ALBERTO GORINES SÁNCHEZ

Monitorización de instancias y servicios de EC2

Alberto Gorines

24/07/2015

Implementación y documentación de una solución óptima en AWS EC2 para la monitorización de instancias en Linux según una investigación y comparativa deproductos y servicios en el mercado

Contenido

Objetivo del proyecto

- El objetivo del proyecto consiste en diseñar, implementar y documentar una solución de monitorización de instancias EC2 en Amazon Web Service (AWS en adelante).

- Definir un paquete de métricas y notificaciones básicas dentro de un modelo servidor-agentes, con una configuración mínima que sea fácil de implementar en nuevas instancias de manera automatizada.

- Mostrar datos recolectados en los agentes a través de una interfaz gráfica.

- Enviar notificaciones a través de correo electrónico.

Análisis de la solución

Para implementar la solución arriba descrita, se ha realizado un estudio comparativo de herramientas para despliegue de configuraciones de manera automatizada, así como de herramientas de monitorización. Aquí detallamos alguna de estas comparativas.

Herramientas de automatización:

- **Puppet Vs Chef**
    - Puppet es bastante más utilizado de Chef en el entorno profesional. Hablamos de más de 80 compañías, entre las que se incluyen Google, RedHat Siemens y muchas más, además de algunas de las más importantes universidades como puede ser Hardvard o Standford.
    - Puppet tiene gran cantidad de contribuciones de desarrolladores a su código y además tiene componentes de empresas externas específicos para él, como por ejemplo Foreman.
    - Puppet tiene mucho más recorrido que Chef y por tanto está mucho más maduro, sobre todo para despliegues en entornos de producción.
    - Puppet tiene una gran cantidad de documentación, mientras que a Chef todavía le falta mejorar bastante este aspecto.
    - Puppet soporta múltiples plataformas.
    - Chef está fuertemente inspirado en puppet.
    - Chef es parecido a Shell script, mientras que Puppet utiliza un lenguaje de tipo declarativo en el que el código puede ser reordenado sin afectar al órden de ejecución.
    - Puppet dispone de una comunidad muy colaborativa a la que poder recurrir en caso de necesitar ayuda.

- **Puppet Vs Ansible Vs Cloud Formation**

    En este caso no hemos visto una clara diferencia como con el caso de Chef.

    Con respecto a Cloud Formation, nos decantamos por Puppet, ya que esta herramienta no es sólo válida para la nube, si no que se puede aplicar en cualquier entorno.

    Con respecto a Ansible, parece que Puppet también es más maduro, y tiene más interoperabilidad y por comparativas como la de la siguiente tabla optamos por Puppet.

InfoWorld Scorecard	Scalability (20.0%)	Availability (20.0%)	Performance (10.0%)	Value (10.0%)	Management (20.0%)	Interoperability (20.0%)	Overall Score (100%)
AnsibleWorks Ansible 1.3	8.0	9.0	9.0	9.0	8.0	7.0	8.2
Enterprise Chef 11.4	9.0	9.0	8.0	9.0	7.0	8.0	8.3
Puppet Enterprise 3.0	9.0	9.0	9.0	9.0	9.0	9.0	9.0

http://www.infoworld.com/article/2609482/data-center/data-center-review-puppet-vs-chef-vs-ansible-vs-salt.html

Herramientas de monitorización:

- **Nagios Vs Zabbix**
  - Ambos pueden monitorizar los protocolos principales (HTTP, FTP, SSH, POP3, SMTP, SNMP, MySQL, etc)
  - Ambos tienen la capacidad de enviar notificaciones tanto por e-mail como por SMS.
  - Nagios muestra automáticamente un mapa de los hosts que y servicios que está monitorizando.
  - Nagios tiene múltiples niveles de aletar: OK, Warning, Critical
  - Nagios tiene detección de flapping.
  - Nagios necesita acceso SSH a las máquinas remotas a monitorizar o el componente NRPE.
  - Zabbix tiene agentes nativos multiplataforma.
  - Zabbix tiene dashboards personalizables.
  - Zabbix es más complejo de instalar.
  - La documentación de Zabbix no es demasiado clara.

