacenar las variables en ese fichero.

Haproxy.yaml

```
---
  haproxy_tarball: 'haproxy-1.4.24.tar.gz'
  haproxy_version: 'haproxy-1.4.24'
  haproxy_service: 'haproxy'
  haproxy_socat_package: 'socat'

  config_server1: 'server'
  config_name1: 'backend2'
  config_ip1: '192.168.122.6'

  config_server2: 'server'
  config_name2: 'client2'
  config_ip2: '192.168.122.4'

  config_server3: ''
  config_name3: ''
  config_ip3: ''
```

Como podemos ver, en los config server configuramos los servidores que van a ser balanceados, pero el tercero lo dejo vacío para el siguiente apartado del proyecto.

8.1 - Automatización de un nodo con Haproxy

El objetivo de este apartado es el siguiente:

Ya tengo un modulo que instala haproxy en un nodo y puede balancear la carga HTTP

Mediante unos scripts voy a monitorizar el numero de peticiones que le llegan al balanceador y cuando supere un numero x de peticiones concurrentes, automáticamente se creara un nodo, se añadirá a puppet y se le aplicaran los módulos de apache y php y se añadirá al balanceador de forma automática. De forma que no habrá dos servidores web respondiendo si no tres. Todo este proceso de forma automática.

Funcionamiento:

monitor.sh

```
#!/bin/bash


IP_BAL=$1

NUM=1

ADD_BALANCEADOR='/home/jose/puppet-project/scripts/vmtool.sh
balanaux 14 1'


while [ $NUM -eq 1 ]; do


# Con el paquete socat puedo hacer consultas al socket de
#haproxy en concreto el numero de peticiones concurrentes que
#hay en ese instante. While infinito que solo se para si se
#alcanzan las conexiones.


    CURRCONNS=$(ssh root@192.168.122.$IP_BAL "echo "show info" |
socat unix-connect:/tmp/haproxy stdio | grep CurrConns | awk
'{print $2}'")


#Me quedo solo con el numero de conexiones

    CONEX=$(echo $CURRCONNS | awk '{print $2}')

    echo $CONEX

    sleep 3
```

```
# Esta configurado para que cuando supere las 300 conexiones
# concurrentes ejecute el script para añadir el balanceador.
# 300 conexiones no son muchas, es un numero simbolico para
# realizar la prueba.

  if [ $CONEX -ge 300 ]; then

     echo "Numero de conexiones altas, añadiendo balanceador"
#ejecuto el script para añadir el balanceador
     $ADD_BALANCEADOR
     echo "Balanceador añadido"
#fuerzo a que se actualize el nuevo balanceador para que se le
#instale apache y php
     ssh root@192.168.122.$IP_BAL "puppet agent -t"
     exit
    fi
```

