PROYECTO INTEGRADO

Juan Antonio Beato Quiñones 2º ASIR

Tabla de contenido

1. Introducción

1.1 ¿Qué es Chef?

Chef es un sistema de automatización de infraestructura desarrollada por Opscode y hace más fácil desplegar servidores y aplicaciones a cualquier ubicación física, virtual o en la nube, sin importar el tamaño de la infraestructura. Cada organización se compone de uno (o más) Workstations (estaciones de trabajo), un único servidor, y cada uno de los nodos que va a ser configurado y mantenido por Chef.

Está escrito en Ruby y Erlang, el desplieque de cookbook, recipes, etc... lo realiza con ruby.

1.2 ¿Cómo lo uso?

Chef nos proporciona 3 opciones:

1.2.1 Hosted Chef: Nos proporcionan un Chef-server al que nos conectaremos desde nuestra máquina con el par de claves que nos proporcionan en la web, nosotros tendríamos nuestro Workstation que sincronizará con el servidor.

La versión FREE nos permite administrar hasta 5 servidores, estos son los precios:

	Free	Launch	Standard	Premium
Price per Month	Free	$120	$300	$600
Nodes	5	20	50	100
Standard Support	–	✓	✓	✓

1.2.2 Chef-Solo: Es una versión de código abierto de Chef que permite usar recetas en nodos que no tienen acceso a un Chef-Server.
Se ejecuta localmente y necesita tener la recetas en la máquina y todas sus dependencias.

Chef-Solo no incluye las siguiente funcionalidades:

- Almacenamiento de datos de los nodos
- Distribución centralizada de Cookbooks
- Autenticación y Autorización
- Una API centralizada que interactúa con los integrantes de la infraestructura
- Atributos persistentes

1.2.3 Chef-Server y Chef-Client: Esta opción permite instalarte tu propio servidor Chef en tu entorno y tus clientes Chef, te permitirá tener centralización de Cookbooks, puedes realizar configuraciones en los Nodos sin necesidad de tener la recetas en los nodos ya que todas están centralizadas en el servidor, la documentación sobre la instalación del servidor y los clientes la veremos a continuación.

Los componente que tendrá nuestro servidor serán los siguientes:

- **CouchDB:** Apache CouchDB es una base de datos documental sin esquemas, deja a un lado el modelo tradicional de almacenado de datos en columnas, tablas, filas, etc.. para CouchDB solo existen documentos, estos documentos son almacenados en JSON

- **RabbitMQ:** Es un broker de mensajería, es un intermediario que traduce los mensajes del sistema de un lenguaje a otro. Es un software de código abierto.

- **Chef-server:** Chef-Server actúa como un Hub de configuración. Almacena los Cookbooks, las políticas aplicadas a estas y los nodos definidos en la estructura. Los nodos utilizan Chef-Client para comunicarse con Chef-Server y pedirle las recetas, plantillas, etc...

- **Chef-server-webui:** Es una aplicación hecha en Ruby on Rails 3.0 que interfaz web a nuestro Chef-server.

- **Chef-Solr:** Utiliza Apache Sorl para la realización de búsquedas.

- **Chef-expander:** Es nuevo en Chef 0.10 y sustituye a chef-solr-indexer.

2. Conceptos Básicos

- **Node:** Es cualquier servidor, servidor virtual o instancia que este configurada para ser mantenido por chef-client.
- **Workstation:** Es una máquina que está configurada para usar knife, sincronizar con los repositorios y con el chef-server.
- **Rol:** Es una forma de definir atributos y recetas que queremos que se ejecuten en uno o carios nodos.
- **Cookbook:** Es la unidad fundamental de la distribución de configuración y políticas de Chef, define un escenario con todo lo necesario para realizar las instalaciones, configuraciones, etc..
- **Recipe:** Es el elemento de configuración más importante dentro del entorno Chef.
    - Están escritas en Ruby
    - Debe definir todo lo que queremos configurar de un sistema.
    - Se almacenan en Cookbooks
    - Pueden incluirse en otras recetas
    - Puede depender de una o varias recetas.

- **Attributes:** Un atributo es un detalle especifico de un nodo, se puede definir en un cookbook o recipe.
- **Run-list:** Es una lista en el que se pone los roles o recetas que queramos que se ejecute, se ejecutarán en el orden que pongamos.
- **Databags:** Es una variable global definida en un fichero JSON y que es accesible por un Chef-server y cargada en recetas.
- **Environs:** Es una forma de mapear u organizar el trabajo.

- **Ohai:** Es una herramienta que utiliza el servidor para detectar ciertas propiedades de cada nodo para después proporcionarlos al chef-client durante la ejecución de este.

3. Chef-Solo

La instalación de Chef-solo en Debian se realiza con las instalación del Chef-Client por lo que el procedimiento será el mismo que utilizaremos para instalar el cliente de chef, podemos verlo un poco más adelante en el punto 5.

Si la instalación la queremos realizar en una máquina Ubuntu podemos ayudarnos del script que nos proporciona la gente de Opscode ejecutando el siguiente comando:

```
curl -L https://www.opscode.com/chef/install.sh | bash
```

4. Chef-Server

4.1 Introducción

Chef-Server actúa como un Hub de configuración. Almacena los Cookbooks, las políticas aplicadas a estas y los nodos definidos en la estructura. Los nodos utilizan Chef-Client para comunicarse con Chef-Server y pedirle las recetas, plantillas, etc...

4.2 Instalación de Chef-Server en Ubuntu-Server 12.04 LTS

La instalación de Chef-Server la vamos a realizar en una máquina Ubuntu Server 12.04 LTS.

Hay varias formas de realizar la instalación, nosotros vamos a realizarla añadiendo los repositorios de Opscode e instalado desde repositorios.

Lo primero que tenemos que tener configurado correctamente será el FQDN de la máquina, no cuento con un servidor DNS por lo que yo lo haré en el fichero /etc/hosts:

```
127.0.0.1       servidor2.example.com   servidor2
#127.0.1.1      ubuntu

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Comprobamos la correcta configuración con el siguiente comando:

```
root@servidor2:~# hostname -f
```

```
servidor2.example.com
```

Hecho esto ya podemos comenzar con la instalación de Chef-Server, lo primero que haremos será añadir los repositorios de Opscode para el SO que tengamos en nuestra máquina:

```
echo "deb http://apt.opscode.com/ `lsb_release -cs`-0.10
main" | sudo tee /etc/apt/sources.list.d/opscode.list
```

Ahora vamos a añadir la llave GPG de Opscode para apt:

```
sudo mkdir -p /etc/apt/trusted.gpg.d
gpg --keyserver keys.gnupg.net --recv-keys 83EF826A
gpg --export packages@opscode.com | sudo tee
/etc/apt/trusted.gpg.d/opscode-keyring.gpg > /dev/null
```

Actualizamos:

```
sudo apt-get update
```

El siguiente paso es instalar el paquete "opscode-keyring" para mantener la llave actualizada:

```
sudo apt-get install opscode-keyring # permanent
upgradeable keyring
```

Realizamos el Upgrade:

```
sudo apt-get upgrade
```

Ahora ya podemos instalar Chef y Chef-Server:

```
sudo apt-get install chef chef-server
```

Durante la instalación recibimos el fallo a la hora de iniciar "chef-server-webui" y "chef-server", al finalizar la instalación podemos iniciarlos sin problemas:

```
/etc/init.d/chef-server restart
/etc/init.d/chef-server-webui start
```

Para comprobar que todo es correcto debemos mirar los siguientes puertos:

4000: Chef-Server
4040: Chef-Server-webui (Interfaz Web)
5984: CouchDB
5672: RabbitMQ
8983: Chef Solr

Lo podemos comprobar con el siguiente comando:

```
netstat -putan
```

4.2.1 Configuración de nuestro Workstation

Lo primero que haremos será crear el directorio .chef en el directorio del usuario que vayamos a usar:

```
mkdir -p ~/.chef
```

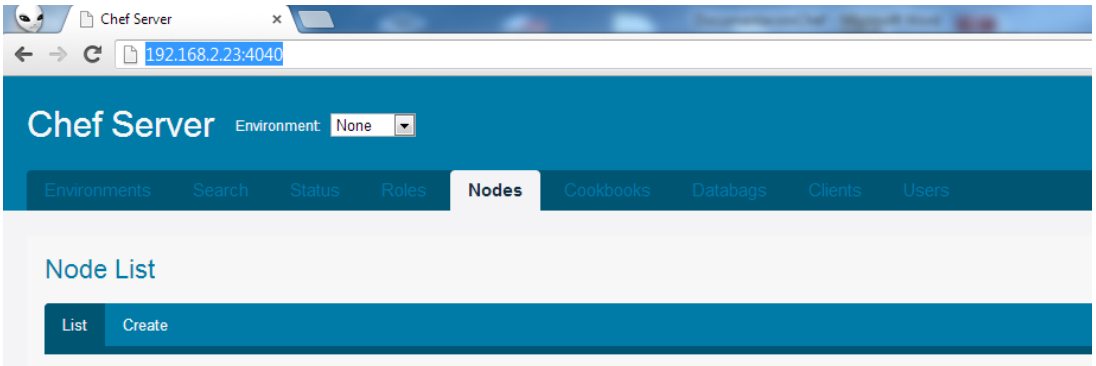
Copiamos los certificados necesarios creados por el servidor al directorio que hemos creado:

```
sudo cp /etc/chef/validation.pem /etc/chef/webui.pem
~/.chef
```

Ahora vamos a crear la API para un cliente en caso de que vayamos a trabajar desde otro equipo que no sea el servidor:

```
Please enter the chef server URL:
[http://servidor1.example.com:4000]
Please enter a clientname for the new client: [root]
juanbeato
Please enter the existing admin clientname: [chef-webui]
Please enter the location of the existing admin client's
private key: [/etc/chef/webui.pem] .chef/webui.pem
Please enter the validation clientname: [chef-validator]
Please enter the location of the validation key:
[/etc/chef/validation.pem] validation.pem
Please enter the path to a chef repository (or leave
blank):
Creating initial API user...
Created client[juanbeato]
Configuration file written to /root/.chef/knife.rb
```

Hecho esto ya tenemos nuestro servidor Chef configurado, ya podemos añadir nodos a través de la interfaz Web Chef-Server-Webui situada en la URL http://ipdelservidor:4040 o en línea de comando con Knife.



5. Chef-Client

5.1 Introducción

Chef-Client es un agente que se ejecuta localmente en todos los nodos registrados en el Chef-Server.

5.2 Instalación de Chef-Client en Ubuntu y Debian

Si no tenemos un servidor DNS deberemos definir el Chef-Server en el /etc/hosts:

```
127.0.0.1       cliente3.example.com    cliente3
192.168.2.23    server.example.com      server

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Esta instalación se va a llevar a cabo en un cliente Debian Wheezy por lo cual lo primero que deberemos hacer será instalar los siguientes paquetes:

```
apt-get install sudo wget lsb-release
```

El siguiente paso será añadir los repositorios de chef a nuestro sources.list, lo agregamos con el siguiente comando:

```
root@cliente3:~# echo "deb http://apt.opscode.com/
`lsb_release -cs`-0.10 main" | sudo tee
/etc/apt/sources.list.d/opscode.list
```

Ahora vamos a agregar la clave GPG de Opscode a apt:

```
sudo mkdir -p /etc/apt/trusted.gpg.d
gpg --keyserver keys.gnupg.net --recv-keys 83EF826A
gpg --export packages@opscode.com | sudo tee
/etc/apt/trusted.gpg.d/opscode-keyring.gpg > /dev/null
```

Actualizamos:

```
root@cliente3:~# apt-get update
```

Instalamos el paquete "opscode-keyring" que se encargará de tener las claves actualizadas:

```
sudo apt-get install opscode-keyring # permanent
upgradeable keyring
```

Realizamos el upgrade:

```
root@cliente3:~# apt-get upgrade
```

Ahora vamos a instalar chef, lo haremos con este comando:

```
root@cliente3:~# apt-get install chef
```

Instalamos ruby y algunas dependencias más:

```
apt-get install ruby ruby-dev libopenssl-ruby rdoc ri irb
build-essential wget ssl-cert curl
```

Instalamos la Gemas necesarias para Chef:

```
gem install chef --no-ri --no-rdoc
```

5.3 Configuración de Chef-client

En el cliente tenemos que crear el directorio /etc/chef donde deberemos colocar el archivo de configuración client.rb y la clave validation.pem.

Vamos al cliente y creamos el directorio:

```
mkdir -p /etc/chef
```

Ahora vamos al servidor para crear el fichero de configuración client.rb y la clave validation.pem:

```
root@server:~# knife configure client ./
Creating client configuration
Writing client.rb
Writing validation.pem
```

Estos dos ficheros debemos colocarlos en el directorio /etc/chef del cliente:

```
root@cliente3:~# scp root@192.168.2.15:/root/client.rb
/etc/chef/
root@192.168.2.15's password:
client.rb                                     100%  137
0.1KB/s   00:00
root@cliente3:~# scp root@192.168.2.15:/root/validation.pem
/etc/chef/
root@192.168.2.15's password:
validation.pem                                100% 1679
1.6KB/s   00:00
```

Abrimos el fichero de configuración client.rb y ponemos nuestro servidor si está puesto:

```
log_level        :info
log_location     STDOUT
chef_server_url  'http://server.example.com:4000'
validation_client_name 'chef-validator'
```

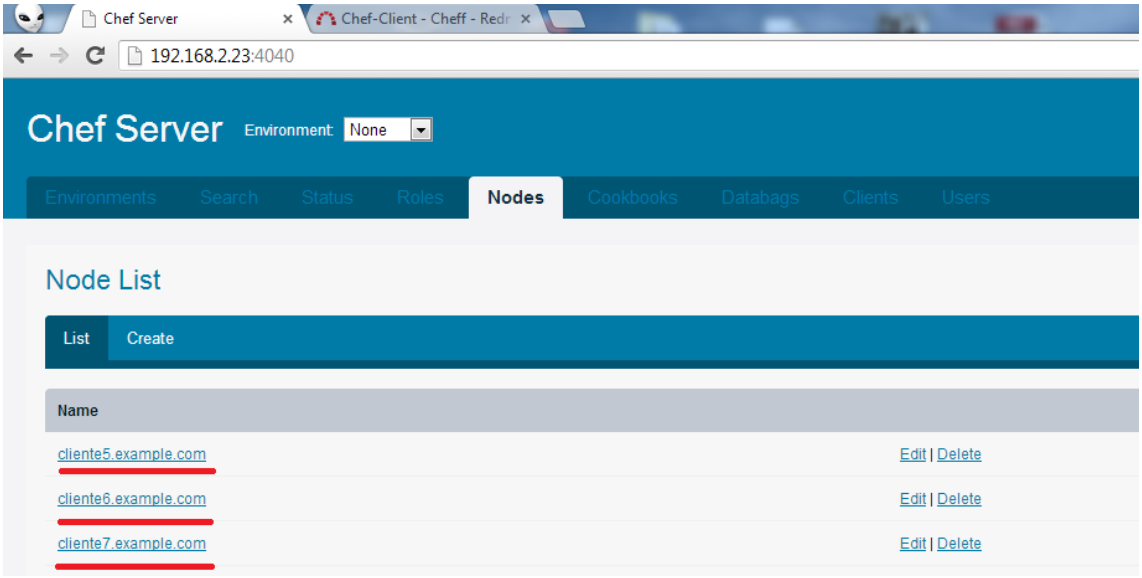
Por último reiniciamos el cliente:

```
root@cliente3:~# service chef-client restart
[ ok ] Restarting chef-client: chef-client.
```