Tras esta comparativa vemos que Zabbix tiene bastante parecido con Nagios, aunque probablemente éste último sea el más extendido de los dos.
También se ha evaluado la posibilidad de utilizar Icinga pero vemos que es un fork de Nagios, también bastante similar en cuanto a funcionalidad.
Para realizar el estudio no hemos apoyado en la siguiente tabla.
Tras analizar las distintas features de cada sistema, optamos por utilizar Nagios para el proyecto.
Para la representación gráfica de los datos utilizaremos el software pnp4Nagios, el cuál se integra con Nagios.

Software seleccionado:

Nagios:

Nagios es un sistema de monitorización de redes ampliamente utilizado, de código abierto, que vigila los equipos (hardware) y servicios (software) que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. Entre sus características principales figuran la monitorización de servicios de red (SMTP, POP3, HTTP, SNMP...), la monitorización de los recursos de sistemas hardware (carga del procesador, uso de los discos, memoria, estado de los puertos...), independencia de sistemas operativos, posibilidad de monitorización remota mediante túneles SSL cifrados o SSH, y la posibilidad de programar plugins específicos para nuevos sistemas.

Se trata de un software que proporciona una gran versatilidad para consultar prácticamente cualquier parámetro de interés de un sistema, y genera alertas, que pueden ser recibidas por los responsables correspondientes mediante (entre otros medios) correo electrónico y mensajes SMS, cuando estos parámetros exceden de los márgenes definidos por el administrador de red.

Llamado originalmente Netsaint, nombre que se debió cambiar por coincidencia con otra marca comercial, fue creado y es actualmente mantenido por Ethan Galstad, junto con un grupo de desarrolladores de software que mantienen también varios complementos.

Nagios fue originalmente diseñado para ser ejecutado en GNU/Linux, pero también se ejecuta bien en variantes de Unix.

Nagios está licenciado bajo la GNU General Public License Version 2 publicada por la Free Software Fundation.

Puppet:

Puppet es una herramienta de gestión de la configuración de código abierto. Diseñada para administrar la configuración de sistemas similares a Unix y a Microsoft Windows de forma declarativa. El usuario describe los recursos del sistema y sus estados utilizando el lenguaje declarativo que proporciona Puppet. Esta información es almacenada en archivos denominados manifiestos Puppet. Esta herramienta descubre la información del sistema a través de una utilidad llamada Facter, y compila los manifiestos en un catálogo específico del sistema que contiene los recursos y la dependencia de dichos recursos, estos catálogos son ejecutados en los sistemas de destino.

Está escrito en Ruby y fue liberado bajo la Licencia Pública General de GNU (GPL) hasta la versión 2.7.0 y después bajo la licencia Apache 2.0. Puppet Labs y Puppet fueron fundados por Luke Kanies en el 2005.

Software utilizado y componentes de AWS.

Software:

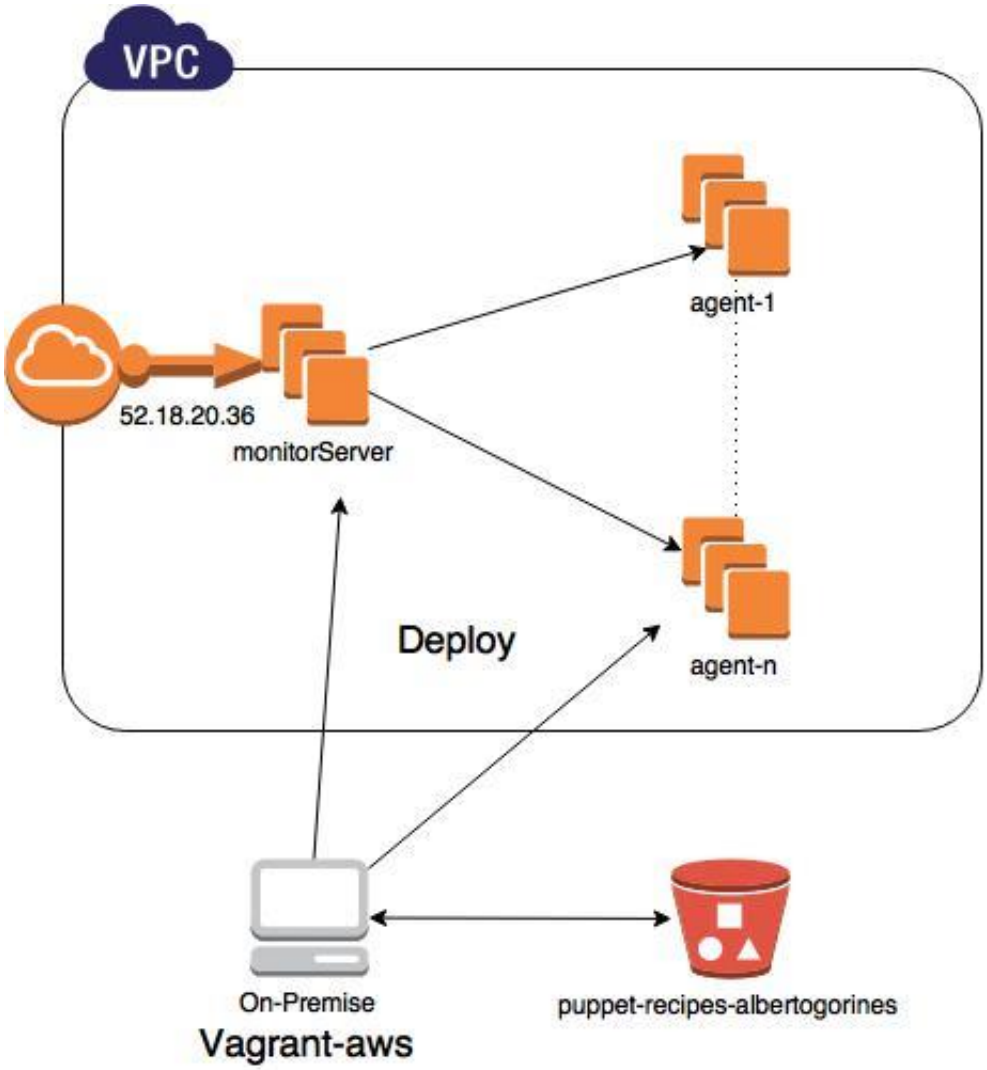
- **Sistema Operativo:** Ubuntu 14.04.
- **Git:** Utilizado en local para realizar el control de versiones tanto del código Puppet, como de las recetas de despliege.
- **Vagrant 1.7.2:** Utilizado para levantar la máquina virtual que descarga el código y recetas de Puppet y desde la cual se lanzan las instancias EC2.
- **Vagrant-aws:** Plugin para Vagrant necesario para la interacción con AWS.
- **Puppet 3.7.5:** Utilizado para la automatización de la configuración, tanto para la instancia manager, como para las instancias en AWS.
- **Shell Script:** Utilizado para preparar las instancias levantadas con Vagrant. Instalación de paquetes, actualización del Sistema Operativo.
- **S3cmd:** Sincronización de archivos entre las instancias y el bucket S3.
- **Apache2:** Utilizado para alojar el entorno web de Nagios.
- **Postfix:** Utilizado para el envío de notificaciones vía email desde Nagios.
- **Mailutils:** Necesario para que Nagios puede enviar emails con el plugin mailx.
- **Nagios4.0.8:** Utilizado para monitorizar las instancias agent. Instalado en monitorServer.
- **Nagios-nrpe-server:** Servicio que corre en las instancias agent para que éstas puedan ser monitorizdas.
- **Nagios-plugins:** Instala en las instancias agent diversos plugins para su monitorización.
- **Pnp4nagios:** Plugin utilizado para que el Nagios Server sea capaz de mostrar gráficas.
- **Check_nrpe:** Plugin necesario para que Nagios Server sea capaz de monitorizar instancias remotas.
- **Check_linux_stats:** Plugin para recolectar las métricas típicas en las instancias agent. CPU, Load Avg, Uso de disco, Memoria, etc.

Componentes AWS:

- **VPC:** Las instancias de este proyecto estarán dentro de una VPC en el rango 192.168.0.0/22
- **Subnet:** Se ha creado una subnet para todas las instancias con el rango 192.168.0.0/23
- **Elastic IP:** Se ha asignado una ElasticIP a la instancia del Nagios Server.
- **Internet Gateway:** Necesario para conexión a Internet del Nagios Server.

Diseño de la solución

En la siguiente figura podemos ver un esquema en el que se muestra de manera gráfica lo que describiremos a continuación.



Para desarrollar la solución descrita en el objetivo del proyecto son tomadas las siguientes decisiones:

On-Cloud

1) Se crea una VPC específica para las instancias del proyecto. El rango de ésta es 192.168.0.0/23.
2) Se crea una subred dentro de la VPC mencionado con el rango 192.168.0.0/23. Quedando otra subred /23, reservada para crecimiento.
3) Se crea un Internet Gateway y se asigna a esta VPC.
4) Se reserva una Elastic IP para asignarla a la instancia en la que corre el Nagios Server.