vmtool.sh

Este script me sirve tanto para crear nodos y añadirlos automáticamente a puppet como para añadir el balanceador, depende del tercer parámetro.

```
#!/bin/bash

NAME=$1 #nombre del nodo
IP=$2 #ip, solo el ultimo numero 192.168.122.X
BALANCEADOR=$3
NODE='/home/jose/puppet-project/manifests/nodes.pp'
HIERA_HAPROXY='/home/jose/puppet-project/hieradata/haproxy.yaml'
UPDATE_SCRIPT='/home/jose/puppet-project/scripts/update.sh'
```

```
#Borro los certificados en el puppetmaster por si el nombre del
#nodo ya hubiera estado antes añadido.
ssh root@192.168.122.2 "puppet cert clean $NAME.example.com"


#Si el tercer parámetro es 1, añado la configuración del nodo
#a nodes.pp y como anteriormente vimos, en el fichero
#haproxy.yaml, actualizo las variables del servidor3


if [ $BALANCEADOR = 1 ]; then


    echo "node '$NAME.example.com' {
  include apache
  include php
}" >> $NODE


    sed -i "s/config_server3: ''/config_server3: 'server'/g"
$HIERA_HAPROXY
    sed -i "s/config_name3: ''/config_name3: '$NAME'/g"
$HIERA_HAPROXY
    sed -i "s/config_ip3: ''/config_ip3: '192.168.122.$IP'/g"
$HIERA_HAPROXY
#Este script lo he creado para hacer un git push/pull en mi
#maquina anfitriona y en el puppetmaster, ya que trabajo desde
# mi maquina anfitriona.
    $UPDATE_SCRIPT


fi


# Empiezo el proceso de clonado de una imagen base que solo
tiene instalado el paquete puppet. En este proceso cambio nombre
de la maquina, ip etc..
virt-clone --connect qemu:///system --original base --name $NAME
--file /media/linux/imagen/$NAME.img
virsh start $NAME
sleep 16
ssh root@192.168.122.122 "echo $NAME > /etc/hostname"
```

```
ssh root@192.168.122.122 "sed -i 's/base/$NAME/g' /etc/hosts"
ssh root@192.168.122.122 "echo '192.168.122.2 master.example.com
master' >> /etc/hosts"
ssh root@192.168.122.122 "echo 'report=true' >>
/etc/puppet/puppet.conf"
ssh root@192.168.122.122 "sed -i
's/192.168.122.122/192.168.122.$IP/g' /etc/network/interfaces"
ssh root@192.168.122.122 "sed -i 's/START=no/START=yes/g'
/etc/default/puppet"
ssh root@192.168.122.122 "reboot"
#Por ultimo firmo el certificado del nuevo nodo.
sleep 26
ssh root@192.168.122.2 "puppet cert --sign $NAME.example.com"
sleep 2
ssh root@192.168.122.$IP "puppet agent -t"
```

Podemos ver el proceso:

Se le pasa como parámetro la ip del balanceador.

```
jose@josepc:~/puppet-project/scripts$ ./monitor.sh 3
1
1
1
401
Numero de conexiones altas, añadiendo balanceador
Notice: Revoked certificate with serial 18
Notice: Removing file Puppet::SSL::Certificate balanaux.example.com at
'/var/lib/puppet/ssl/ca/signed/balanaux.example.com.pem'
Notice: Removing file Puppet::SSL::Certificate balanaux.example.com at
'/var/lib/puppet/ssl/certs/balanaux.example.com.pem'
[master 4313938] automatic
 2 files changed, 7 insertions(+), 3 deletions(-)
Counting objects: 11, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 592 bytes, done.
Total 6 (delta 4), reused 0 (delta 0)
To git@github.com:joseluisjaime/puppet-project.git
   7d1b596..4313938  master -> master
remote: Counting objects: 11, done.
remote: Compressing objects: 100% (2/2), done.
```

```
remote: Total 6 (delta 4), reused 6 (delta 4)
Unpacking objects: 100% (6/6), done.
From https://github.com/joseluisjaime/puppet-project
   7d1b596..4313938  master     -> origin/master
Updating 7d1b596..4313938
Fast-forward
 hieradata/haproxy.yaml |    6 +++---
 manifests/nodes.pp     |    4 ++++
 2 files changed, 7 insertions(+), 3 deletions(-)
Connection to 192.168.122.2 closed.
Clonando base.img
| 1.1 GB     00:09


El clon 'balanaux' ha sido creado exitosamente.
Se ha iniciado el dominio balanaux


Notice: Signed certificate request for balanaux.example.com
Notice: Removing file Puppet::SSL::CertificateRequest balanaux.example.com at
'/var/lib/puppet/ssl/ca/requests/balanaux.example.com.pem'
info: Caching certificate for balanaux.example.com
info: Caching certificate_revocation_list for ca
info: Caching catalog for balanaux.example.com
info: Applying configuration version '1371766128'
notice: /Stage[main]/Apache::Install/Package[apache]/ensure: ensure changed 'purged' to 'present'
notice: /Stage[main]/Php::Install/Package[php]/ensure: ensure changed 'purged' to 'present'
notice: /Stage[main]/Php::Install/Package[php_mysql]/ensure: ensure changed 'purged' to 'present'
notice: /Stage[main]/Apache::Config/File[index.html]/content:
--- /var/www/index.html  2013-06-21 00:09:27.257170001 +0200
+++ /tmp/puppet-file20130621-2445-e9q673-02013-06-21 00:12:11.649170005 +0200
@@ -1,4 +1,8 @@
-<html><body><h1>It works!</h1>
-<p>This is the default web page for this server.</p>
-<p>The web server software is running but no content has been added, yet.</p>
-</body></html>
+<html>
+<body>
+<h1>Apache funciona correctamente</h1>
+<p>Nombre de la maquina: balanaux</p>
+<p>Ip de la maquina: 192.168.122.14</p>
+<p>Sistema operativo: Debian</p>
+</body>
+</html>


info: FileBucket adding {md5}21dde95d9d269cbb2fa6560309dca40c
info: /Stage[main]/Apache::Config/File[index.html]: Filebucketed /var/www/index.html to puppet with
sum 21dde95d9d269cbb2fa6560309dca40c
notice: /Stage[main]/Apache::Config/File[index.html]/content: content changed
```

```
'{md5}21dde95d9d269cbb2fa6560309dca40c' to '{md5}0de80ff6f623f0771fb72fc3364c85c9'

notice: /Stage[main]/Apache::Config/File[index.html]/owner: owner changed 'root' to 'www-data'

notice: /Stage[main]/Apache::Config/File[index.html]/group: group changed 'root' to 'www-data'

notice: /Stage[main]/Apache::Config/File[index.html]/mode: mode changed '0644' to '0640'

info: Class[Apache::Config]: Scheduling refresh of Class[Apache::Service]

notice: /Stage[main]/Php::Config/File[phpinfo]/ensure: defined content as
'{md5}767d787126c09cf4a3bc19218b47de4c'

info: Class[Apache::Service]: Scheduling refresh of Service[apache]

notice: /Stage[main]/Php::Config/File[phpini]/content:

--- /etc/php5/apache2/php.ini     2013-06-05 10:14:27.000000000 +0200

+++ /tmp/puppet-file20130621-2445-nqjo8v-02013-06-21 00:12:11.961170001 +0200

@@ -873,7 +873,7 @@

 [Date]

 ; Defines the default timezone used by the date functions

 ; http://php.net/date.timezone

-;date.timezone =

+date.timezone = "Europe/Madrid"


 ; http://php.net/date.default-latitude

 ;date.default_latitude = 31.7667


info: FileBucket adding {md5}29f7729de178b69c113a697bbc0035ce

info: /Stage[main]/Php::Config/File[phpini]: Filebucketed /etc/php5/apache2/php.ini to puppet with
sum 29f7729de178b69c113a697bbc0035ce

notice: /Stage[main]/Php::Config/File[phpini]/content: content changed
'{md5}29f7729de178b69c113a697bbc0035ce' to '{md5}ad41dd02198bdc6971d3b376ef114ed6'

notice: /Stage[main]/Apache::Service/Service[apache]: Triggered 'refresh' from 1 events

info: Creating state file /var/lib/puppet/state/state.yaml

notice: Finished catalog run in 205.25 seconds

Balanceador añadido

info: Caching catalog for client.example.com

info: Applying configuration version '1371766128'

notice: /Stage[main]/Haproxy::Config/File[haproxy config]/content:

--- /etc/haproxy/haproxy.cfg      2013-06-20 23:51:09.366259002 +0200

+++ /tmp/puppet-file20130621-3711-1alqyx7-0        2013-06-21 00:12:15.126259002 +0200

@@ -23,7 +23,7 @@

 balance roundrobin

 server client2 192.168.122.4

 server prueba1 192.168.122.10

-

+server balanaux 192.168.122.14



 listen stats 0.0.0.0:9000       #Listen on all IP's on port 9000


info: FileBucket adding {md5}9c278dc9f27d01b710f1af9a8e5548cb

info: /Stage[main]/Haproxy::Config/File[haproxy config]: Filebucketed /etc/haproxy/haproxy.cfg to
```

```
puppet with sum 9c278dc9f27d01b710f1af9a8e5548cb

notice: /Stage[main]/Haproxy::Config/File[haproxy config]/content: content changed
'{md5}9c278dc9f27d01b710f1af9a8e5548cb' to '{md5}f1dae1f0e9626e6f2a2594bbb163bf0f'

info: Class[Haproxy::Config]: Scheduling refresh of Class[Haproxy::Service]

info: Class[Haproxy::Service]: Scheduling refresh of Service[haproxy service]

notice: /Stage[main]/Haproxy::Service/Service[haproxy service]: Triggered 'refresh' from 1 events

notice: Finished catalog run in 0.68 seconds
```