Pasados unos minutos si vamos al servidor Chef y ejecutamos el siguiente comando podremos ver el nuevo nodo agregado al servidor:

```
root@servidor2:~# knife node list
  cliente5.example.com
  cliente6.example.com
  cliente7.example.com
  servidor2.example.com
```

Podemos hacer la misma comprobación desde la interfaz gráfica si vamos a la pestaña "Nodes", en cada nodo podremos ver los atributos de cada uno.



El intervalo de tiempo con el que se ejecutará chef-client en el nodo lo definimos en el fichero "/etc/chef/client.rb" con el parámetro "interval" seguido del tiempo en segundo:

```
log_level        :info
log_location     STDOUT
chef_server_url  'http://servidor-chef.example.com:4000'
validation_client_name 'chef-validator'
interval         900
```

6. Knife

6.1 ¿Qué es Knife?

Cualquier trabajador necesita de la ayuda de herramientas para facilitar la realización de las distintas labores, en el caso de los Chefs una de las herramientas imprescindibles es el cuchillo (knife).

Knife es una herramienta imprescindible para Chef, ya que nos permitirá la creación, listado, borrado, etc... de nodos, roles, cookbooks, etc...

Estas acciones también se pueden realizar desde la interfaz web de Chef pero, en mi caso, prefiero hacerlo desde un terminal con knife.

6.2 Usando Knife

La sintaxis de Knife es muy simple:

```
knife list [PATTERN...] (options)
```

6.2.1 Nodos

Configuración inicial:

Debemos definir la variable de entorno EDITOR e indicarle el editor que queremos utilizar, si no hacemos esto recibiremos el siguiente error:

```
root@server:~# knife node create prueba1
ERROR: RuntimeError: Please set EDITOR environment variable
```

Para definir esta variable ejecutamos el siguiente comando, en mi caso, el editor seleccionado es "nano":

```
root@server:~# export EDITOR=nano
```

- Listar nodos:

```
root@server:~# knife node list
debian.example.com
server.example.com
```

- Crear nodos:

La creación de un nodo se realiza con el siguiente comando, generará un fichero json donde podremos modificar los distintos parámetros del nodo, podremos asignar listas de ejecución, etc...

```
root@server:~# knife node create prueba1
Created node[prueba1]
```

- Eliminar nodos:

Con el siguiente comando podemos eliminar nodos:

```
root@server:~# knife node delete prueba1
Do you really want to delete prueba1? (Y/N) Y
Deleted node[prueba1]
```

- Editar nodos:

```
root@server:~# knife node edit lb.example.com
```

Este comando abrirá el fichero .json donde podemos modificar los parámetros del nodo.

- Añadir cookbooks y roles a la run_list de un nodo:

```
root@server:~# knife node run_list add lb.example.com
'recipe[balanceador]'
```

```
root@server:~# knife node run_list add lb.example.com
'role[decimo]'
```

- Eliminar cookbooks y roles de la run_list de un nodo:

```
root@server:~# knife node run_list remove lb.example.com
'recipe[balanceador]'
```

```
root@server:~# knife node run_list remove lb.example.com
'role[decimo]'
```

6.2.2 Roles

- Crear roles:

```
root@server:~# knife role create ejemplo
Created role[ejemplo]
```

Esto nos abrirá un fichero .json con las características del rol.

- Eliminar roles:

```
root@server:~# knife role delete ejemplo
Do you really want to delete ejemplo? (Y/N) Y
Deleted role[ejemplo]
```

- Editar roles:

```
root@server:~# knife role edit ejemplo2
```

Este comando abrirá el fichero .json donde podemos modificar los parámetros del rol.

- Listar roles:

```
root@servidor-chef:~# knife role list
ejemplo
```

6.2.3 Cookbooks

- Crear cookbooks

```
root@servidor-chef:~# knife cookbook create ejemplo
** Creating cookbook ejemplo
** Creating README for cookbook: ejemplo
** Creating CHANGELOG for cookbook: ejemplo
** Creating metadata for cookbook: ejemplo
```

- Actualizar cookbooks

```
root@servidor-chef:~# knife cookbook upload -a
Uploading balanceador    [0.1.0]
Uploading bitnami        [0.1.0]
Uploading ejemplo        [0.1.0]
Uploading iscsi          [0.1.0]
Uploading iscsi-cli      [0.1.0]
Uploading nodo1cook      [0.1.0]
Uploading nodo2cook      [0.1.0]
Uploading ocfs           [0.1.0]
Uploading pruebas        [0.1.0]
Uploaded all cookbooks.
```

- Borrar cookbooks

```
root@servidor-chef:~# knife cookbook delete ejemplo
Do you really want to delete ejemplo version 0.1.0? (Y/N) Y
Deleted cookbook[ejemplo version 0.1.0]
```

7. Ejemplos

En este apartado vamos a ver algunos ejemplos simple, como instalar un paquete, ejecutar un comando, etc...

- Instalar un paquete:

```
package "apache2" do
  action :install
end
```

- Iniciar, parar o reiniciar un servicio:

```
service "apache2" do
  action :(restart|start|stop)
```

```
end
```

- Ejecutar un comando del sistema:

```
execute "update" do
  command "apt-get update"
  action :run
end
```

- Ejecutar si no, esta opción es muy útil a la hora de crear directorio, ficheros y demás, ejecuta un comando para ver si existe y en caso de que no ejecuta el comando que le indiquemos, aquí un ejemplo:

```
execute "crear_run" do
  command "mkdir /var/run/haproxy/"
  not_if "find /var/run/haproxy"
  action :run
end
```

- Crear fichero a partir de una plantilla, podemos indicarle propietario, grupo, permisos, etc:

```
template node['lb']['defa'] + '/haproxy' do
  source 'default.erb'
  mode 00644
  owner 'root'
  group 'root'
end
```

- Definir atributo, esto debe hacerse en el directorio attributes del cookbook correspondiente en un fichero de extensión .rb:

```
default["nodo1"]["ip1"] = "192.168.56.102"
```

- Utilizar atributo definido anteriormente en una receta:

```
template node['lb']['sysctl'] do
  source 'sysctl.erb'
  mode 00644
  owner 'root'
  group 'root'
end
```

8. Despliegue de Servidor LAMP y Wordpress con Chef

8.1 Introducción

En este apartado vamos a realizar el despliegue de un servidor LAMP y la aplicación Wordpress tanto con Chef-solo como con Chef-Server en varios nodos.

La instalación de Chef-Server, Chef-Client y Chef-Solo la tenemos en los apartados anteriores.

