Introducción a ansible

Ansible es un software de gestión de la configuración automática. Nos permite centralizar la configuración de numerosos servidores de una forma sencilla y automática.

¿Cómo funciona Ansible?

- Ansible configura las máquinas clientes desde otra máquina donde tenemos instalado el software de Ansible.
- Ansible accede a los clientes por SSH para realizar las diferentes tareas de configuración.
- Ansible funciona habitualmente en modo “push”, aunque puede configurarse en modo “pull”.

Instalación de ansible

Lo más sencillo es utilizar la versión empaquetada en tu sistema:

```
# apt-get install ansible
```

Aunque si quieres la última versión puedes obtenerla de pypi (pip install ansible)

Configuración de ansible

En el fichero *hosts* se configuran los clientes con los que se van a trabajar, pudiéndose agrupar:

```
[servidores]
servidor1 ansible_ssh_host=212.231.128.33 ansible_ssh_user=debian
```

Hemos indicado la IP del cliente, y el usuario que se va a utilizar para el acceso SSH.

En el fichero *ansible.cfg* se indica la configuración de ansible, por ejemplo:

«««< HEAD [defaults] hostfile = hosts remote_user = debian sudo = yes private_key_file = ~/.ssh/clave-ow ======= [defaults] hostfile = hosts remote_user = debian private_key_file = ~/.ssh/clave-ow »»»> 4e9bbc6839fcc220c5e45d5034d877a2b6e9f94d

Se indica el fichero con el catálogo de clientes, el usuario remoto a utilizar y la cavle pública que se va a autilizar para el acceso SSH.

Ejecutando comandos sencillos

El primer comando que vamos a usar, nos va a permitir comprobar que tenemos acceso a los clientes:

```
ansible -m ping all
```

Otro ejemplo, para ejecutar un comando en el cliente:

```
ansible -m shell -a 'free -m' servidor1
```

Creando recetas (playbooks)

Un playbook es un fichero de texto escrito en YAML donde indicamos los comandos que queremos ejecutar en los clientes. Por ejemplo:

```
---
- hosts: servidores
  tasks:
    - name: Installs nginx web server
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - start nginx

  handlers:
    - name: start nginx
      service: name=nginx state=started
```

La tareas a realizar (tasks) se van a llevar a cabo en todos los clientes del grupo “servidores”. En este caso se va a instalar el paquete nginx asegurándonos que la lista de paquetes está actualizada (update_cache=true). Sólo si se ha realizado esta tarea

(es decir, se ha instalado el paquete) se llama a un “manejador” (handlers) que reinicia el servicio.

Para ejecutar el playbook, suponiendo que lo hemos guardado en un fichero playbook.yml:

```
ansible-playbook playbook.yml
```

Que utilizará los parámetros del fichero ansible.cfg

Añadiendo una página index por defecto

Vamos a ampliar la funcionalidad de nuestro playbook, copiando en el cliente una página index.html:

Para ello creamos un directorio files, y dentro un fichero index.html:

```
<html>
  <head>
    <title>This is a sample page</title>
  </head>
  <body>
    <h1>Here is a heading!</h1>
      <p>Here is a regular paragraph.  Wow!</p>
  </body>
</html>
```

Añadimos una tarea nueva a nuestro playbook:

```
---
- hosts: servidores
  tasks:
    - name: Installs nginx web server
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - start nginx

    - name: Upload default index.html for host
      copy: src=files/index.html dest=/var/www/html/ mode=0644 owner=www-data group=www-data

  handlers:
    - name: start nginx
      service: name=nginx state=started
```

Una de las características de ansible es la idempotencia, es decir, la segunda vez que ejecutamos la receta, si el estado que consigue la ejecución de una tarea ya lo tenemos, dicha tarea no se ejecuta. Por lo tanto, si nginx está instalado en el cliente la primera tarea no se vuelve a realizar.