5) Las instancias en las que corren los agentes, se levantan en la misma subred que el Nagios Server. Podríamos haberlas creado en una subred privada complicando la estructura de red, pero ese no era el objeto de la práctica. Las IPs privadas de las mismas son fijadas en el Script de arranque de Vagrant, de modo que facilitamos la tarea de configuración de los agentes en el Nagios Server.

6) Se crea un security group que comparten todas las instancias en el que se habilita el tráfico ssh y http desde la IP del cliente on-premise, el tráfico TCP por el puerto 5666 para la subred 192.168.0.0/23. Este es el puerto utilizado por el Nagios Server utilizado para obtener los datos de los agentes.

7) Se crea un Bucket en S3 (puppet-recipes-albertogorines) para subir el código de Puppet y las recetas de despliegue. Este código se descargará en una máquina virtual On-premise, que veremos más adelante, desde la que se realizará el despliegue de la plataforma de monitorización.

8) Se crea una "ami" propia, a partir de una instancia limpia con Ubuntu14.04, que se utilizará para levantar todas las instancias utilizadas en el proyecto.

On-Premise

Utilizaremos Vagrant para levntar una máquina virtual on-premise. El utilizar una máquina virtual manager, no solo dota al proyecto de gran portabilidad, sino que, además evita "contaminar" el entorno de trabajo, ya que el único requisito será tener instalado Vagrant. Dicha máquina correrá el sistema operativo Ubuntu 14.04.

La configuración de ésta instancia se realiza de manera automatizada mediante Shell Script y Puppet. Todo esto está configurado en el VagrantFile de la instancia manager. (ver adjunto 1)

1) Mediante el script vagrant-setup.sh (ver adjunto 2) se instalarán los siguientes paquetes:

    a) Puppet

    b) Vagrant

    c) Vagrant-aws

    d) Ruby 1.9.1-dev

    e) Librerías zlib

2) Mediante la receta de Puppet env-manager.pp (ver adjunto 3) se instala el paquete S3cmd.

3) Mediante el script download-files-s3.sh (ver adjunto 4) se descargarán los archivos que contienen el código de Puppet y las recetas necearias para realizar los despliegues en AWS.

    a) Puppet/modules (Contiene todo el código de que ejecutarán las recetas de Puppet)

    b) Puppet/manifests (Contiene las recetas que se ejecutarán cada una en su correspondiente máquina)

c) Deployment-recipe/Vagrantfile. (Dado que el método seleccionado para levantar las instancias en AWS ha sido la integración de Vagrant y Puppet, el Vagrantfile en este caso contendrá todos los datos de las instancias: Número de instancias, ami, región, type_instance, credenciales para el API, etc. (ver adjunto 5).)

d) Deployment-recipe/Vagrant-setup. (En este caso este script solo hace un "apt-get update" e instala Puppet en la máquina destino.

Todos estos paquetes se descargarán en el directorio /home/vagrant/nagios-deploy, creado previamente por el script.

4) Llegados a este punto, ya tenemos todo listo en la máquina virtual para poder realizar el despliegue de nuestro entorno en AWS.

Configuración de los paquetes utilizados

Aunque toda la configuración está automatizada con Puppet y se podría desplegar la plataforma sin conocer los detalles, en este punto explicaremos como se han configurado los paquetes más relevantes.

Vagrant-aws:

En realidad vagrant-aws es un plugin que posibilita que en el Vagrantfile podamos configurar como proveedor AWS para así poder levantar instancias con ciertos parámetros configurados. Vamos a detallar lo configurado en el Vagrantfile del proyecto:

node.vm.provider :aws do |aws, override| Nodo se levantará en AWS

aws.access_key_id = "AKIAJXHPG7OIQ3XXXXXX" Access key de nuestra cuenta

aws.secret_access_key = "hq9HLMuWtjGbBaAg9F7Lelh3X+h60S0Z8YXXXXXX" Secret key de nuestra cuenta