8.2 Despliegue en Chef-Solo

Para desplegar el servidor LAMP y la aplicación Wordpress vamos a crear un nuevo "Cookbook", antes de hacer esto necesitamos una estructura donde tengamos nuestros ficheros y Cookbooks, vamos a utilizar la creada por Opscode los creadores de Chef, lo llaman Chef-Repository:

```
root@Aspire-One-Juan:~# wget
http://github.com/opscode/chef-repo/tarball/master
root@Aspire-One-Juan:~# tar -zxf master
root@Aspire-One-Juan:~# mv opscode-chef-repo* chef-repo
```

Ya tenemos la estructura:

```
root@Aspire-One-Juan:~# ls chef-repo/
certificates  config     data_bags     LICENSE   README.md
chefignore    cookbooks  environments  Rakefile  roles
```

Nuestro primer cookbook estará en el directorio cookbooks, para administrar las cookbooks utilizaremos knife, lo primero que haremos será indicarle donde almacenamos las cookbooks:

```
root@Aspire-One-Juan:~/chef-repo# mkdir .chef
root@Aspire-One-Juan:~/chef-repo# echo "cookbook_path
['/root/chef-repo/cookbooks' ]" > .chef/knife.rb
```

El siguiente paso es crear el nuevo cookbook:

```
root@Aspire-One-Juan:~/chef-repo# knife cookbook create
soloword
** Creating cookbook soloword
** Creating README for cookbook: soloword
** Creating CHANGELOG for cookbook: soloword
** Creating metadata for cookbook: soloword
```

Nos situamos en el directorio cookbooks y descargamos el cookbook "apache2" y lo descomprimimos:

```
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download apache2
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf
apache2*
```

También descargaremos el cookbook "apt" que nos actualizará antes de instalar cualquier paquete:

```
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download apt
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf apt*
```

Nos situamos en el directorio de nuestro cookbook "soloword" , abrimos el fichero "metadata.rb" e indicamos que depende del cookbook "apache2"

```
name             'soloword'
maintainer       'YOUR_COMPANY_NAME'
maintainer_email 'YOUR_EMAIL'
license          'All rights reserved'
description      'Installs/Configures soloword'
long_description IO.read(File.join(File.dirname(__FILE__),
'README.md'))
version          '0.1.0'

depends "apache2"
```

Situados en el directorio de nuestro cookbook vamos a recipes/default.rb para indicarle la receta de apache:

```
#
# Cookbook Name:: soloword
# Recipe:: default
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

include_recipe "apache2"
```

Ahora nos situamos en el directorio "Chef-repo" donde crearemos el fichero "solo.rb" en el que le indicaremos la cache de ficheros y el directorio donde se encuentran los cookbooks:

```
file_cache_path "/root/chef-solo"
cookbook_path "/root/chef-repo/cookbooks"
```

ahora creamos el fichero web.json , con este fichero le indicamos que primero ejecute el cookbook apt y después soloword:

```
{
  "run_list": [ "recipe[apt]", "recipe[soloword]" ]
}
```

Nos situamos en el directorio "Chef-repo" y lo ejecutamos para realizar la instalación del apache2:

```
root@Aspire-One-Juan:~/chef-repo# chef-solo -c solo.rb -j
web.json
```

Podemos comprobar que está instalado ejecutando el siguiente comando:

```
root@Aspire-One-Juan:~/chef-repo# aptitude search apache2
i   apache2                          - Metapaquete del
servidor HTTP Apache
```

El siguiente paso será instalar "MySQL", para esto vamos a descargar el cookbook de MySQL y a descomprimirlo en el directorio de cookbooks donde hemos situado las anteriores:

```
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download mysql
Downloading mysql from the cookbooks site at version 3.0.0
to /root/chef-repo/cookbooks/mysql-3.0.0.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/mysql-
3.0.0.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf mysql*
```

Ahora como hicimos anteriormente vamos a nuestro cookbook "soloword" y añadimos la siguiente línea en el fichero metadata.rb:

```
name             'soloword'
maintainer       'YOUR_COMPANY_NAME'
maintainer_email 'YOUR_EMAIL'
license          'All rights reserved'
description      'Installs/Configures soloword'
long_description IO.read(File.join(File.dirname(__FILE__),
'README.md'))
version          '0.1.0'

depends "apache2"
depends "mysql"
```

También debemos indicar las recetas que vamos a utilizar de este cookbook, esto lo haremos en el fichero "recipes/default.rb" que hay dentro del directorio de nuestro cookbook "soloword":

```
#
# Cookbook Name:: phpapp
# Recipe:: default
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#
```

```
include_recipe "apache2"
include_recipe "mysql::client"
include_recipe "mysql::server"
```

Ahora tendremos que descargar los cookbooks "openssl" y "build-essential", ya que el cookbook "mysql" depende de ellos:

```
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download
opensslDownloading openssl from the cookbooks site at
version 1.0.2 to /root/chef-repo/cookbooks/openssl-
1.0.2.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/openssl-
1.0.2.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf
openssl*.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download build-essential
Downloading build-essential from the cookbooks site at
version 1.4.0 to /root/chef-repo/cookbooks/build-essential-
1.4.0.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/build-essential-
1.4.0.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf build-
essential-*.tar.gz
```

Lo siguiente que tenemos que hacer es editar el fichero "web.json" que está situado en el directorio "Chef-repo" y añadir las contraseñas que necesitaremos proporcionarle:

```
{
  "mysql": {"server_root_password": "usuario",
"server_debian_password": "usuario",
"server_repl_password": "usuario"},
  "run_list": [ "recipe[apt]", "recipe[soloword]" ]
}
```

Ejecutamos chef-solo situado en el directorio "Chef-repo" para realizar la instalación de mysql:

```
root@Aspire-One-Juan:~/chef-repo# chef-solo -c solo.rb -j
web.json
```

El siguiente paso será descargar el cookbook "php":

```
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download php
Downloading php from the cookbooks site at version 1.2.0 to
/root/chef-repo/cookbooks/php-1.2.0.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/php-1.2.0.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf
php*.tar.gz
```

La cookbook de "php" depende de "xml" por lo que también la descargaremos:

```
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download xml
Downloading xml from the cookbooks site at version 1.1.2 to
/root/chef-repo/cookbooks/xml-1.1.2.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/xml-1.1.2.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf xml-
*.tar.gz
```

Volvemos a editar el fichero metadata.rb para añadir la dependencia del cookbook "php":

```
name             'phpapp'
maintainer       'YOUR_COMPANY_NAME'
maintainer_email 'YOUR_EMAIL'
license          'All rights reserved'
description      'Installs/Configures phpapp'
long_description IO.read(File.join(File.dirname(__FILE__),
'README.md'))
version          '0.1.0'

depends "apache2"
depends "mysql"
depends "php"
```