Si entramos en las estadísticas de haproxy:

client

		Queue			Session rate			Sessions		
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit
	Frontend				0	0	-	0	0	50 000
☐	client2	0	0	-	0	0		0	0	-
☐	prueba1	0	0	-	0	0		0	0	-
☐	balanaux	0	0	-	0	0		0	0	-
	Backend	0	0		0	0		0	0	50 000

Al cambiar el fichero de configuración se ha reiniciado el servicio y se han reseteado las estadísticas.

9 - Puppet Dashboard

Puppet dashboard es una herramienta web que nos permite monitorizar el estado de los nodos así como los reportes que envían los nodos cada vez que hacen una petición al servidor.

Para ejecutar puppet dashboard es recomendable usar ruby1.8 y no ruby1.9, pues tiene varios bugs conocidos sin resolver.

Al tener instalado los repositorios de PuppetLabs podemos instalar el paquete desde los repositorios.

```
apt-get install puppet-dashboard
```

Esto nos instalara puppet-dashboard y las dependencias necesarias.

También nos hace falta una base de datos mysql.

```
apt-get install mysql-server
```

Creamos una base de datos y un usuario:

```
CREATE DATABASE dashboard_production CHARACTER SET utf8;
CREATE USER 'dashboard'@'localhost' IDENTIFIED BY 'dashboard';
GRANT ALL PRIVILEGES ON dashboard_production.* TO
'dashboard'@'localhost';
```

Una vez instalado todos los paquetes y creada la base de datos vamos a configurar el fichero:

“/etc/puppet-dashboard/database.yml”

Configuramos los parámetros con la base de datos que acabamos de crear:

```
production:
  database: dashboard_production
  username: dashboard
  password: dashboard
  encoding: utf8
  adapter: mysql
```

Ahora vamos a rellenar la base de datos para ello:

```
cd /usr/share/puppet-dashboard/
rake RAILS_ENV=production db:migrate
```

Ahora podemos iniciar los servicios

```
/etc/init.d/puppet-dashboard start
/etc/init.d/puppet-dashboard-workers start
```

En la parte del puppetmaster debemos editar y añadir a la sección [master] en "/etc/puppet/puppet.conf" estas dos lineas:

```
reports = store, http
reporturl = http://localhost:3000/reports/upload
```

Reiniciamos el servicio puppetmaster.

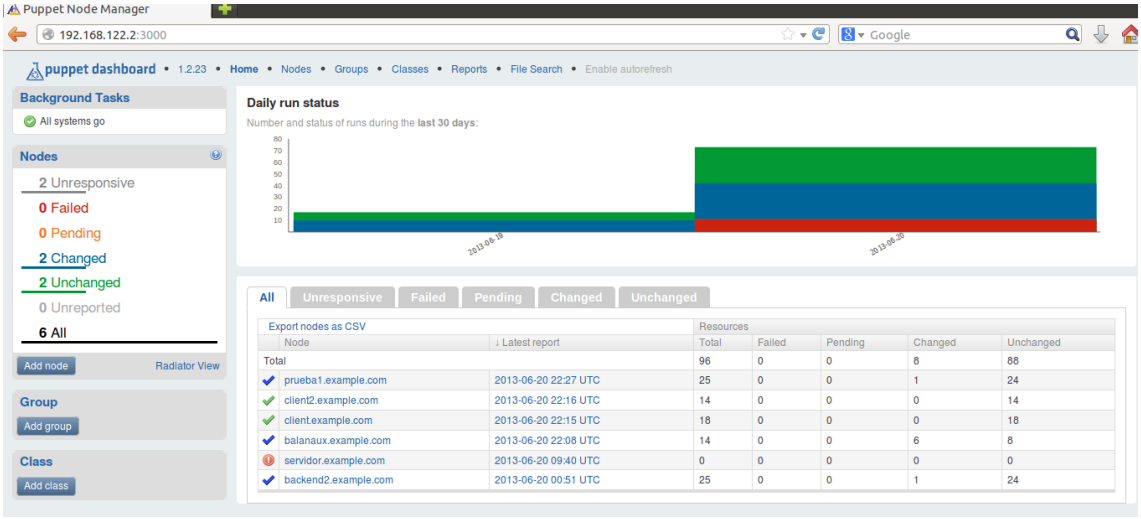
Y en la parte de los clientes añadimos al fichero de configuración en la sección [agent] la siguiente linea:

"/etc/puppet/puppet.conf"

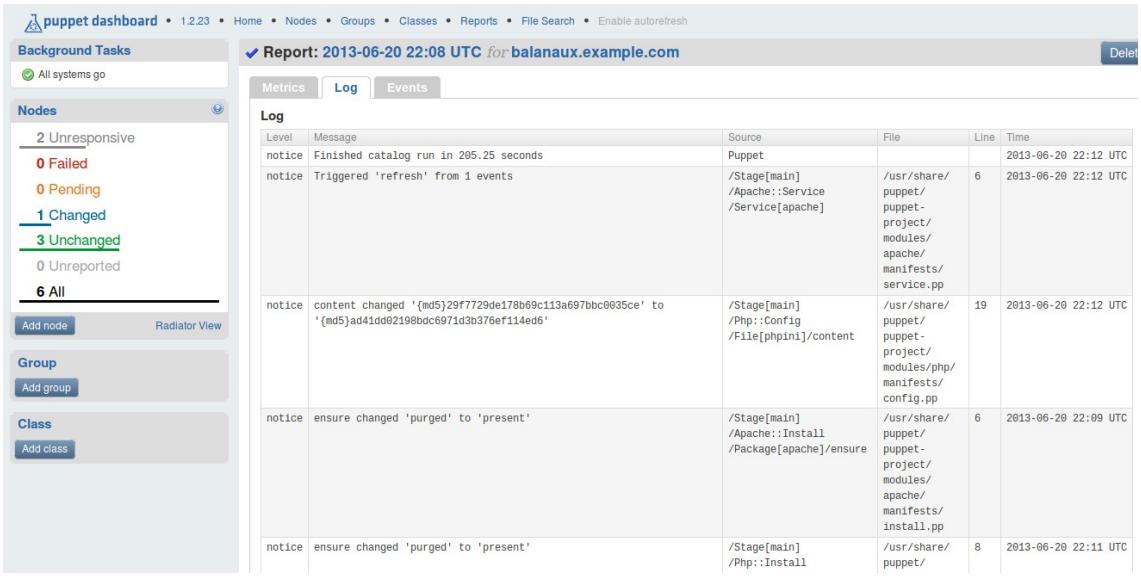
```
[agent]
server=master.example.com
report=true
```

Y reiniciamos el servicio.

Ya podemos entrar en http://192.168.122.2:3000 y ver el panel web:



Podemos ver si pulsamos en cualquier reporte, los cambios realizados:



10 - Webgrafia

https://puppetlabs.com/

https://puppetlabs.com/puppet/requirements/

http://www.puppetcookbook.com/

http://www.drivard.com/2013/05/install-your-puppetmaster-server-on-debian-wheezy-7-0/

https://github.com/puppetlabs/hiera/blob/master/docs/tutorials/getting_started.md

http://www.ant30.es/2011/11/puppet-gestion-de-configuracion-centralizada/

http://docs.puppetlabs.com/guides/installation.html#debian-and-ubuntu

http://www.mgoff.in/2010/07/14/haproxy-gathering-stats-using-socat/