aws.keypair_name = "alberto.gorines" Nombre de nuestro keypair

aws.ami = "ami-8da6d9fa" ami utilizada para levantar la instancia

aws.region = "eu-west-1" Región

aws.availability_zone = "eu-west-1a" Zona de disponibilidad

aws.instance_type = "m3.medium" Tipo de máquina

aws.security_groups = "sg-9fb8bafa" Security group para la instancia

aws.elastic_ip = "52.18.20.36" Elastic IP

aws.private_ip_address = "192.168.0.11" Private IP

aws.subnet_id = "subnet-0349f75a" Subred en la que levantará la instancia

aws.associate_public_ip = "true" Si queremos que se le asocie IP pública o no

aws.tags = { Tags varias

'Name' => HOSTNAME_SERVER

}

override.vm.box = "https://github.com/mitchellh/vagrant-aws/raw/master/dummy.box"

override.ssh.username = "ubuntu"

override.ssh.private_key_path = "albertogorines.pem"

end

Nagios-4.0.8:

1) Creamos el usuario y grupos correspondientes para Nagios.
   a) Usuario nagios y grupo nagcmd.
2) Descargamos Nagios.
   a) wget http://sourceforge.net/projects/nagios/files/nagios-4.x/nagios-4.0.8/nagios-4.0.8.tar.gz/download
3) Compilamos ejecutando lo siguiente una vez descomprimido el fichero descargado.
   a) configure --with-command-group=nagcmd
   b) make all
   c) make install
   d) make install-init
   e) make install-config
   f) make install-commandmode
4) Habilitamos el sites para Nagios en Apache:
   a) install -c -m 644 sample-config/httpd.conf /etc/apache2/sites-enabled/nagios.conf
5) Creamos el usuario nagiosadmin para entrar en la web de Nagios.
6) Instalamos el paquete nagios-plugins. (En realidad este paso se puede omitir si se tienen claro los plugins a utilizar)
7) Habilitar el módulo cgi y rewrite de Apache.
8) Reiniciar el servicio Apache.

9) En el archivo de configuración /usr/local/nagios/etc/nagios.cfg tenemos que descomentar la línea: cfg_dir=/usr/local/nagios/etc/servers

10) Crear el directorio servers en el path del punto 9.

11) En el directorio servers, debemos crear el archivo o los archivos de definición de hosts y servicios. Podemos definir todo en el mismo archivo o crear archivos diferentes por host y servicio como es el caso del proyecto. (Ver adjunto 6)

12) **Importante**, todos los comandos definidos en el punto 11 tienen que estar definidos en el fichero /usr/local/nagios/etc/objects/commands.cfg.
Así como los contactos, deben estar definido en el fichero /usr/local/nagios/etc/objects/contacts.cfg. (ver adjunto 7)

13) Si utilizamos algún plugin específico para algún comando deberemos especificar la ruta en la definición del comando. En nuestro caso, hemos tenido que definir el comando **check_nrpe,** utilizado para poder monitorizar instancias remotas.

14) Reiniciar servicio nagios.

Nagios-nrpe-server:

Este servicio se instala en las instancias a monitorizar. Para su correcto funcionamiento seguiremos los siguientes pasos:

1) Añadir en el fichero /etc/nagios/nrpe.cfg el parámetro la IP del servidor desde el que monitorizaremos la instancia allowed_hosts. Para permitir que en los comandos se pasen argumentos desde el servidor, tendremos que poner a 1 el parámetro dont_blame_nrpe.

2) Añadir en el fichero /etc/nagios/nrpe.cfg todos los comando que queremos que el servidor sea capaz de monitorizar. En nuestro caso serán de este estilo:
command[check_mem]=/usr/lib/nagios/plugins/check_linux_stats.pl -$ARG1 -w $ARG2 -c $ARG2

3) Todos los plugins que utilicemos aquí tienen que estar presentes en el directorio indicado. En el caso del punto 2, tenemos que tener el plugin check_linux_stats.pl en el directorio /usr/lib/nagios/plugins

4) Instalar el paquete libsys-statistics-linux-perl para que la instancia pueda sacar estadísticas. Sin este paquete instalado en las instancias el servidor no podrá obtener datos.

5) Reiniciar el servicio nagios-nrpe-server.