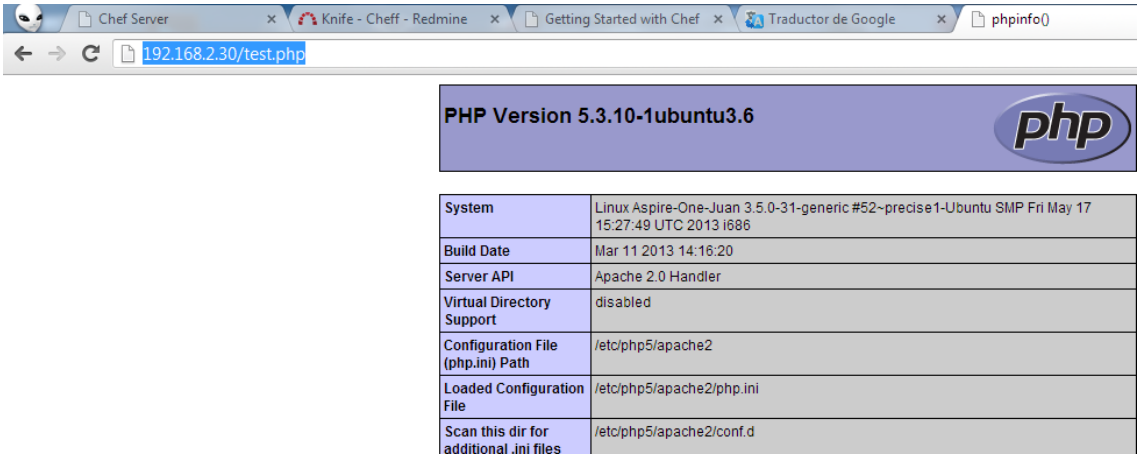
Editamos el fichero "recipes/default.rb" de nuestro cookbook para indecarle las recetas:

```
#
# Cookbook Name:: phpapp
# Recipe:: default
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

include_recipe "apache2"
include_recipe "mysql::client"
include_recipe "mysql::server"
include_recipe "php"
include_recipe "php::module_mysql"
include_recipe "apache2::mod_php5"

apache_site "default" do
  enable true
end
```

Ejecutamos el chef-solo en el directorio "Chef-repo" para que realice la instalación de php:

```
root@Aspire-One-Juan:~/chef-repo# chef-solo -c solo.rb -j
web.json
```

Para probar el correcto funcionamiento de lo realizado he creado el fichero "test.php" en la ruta "/var/www", podemos comprobar que todo es correcto poniendo la siguiente URL en el navegador http:/ipdelservidor/test.php



Lo siguiente que haremos será configurar la base de datos para ello nos ayudaremos del cookbook "database", esta cookbook depende de "postgresql", "xfs", "aws".

```
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download database
Downloading database from the cookbooks site at version
1.4.0 to /root/chef-repo/cookbooks/database-1.4.0.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/database-
1.4.0.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf
database-*.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download postgresql
Downloading postgresql from the cookbooks site at version
3.0.0 to /root/chef-repo/cookbooks/postgresql-3.0.0.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/postgresql-
3.0.0.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf
postgresql-*.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download xfs
Downloading xfs from the cookbooks site at version 1.1.0 to
/root/chef-repo/cookbooks/xfs-1.1.0.tar.gz
Cookbook saved: /root/chef-repo/cookbooks/xfs-1.1.0.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf xfs-
*.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# knife cookbook
site download aws
Downloading aws from the cookbooks site at version 0.101.0
to /root/chef-repo/cookbooks/aws-0.101.0.tar.gz
```

```
Cookbook saved: /root/chef-repo/cookbooks/aws-
0.101.0.tar.gz
root@Aspire-One-Juan:~/chef-repo/cookbooks# tar zxf aws-
*.tar.gz
```

Editamos el fichero "metadata.rb" y añadimos la dependencia del cookbook "database":

```
name             'phpapp'
maintainer       'YOUR_COMPANY_NAME'
maintainer_email 'YOUR_EMAIL'
license          'All rights reserved'
description      'Installs/Configures phpapp'
long_description IO.read(File.join(File.dirname(__FILE__),
'README.md'))
version          '0.1.0'

depends "apache2"
depends "mysql"
depends "php"
depends "database"
```

Ahora editamos el fichero "recipes/default.rb" de nuestro cookbook "soloword" y añadimos las recetas:

```
#
# Cookbook Name:: phpapp
# Recipe:: default
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

include_recipe "apache2"
include_recipe "mysql::client"
include_recipe "mysql::server"
include_recipe "php"
include_recipe "php::module_mysql"
include_recipe "apache2::mod_php5"
include_recipe "mysql::ruby"

apache_site "default" do
  enable true
end

mysql_database node['soloword']['database'] do
  connection ({:host => 'localhost', :username => 'root',
:password => node['mysql']['server_root_password']})
  action :create
end
```

```
mysql_database_user node['soloword']['db_username'] do
  connection ({:host => 'localhost', :username => 'root',
:password => node['mysql']['server_root_password']})
  password node['soloword']['db_password']
  database_name node['soloword']['database']
  privileges [:select,:update,:insert,:create,:delete]
  action :grant
end
```

Tenemos que definir los atributos que vamos a utilizar en el fichero "recipes/default.rb" por lo que lo definimos en la siguiente ruta dentro de nuestro cookbook "attributes/default.rb":

```
default["soloword"]["database"] = "soloword"
default["soloword"]["db_username"] = "soloword"
```

Indicamos la contraseña en el fichero "web.json" situado en el directorio "Chef-repo":

```
{
  "mysql":       {"server_root_password":       "usuario",
"server_debian_password":                       "usuario",
"server_repl_password": "usuario"},
"soloword": {"db_password": "usuario"},
  "run_list": [ "recipe[apt]", "recipe[soloword]" ]
}
```

Ejecutamos chef-solo para crear la base de datos y el usuario:

```
root@Aspire-One-Juan:~/chef-repo# chef-solo -c solo.rb -j
web.json
```

El siguiente paso es crear el directorio donde vamos a almacenar "wordpress" y descargarlo, para ellos lo primero que haremos será editar "recipes/default.rb" y añadir las siguientes líneas:

```
wordpress_latest = Chef::Config[:file_cache_path] +
"/wordpress-latest.tar.gz"

remote_file wordpress_latest do
  source "http://wordpress.org/latest.tar.gz"
  mode "0644"
end

directory node["soloword"]["path"] do
  owner "root"
  group "root"
  mode "0755"
  action :create
  recursive true
end

execute "untar-wordpress" do
```

```
  cwd node['soloword']['path']
  command "tar --strip-components 1 -xzf " +
wordpress_latest
  creates node['soloword']['path'] + "/wp-settings.php"
end
```

Ahora tenemos que definir el atributo en "attributes/default.rb":

```
default["soloword"]["database"] = "soloword"
default["soloword"]["db_username"] = "soloword"
default["soloword"]["path"] = "/var/www/soloword"
```

Ejecutamos chef-solo y se creará el directorio y se descargará el paquete wordpress:

```
root@Aspire-One-Juan:~/chef-repo# chef-solo -c solo.rb -j
web.json
```

Ahora vamos a crear las plantillas necesarias, lo primero que haremos será añadir las siguientes líneas en el fichero "recipes/default.rb":

```
wp_secrets = Chef::Config[:file_cache_path] + '/wp-
secrets.php'

remote_file wp_secrets do
  source 'https://api.wordpress.org/secret-key/1.1/salt/'
  action :create_if_missing
  mode 0644
end
```

Ahora vamos al directorio "templates" de nuestro cookbook "soloword" y creamos el fichero "templates/default/wp-config.php.erb" con el siguiente contenido:

```
<?php

define('DB_NAME', '<%= @database %>');
define('DB_USER', '<%= @user %>');
define('DB_PASSWORD', '<%= @password %>');
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8');
define('DB_COLLATE', '');

<%= @wp_secrets %>

$table_prefix  = 'wp_';

define('WPLANG', '');
define('WP_DEBUG', false);

if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');
```

```
require_once(ABSPATH . 'wp-settings.php');

nano recipes/default.rb

salt_data = ''

ruby_block 'fetch-salt-data' do
  block do
    salt_data = File.read(wp_secrets)
  end
  action :create
end

template node['soloword']['path'] + '/wp-config.php' do
  source 'wp-config.php.erb'
  mode 0755
  owner 'root'
  group 'root'
  variables(
    :database        => node['soloword']['database'],
    :user            => node['soloword']['db_username'],
    :password        => node['soloword']['db_password'],
    :wp_secrets      => salt_data)
end
```