Pnp4nagios:

Este es el pluging utilizado para que Nagios sea capaz de mostrar gráficas. Para su instalación y configuración hay que realizar los siguientes pasos:

1) Descargar pnp4nagios:
    a) wget http://sourceforge.net/projects/pnp4nagios/files/PNP-0.6/pnp4nagios-0.6.25.tar.gz/download
2) Instalar los paquetes rrdtool y php5-gd
3) Compilamos ejecutando lo siguiente una vez descomprimido el fichero descargado.
    a) configure --prefix=/usr/local/nagios/pnp4nagios
    b) make all
    c) make fullinstall
4) Mover el archivo status-header.ssi al directorio /usr/local/nagios/share/ssi/
5) Añadir al fichero nagios.conf alojado en Apache, el contenido de sample-config/httpd.conf del directorio descomprimido.
6) Añadir en el template de service la siguiente línea: action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=$SERVICEDESC$' class='tips' rel='/pnp4nagios/index.php/popup?host=$HOSTNAME$&srv=$SERVICEDESC$

    (ojo, es una sola línea)
7) Añadir en el template de hosts la siguiente línea: action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=_HOST_' class='tips' rel='/pnp4nagios/index.php/popup?host=$HOSTNAME$&srv=_HOST_

    (ojo, es una sola línea)
8) Renombrar el archivo /usr/local/nagios/pnp4nagios/share/install.php a install-orig.php
9) Reiniciar el servicio apache y el servicio Nagios.

Postfix:

Este es el paquete que utilizamos para que Nagios pueda enviar notificaciones por correo electrónico. Se ha valorado la posibilidad de utilizar el servicio SES de AWS o directamente el smtp de Gmail. Finalmente optamos por utilizar el servidor de Gmail con un email de pruebas. Para su instalación y configuración hay que realizar los siguientes pasos:

1) Instalar el paquete postfix desde los repositorios de Ubuntu.
2) Configurar en el fichero /etc/postfix/main.cf el servidor smtp desde el que queremos realizar el envío. En nuestro caso la configuración del fichero es:

relayhost = [smtp.gmail.com]:25

smtp_sasl_auth_enable = yes

smtp_sasl_security_options = noanonymous

smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd

smtp_use_tls = yes

smtp_tls_security_level = encrypt

smtp_tls_note_starttls_offer = yes

3) Configurar nuestro usuario y password en para el servidor smtp anterior en el fichero /etc/postfix/sasl_passwd
   [smtp.gmail.com]:25 usuario:password
4) En los comando del Nagios server relativos a notificaciones tenemos que cambiar mail por mailx.

   Los comando son: Notify-host-by-email y notify-service-by-email.
5) Reiniciar servicio postfix

Documentación de despliegue

Comandos Vagrant más relevantes:

Vagrant up [máquina] : Levanta las máquinas configuradas en el Vagrantfile o la especificada.

Vagrant provision [máquina] : Provisiona una máquina que ya esté levantada. Si sólo hay una no hace falta especificar el nombre.

Vagrant halt [máquina]: Apaga todas las máquinas levantadas o la especificada.

Vagrant status: Devuelve el estado en el que están las máquinas.

Vagrant destroy [máquina]: Destruye todas las máquinas existentes o la especificada.

A continuación mostraremos una pequeña guía a través la cual, cualquier persona podría desplegar una plataforma de monitorización en AWS de manera sencilla y automatizada.

1) Instalar Vagrant en la máquina cliente. Aunque Vagrant puede utilizarse con varios sistemas de virtualización, nosotros utilizaremos VirutalBox ya que es gratuito y funciona con Vagrant sin necesidad de configuraciones adicionales.

2) Descargar desde el Bucket de S3 puppet-recipes-albertogorines el directorio manager-instance-local. Lógicamente para realizar esta descarga hay que tener acceso al Bucket. Para esto podríamos hacer pública dicho directorio.

3) Entrando en la el directorio descargado en el paso 2, solamente tendríamos que ejecutar “vagrant up”. Con esto aparecerán en pantalla un serie de línea que indicarán lo que se está configurando en la instancia, y tendremos que esperar hasta que veamos la línea “Notice: Finished catalog run in X0.XX seconds”. Hay que tener en cuenta que la primera vez que se ejecuta este paso tarda unos minutos ya que se tiene que descargar todos los ficheros de S3.

4) Desde este mismo directorio, podremos ejecutar varios comando Vagrant, para comprobar el estado de la máquina, apagarla, reiniciarla, destruirla o conectarnos a ella. Ejecutaremos “vagrant ssh” para conectarnos a ella.