El siguiente paso es crear la plantilla para crear el virtualhost para wordpress, para ello creamos el fichero "templates/default/site.conf.erb" con el siguiente contenido:

```
# Auto generated by Chef. Changes will be overwritten.

<VirtualHost *:80>
  ServerName <%= @params[:server_name] %>
  DocumentRoot <%= @params[:docroot] %>

  <Directory <%= @params[:docroot] %>>
    Options FollowSymLinks
    AllowOverride FileInfo Options
    AllowOverride All
    Order allow,deny
    Allow from all
  </Directory>

  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>

</VirtualHost>
```

Por último editamos el fichero "recipes/default.rb" y añadimos estas líneas:

```
#
# Cookbook Name:: phpapp
# Recipe:: default
#
# Copyright 2013, YOUR_COMPANY_NAME
#
# All rights reserved - Do Not Redistribute
#

include_recipe "apache2"
include_recipe "mysql::client"
include_recipe "mysql::server"
include_recipe "php"
include_recipe "php::module_mysql"
include_recipe "apache2::mod_php5"
include_recipe "mysql::ruby"

apache_site "default" do
  enable false
end

......

web_app 'soloword' do
  template 'site.conf.erb'
  docroot node['soloword']['path']
  server_name node['soloword']['server_name']
end
```

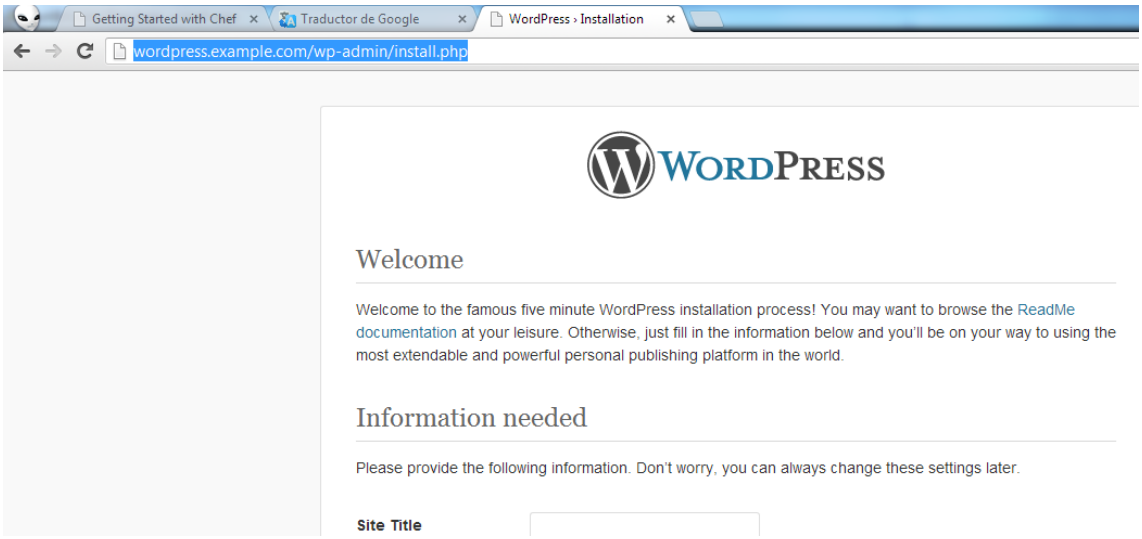
El último paso es añadir el atributo en el fichero "attributes/default.rb":

```
default['soloword']['server_name'] = "soloword"
```

Ejecutamos chef-solo y ya tendremos nuestra aplicación instalada:

```
root@Aspire-One-Juan:~/chef-repo# chef-solo -c solo.rb -j
web.json
```

Lo comprobamos poniendo la dirección en el navegador y vemos que efectivamente todo ha sido correcto:



Ya tenemos la receta lista, si quisiéramos montar todo en otro servidor solo tendríamos que instalar chef-solo y copiarlos el chef-repo y ejecutarlo.

8.3 Despliegue simultáneo en varios nodos con Chef-Server

La primera diferencia que encontramos es que en Chef-Server no existe el fichero solo.rb donde le indicamos la ubicación de las cookbooks, aquí debemos indicarlo en el fichero "/root/.chef/knife.rb" con la siguiente línea.

```
cookbook_path ['/root/chef-repo/cookbooks' ]
```

El proceso de despliegue en Chef server sería similar al realizado en el apartado anterior con Chef-Solo, pero en chef-server los atributos definidos en web.json como puede ser la contraseña de mysql, server_name, etc.. los definiríamos en la ruta "attributes/default.rb" junto con los demás que hemos añadido.

```
default["nueva"]["database"] = "nueva"
default["nueva"]["db_username"] = "nueva"
default["nueva"]["db_password"] = "nueva"
default["nueva"]["path"] = "/var/www/nueva"
default['nueva']['server_name'] = "nueva"
default["nueva"]["server_name"] = "nueva.example.com"
```

La última diferencia que debemos tener en cuenta con respecto a chef-solo es que el parámetro "run_list" del fichero "web.json" donde indicamos el orden en el que se ejecutarán los cookbooks debemos definirlo en el fichero de configuración del nodo en el que queremos que se ejecute o en un rol si queremos que se ejecute en varios nodos.

Aquí un ejemplo de cómo quedaría el rol:

```
{
  "env_run_lists": {
  },
```

```
  "name": "nuevo",
  "run_list": [
    "recipe[apt]",
    "recipe[nueva]"
  ],
  "chef_type": "role",
  "json_class": "Chef::Role",
  "override_attributes": {
  },
  "description": "",
  "default_attributes": {
  }
}
```

Donde los cookbooks se irían ejecutando en el orden en el que están puestas, una vez indicado en el rol, solo tenemos que añadir el rol a los nodos en los que queremos ejecutarlas y ya lo tenemos terminado.

Ahora solo tendríamos que esperar a que se ejecuten en cada cliente cuando se cumpla el intervalo de tiempo, en caso de querer ejecutarlo antes podemos ejecutarlo con el comando chef-client en cada uno de los cliente.