5) Una vez dentro de la máquina tendremos que ir al al directorio ~/nagios-deploy/deployment-recipe, que es dónde se han descargado los archivos de Vagrant para realizar el despliegue en AWS, tal y como comentamos en el punto 3 del apartado anterior.

6) En este caso, al ejecutar el comando “vagrant up”, se levantarán una instancia monitorServer y 2 instancias agente client-1, client-2. El número de instancias agente se puede variar cambiando el valor de la variable NUM_CLIENTS en el Vagrantfile. Si quisiéramos actuar sobre una instancia en concreto, por ejemplo conectarnos a monitorServer, ejecutaríamos “vagrant ssh monitorServer”.

7) Con esto, ya tendríamos levantada y configurada la plataforma de despliegue y podríamos conectarnos a la “ElasticIP”/nagios.
   User: nagiosadmin
   Password: nagios

Analisis de Costes

Realmente en este proyecto no hemos visto necesario realizar el análisis ya que no utilizamos un uso intensivo de de ninguno de los componentes de AWS. Los discos de las máquinas son magnéticos.

El tipo de máquina seleccionado ha sido medium para todas, aunque en una situación real, el tipo de los agentes dependerá de los servicios que vayan a correr.

El consumo de S3 será muy escaso, ya que una vez desplegado. Los archivos no deberían sufrir demasiadas modificaciones.

Adjuntos

Todos estos adjunto están disponibles en el zip “Archivos proyecto Monitorización - Alberto Gorines” o en el Bucket de S3 puppet-recipes-albertogorines.

Dar explicación de cada script si procede.

Adjunto 1

/vagrantMaster/manager-instance-local/Vagrantfile

Este archivo es el que aloja la configuración para que al hacer vagrant up, la instancia se configure debidamente.

Adjunto 2

/vagrantMaster/manager-instance-local /vagrant-setup.sh

Este script es ejecutado desde el Vagrantfile y realiza las siguientes acciones:

- Descarga de actualizaciones del Sistema Operativo.
- Instalación de Vagrant.
- Instalación de vagrant-aws.
- Instalación de Puppet.

Adjunto 3

/puppet/manifests /env-manager.pp

Esta es la receta de Puppet utilizada para instalar y configurar el software necesario en la instancia manager.

- Instalación de S3cmd.

Adjunto 4

vagrantMaster/manager-instance-local /download-files-s3.sh

Este script es ejecutado desde el Vagrantfile y realiza las siguientes acciones:

- Crea el directorio /home/vagrant/nagios-deploy en la instancia.
- Descara los directorios puppet y deployment-recipe desde el Bucket de S3 puppet-recipes-albertogorines

  La descarga se realiza con S3cmd sync, por lo que la primera vez se descargará todo el contenido y las siguientes solamente se descargarán las moficaciones.

Adjunto 5

vagrantMaster/deployment-recipe/Vagrantfile

En este caso el Vagrantfile tiene toda la configuración necesaria para levantar las instancias indicadas en AWS.

En nuestro proyecto está configurado para levantar las siguientes instancias:

- monitorServerV2: esta instancia alojará el servidor de Nagios.
- clientv3-1: será uno de los agentes que monitorizará el Nagios server.
- clientv3-2: el segundo agente a monitorizar.

El número de agentes a monitorizar puede variarse modificando el valor de la variable NUM_CLIENTS.

Adjunto 6

Aquí mostramos un ejemplo de configuración de host en Nagios y otro ejemplo de configuración de services.

Host

```
define host {
   use                     linux-server
   host_name               clientv3-1
   address                 192.168.1.11
   max_check_attempts      3
   alias                   monitorServerV2.eu-west-1.compute.internal
   contact_groups          nagios_notifications
}
```

Services

```
define service {
   use                     generic-service
   service_description     check_cpu_c1
   host_name               clientv3-1
   contact_groups          nagios_notifications
   check_command           check_nrpe!check_cpu
   max_check_attempts      3
}
```

Adjunto 7

Aquí mostramos un ejemplo de configuración de contacts en Nagios.

```
define contact{
 contact_name                    alberto
 service_notification_period     24x7
 host_notification_period        24x7
 service_notification_options    w,u,c,r,f
 host_notification_options       d,u,r,f
 service_notification_commands   notify-service-by-email
 host_notification_commands      notify-host-by-email
 email                           algorines@gmail.com
}
define contactgroup {
    contactgroup_name    nagios_notifications
    alias                grupo para nagios
    members              alberto
}
```