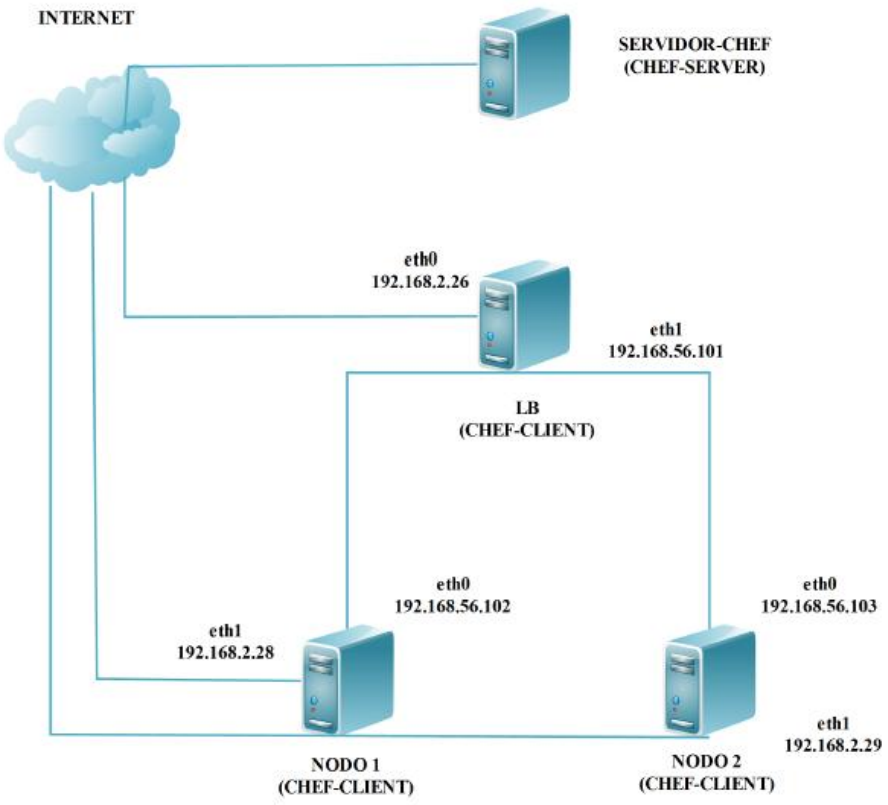
9. Despliegue de Balanceador de cargas con Chef-Server

En este apartado el objetivo es la instalación de dos servidores web simples en las máquinas NODO1 y NODO2 y la instalación y configuración de un balanceador de carga con HAproxy en la máquina LB.

Los nodos tienen una segunda interfaz de red que los conecta a internet ya que he tenido problemas con Chef-Server al hacer SNAT ya que la trama al salir por eth0 lleva la misma IP en los 3 casos y Chef devuelve el siguiente error si intentamos conectarnos:

```
ssh_exchange_identification: read: Connection reset by peer
```

Aquí tenemos el esquema de red que se utilizará en el ejemplo del balanceador:



Vamos a empezar creando el cookbook donde se crearan las recetas "default.rb" que se encargará de la instalación y configuración del balanceador, "nodo1" que se encargará de instalar el servidor web del NODO1 y "nodo2" que se encargará de instalar el servidor web del NODO2:

```
root@servidor-chef:~# knife cookbook create balanceador
** Creating cookbook balanceador
** Creating README for cookbook: balanceador
** Creating CHANGELOG for cookbook: balanceador
** Creating metadata for cookbook: balanceador
```

Lo primero que voy hacer será editar la receta "default.rb" ubicada en la ruta "/root/chef-repo/cookbooks/balanceador/recipes/", se realizarán las siguientes acciones por este orden y separadas por bloques de código:

- Reemplazo de los repositorios de Wheezy por los de Squeeze ya que en los de wheezy no está el paquete "haproxy".
- Ejecuto un apt-get update para actualizar los repositorios.
- Instalación del paquete "haproxy".
- Sustituimos el 0 por 1 en "/etc/default/haproxy" para poder iniciarlo.
- Creamos el directorio "/var/run/haproxy".
- Cambiamos los permisos del directorio que acabamos de crear.
- Cambiamos la ruta en "/etc/init.d/haproxy" por la nueva.

- Sustituimos el fichero de configuración de haproxy por el nuestro con la configuración que queramos aplicar.
- Hacemos que el bit de forward se mantenga siempre a 1.
- Y por último iniciamos haproxy.

Esta es la receta con el código que ejecuta dichas acciones:

```
#######Cambiamos los repositorios#########
cookbook_file "/etc/apt/sources.list" do
  source "repo"
  owner "root"
  group "root"
  mode 00644
end

#####habilitamos el bit de forwarding######
execute "enru" do
  command "echo 1 > /proc/sys/net/ipv4/ip_forward"
  action :run
end

#######Ejecutamos el update##########
execute "update" do
  command "apt-get update"
  action :run
end

#######Instalamos el paquete haproxy########
package "haproxy" do
  action :install
end

######Sustitucion del 0 por 1 en
/etc/default/haproxy##############
cookbook_file "/etc/default/haproxy" do
  source "haproxy-default"
  owner "root"
  group "root"
  mode 00644
end

######Crear el fichero /var/run/haproxy######
execute "crear_run" do
  command "mkdir /var/run/haproxy/"
  not_if "find /var/run/haproxy"
  action :run
end

######Dar los permisos correctos#######
execute "permisos_ha" do
  command "chown -R haproxy:haproxy /var/run/haproxy/"
```

```
  action :run
end

#######Indicamos la nueva ruta en
/etc/init.d/haproxy#######
cookbook_file "/etc/init.d/haproxy" do
  source "haproxy-init"
  owner "root"
  group "root"
  mode 00755
end

######Sustituimos el fichero de configuracion
/etc/haproxy/haproxy.cfg###
template node['lb']['conf'] + '/haproxy.cfg' do
  source 'conf.erb'
  mode 00644
  owner 'root'
  group 'root'
  variables(
    :balan        => node['lb']['balan'],
    :ip1        => node['nodo1']['ip1'],
    :ip2        => node['nodo2']['ip2'])
end

####Hacemos que el bit de forward este siempre a 1#####
cookbook_file "/etc/sysctl.conf" do
  source "sysctl"
  owner "root"
  group "root"
  mode 00755
end

#####Iniciamos el servicio haproxy#####
service "haproxy" do
  action :start
end
```

Esta recete utiliza atributos y varias plantillas que vamos a ver a continuación, las plantillas estarían situadas en "/root/chef-repo/cookbooks/balanceador/templates/default/" y los atributos en el siguiente fichero situado en esta ruta "/root/chef-repo/cookbooks/balanceador/attributes/default.rb"

Vamos primero con los atributos y después con el contenido de las plantillas, como vamos a ver, tanto las rutas como las IPs están definidas en atributos para que la receta sea lo más reutilizable posible:

```
root@servidor-chef:~# cat chef-
repo/cookbooks/balanceador2/attributes/default.rb
default["lb"]["repo"] = "/etc/apt"
default["lb"]["defa"] = "/etc/default"
```

```
default["lb"]["init"] = "/etc/init.d"
default["lb"]["conf"] = "/etc/haproxy"
default["lb"]["balan"] = "192.168.2.26"
default["nodo1"]["ip1"] = "192.168.56.102"
default["nodo2"]["ip2"] = "192.168.56.103"
default["lb"]["sysctl"] = "/etc/sysctl.conf"
default["nodo1"]["path"] = "/var/www"
```

Ahora vamos a ver el contenido de los distintos ficheros estáticos que vamos a utilizar, estos ficheros estarán situados en /root/chef-repo/cookbooks/balanceador/files/":

- repo: Esta plantilla la utilizo para cambiar los repositorios, este es el contenido:

```
deb http://ftp.es.debian.org/debian/ squeeze main
deb-src http://ftp.es.debian.org/debian/ squeeze main
```

- haproxy-default: Aquí ponemos el fichero que sustituimos por "/etc/default/haproxy":

```
# Set ENABLED to 1 if you want the init script to start
haproxy.
ENABLED=1
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
```

- haproxy-init: Este es el fichero que sustituiremos por "/etc/init.d/haproxy":

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          haproxy
# Required-Start:    $local_fs $network $remote_fs
# Required-Stop:     $local_fs $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: fast and reliable load balancing
reverse proxy
# Description:       This file should be used to start and
stop haproxy.
### END INIT INFO

# Author: Arnaud Cornet <acornet@debian.org>

PATH=/sbin:/usr/sbin:/bin:/usr/bin
PIDFILE=/var/run/haproxy/haproxy.pid
CONFIG=/etc/haproxy/haproxy.cfg
HAPROXY=/usr/sbin/haproxy
EXTRAOPTS=
ENABLED=0

                        ..............
```

- sysctl: Con este fichero haremos que el bit de forward se mantenga a 1:

```
#
# /etc/sysctl.conf - Configuration file for setting system
variables
# See /etc/sysctl.d/ for additional system variables
# See sysctl.conf (5) for information.
#

#kernel.domainname = example.com

# Uncomment the following to stop low-level messages on
console
#kernel.printk = 3 4 1 3

##############################################################
###3
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection
(reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for
IPv4
net.ipv4.ip_forward=1

                        .............
```

Ahora vamos a ver el contenido de las plantillas que utilizará nuestra receta y que se encuentran en la siguiente ruta "/root/chef-repo/cookbooks/balanceador/templates/default/"

- conf.erb: Esta plantilla es la que sustituiremos por el fichero "/etc/haproxy/haproxy"

```
# definicion global
global
        # definimos el log local mediante socket
        log /dev/log syslog
        # numero maximo de conexiones globales
        maxconn 4096
```

```
        # usuario y grupo que ejecuta haproxy
        user haproxy
        group haproxy
        # indicamos que se va a ejecutar como
demonio/servicio
        daemon
        # si queremos depurar podemos utilizar las
siguientes directivas
        # deshabilita la ejecuciï¿½n en background y saca
toda la informaciï¿½n por salida estï¿½ndar
        #debug
        # hace que no se muestre informaciï¿½n en el
arranque
        #quiet
        # indicamos el path del fichero de PID
        pidfile /var/run/haproxy/haproxy.pid

# configuracion por defcto que se aplica a todos los
frontend salvo si sobreescribe
defaults
        # se usa el log definido en la seccion global
        log global
        # indicamos que el modo es http ya que se trata de
un balanceador web
        mode http
        # indicamos el numero de reintentos de chequeo de
un servidor de backend antes de darlo por muesrto
        retries 3
        # permite que un cliente sea redirigido si tiene
persistencia en un servidor de backend que se cae
        option redispatch
        # numero maximo de conexiones en el balanceador
        maxconn 2000
        # timeouts en milisegundos, estas directivas en
versiones estï¿½n deprecated y se utilizand las nuevas que
empiezan por timeout
        # tiempo maximo para conectar a un servidor de
backend
        contimeout 10000
        # tiempo que esperamos a un cliente inactivo
        clitimeout 50000
        # tiempo que esperamos a un servidor inactivo
        srvtimeout 50000
# definimos el balanceador HTTP mediante un proxy definido
con listen
listen balanceador <%= node['lb']['balan'] %>:80
        # indica el algoritmo de balanceo utilizado,
rounrobin incluye peso
        balance roundrobin
        #option httpchk GET /ldirectord.html
```

```
        # server es una directiva compleja y admite
multitud de parametros como podemos ver en
http://cbonte.github.com/haproxy-dconv/configuration-
1.4.htm$
        # check provoca que los servidores sean comprobados
cada cierto tiempo para mantenerlos activos
        # inter indica el tiempo en milisengundos entre
chequeos
        # rise indica el numero de chequeos positivos
consecutivos necesarios para considerar el servidor online
        # fall indica el numero de chequeos negativos
consecutivos necesarios para considerar el servidor caido
        # weight indica el peso del servidor dentro del
conjunto
        server host1 <%= node['nodo1']['ip1'] %>:80 check
inter 2000 rise 2 fall 3 weight 50
        server host2 <%= node['nodo2']['ip2'] %>:80 check
inter 2000 rise 2 fall 3 weight 50
```

Ahora vamos a crear una receta simple que instalará apache en los nodos y sustituirá el archivo "/var/ww/index.html" por la plantilla que crearemos más adelante:

Este será el contenido de la receta que aplicaremos al NODO1 que será igual que la receta que se le aplicará al NODO2, solo se cambiará la plantilla que se le aplicará a cada nodo:

```
package "apache2" do
  action :install
end

template node['nodo1']['path'] + '/index.html' do
  source 'index1.html.erb'
  mode 0755
  owner 'root'
  group 'root'
  variables(
    :server        => node['nodo1']['server'])
end

service "apache2" do
  action :restart
end
```

Ahora vamos a ver el contenido de la plantilla que utilizaremos para sustituir el fichero "/var/www/index.html.erb".

- index1.html.erb:

```
<html>
<h1>
```

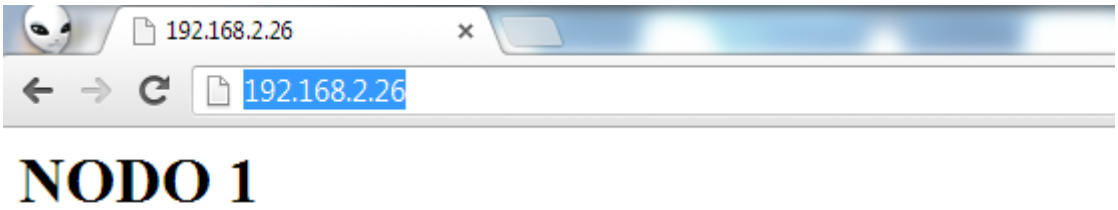
```
<%= node['nodo1']['server'] %>
</h1>
</html>
```

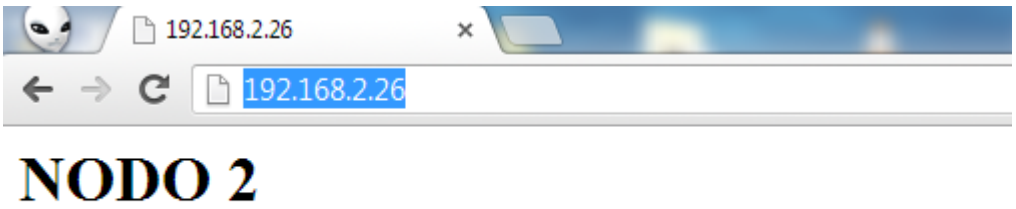
Hecho esto solo queda asignar cada receta a su nodo correspondiente que se haría con los siguientes comandos:

```
root@servidor-chef:~# knife node run_list add lb.example.com
'recipe[balanceador::default]'
run_list:  recipe[balanceador::default]

root@servidor-chef:~# knife node run_list add nodo1.example.com
'recipe[balanceador::nodo1]'
run_list:  recipe[balanceador::nodo1]

root@servidor-chef:~# knife node run_list add nodo2.example.com
'recipe[balanceador::nodo2]'
run_list:  recipe[balanceador::nodo2]
```

Hecho esto podemos esperar a que se ejecuten las recetas en los distintos nodos o podemos ejecutarlas con el comando chef-client en cada nodo:

Una vez terminada la ejecución en todos los nodos podemos comprobar el funcionamiento:





10. Bibliografía

- https://learnchef.opscode.com/
- http://docs.opscode.com/chef_overview.html
- http://wiki.opscode.com/display/chef/Installing+Chef+Server+on+Debian+or+Ubuntu+using+Packages
- http://wiki.opscode.com/display/chef/Installing+Chef+Client+on+Ubuntu+or+Debian