Ansible

Ansible es una herramienta de automatización de infraestructura de código abierto. Está diseñada para ayudar a los administradores de TI y a los desarrolladores a automatizar el despliegue, la configuración y la gestión de la infraestructura de la computadora.

Ansible se puede usar para administrar sistemas de servidor Linux, Windows, máquinas virtuales, sistemas de almacenamiento y más.

Utiliza un lenguaje de configuración sencillo basado en **YAML** para describir los objetivos de configuración y administración de la infraestructura.

Ansible también cuenta con una gran biblioteca de **módulos** que se pueden usar para automatizar tareas comunes.

Funcionamiento

El funcionamiento es el siguiente:

- Ansible conecta con los diferentes nodos
- Les envia pequeños programas llamados módulos
- Estos módulos se ejecutan sobre SSH
- Los módulos definen estados en los que queremos encontrar el sistema

SSH

A la hora de conectar con los nodos para hacer cosas tenemos varias posibilidades:

- Se pueden utilizar passwords
- Se recomienda utilizar claves SSH

Formas de configuracion

- Mediante un unico archivo playbook que contiene las tareas
- Mediante una estructura de directorios por cada proyecto

Modo ad-hoc

Este modo permite ejecutar directamente comandos en una sola línea

Se utiliza cuando queremos realizar una acción simple como:

- Reiniciar un host
- Verificar conectividad

Si queremos realizar configuraciones complejas, son más útiles los playbooks.

Comandos

Podemos hace rping a todos los hosts:

```
ansible all -m ping
```

Pedir que un servidor esté instalado:

```
ansible maquina -m yum -a "name=httpd state=installed"
```

Reiniciar una máquina:

```
ansible maquina -m -a "/usr/sbin/reboot"
```

Playbooks

- Ansible trabaja con playbooks
- Son ficheros de texto plano escritos en yaml
- Describen operaciones a realizar sobre los nodos administrados
- Nos lo podemos encontrar en dos formas:
    - En un solo archivo
    - En varios archivos siguiendo un modelo estructurado

Estructura de un playbook

- Un playbook contiene una lista de plays
- Un play contiene una lista de tasks
- Cada task contiene una lista de módulos

Cuando ejecutamos un playbook, los módulos se ejecutan sobre los hosts remotos

Módulos

- Los módulos son trozos de código que se ejecutan cuando ejecutamos un playbook
- Permiten realizar tareas de sistema
    - Manejar servicios
    - Manejar paquetes
    - Crear y modificar archivos
    - Ejecutar comandos
- Existen módulos ya creados (built-in)
- Se pueden crear manualmente

Facts

Se trata de información sobre el sistema que se está aprovisionando que Ansible recoge antes de ejecutar determinadas tareas.

Por ejemplo:

- Cantidad de cores CPU
- Redes ipv4 y ipv6
- Discos montados
- Distribución de linux

Handlers

- Son tareas que se ejecutan al dispararse un evento concreto
- Se ejecutan siempre al final de un play
- Solo se ejecutan una vez que han terminado de ejecutarse todas las tareas
- Solo se ejecutan una vez, independientemente de cuantas veces se disparen
- Se ejecutan en el orden que aparecen, no en el orden que se disparan

Ejemplo:

Definición de 2 handlers:

```
handlers:
  - name: restart memcached
    service:
      name: memcached
      state: restarted
    listen: "restart web services" ## group handlers in one call
  - name: restart apache
    service:
      name: apache
      state: restarted
    listen: "restart web services"
```

Disparar handlers individualmente

- En este caso todas las acciones se van a realizar sobre el cluste webservers
- Una vez se ejecuta la tarea, hacemos 2 notify
- Un notify con el `name` de cada handler que queremos disparar

```
---
- hosts: webservers
  remote_user: root
  - name: template configuration file
    template:
      src: template.j2
      dest: /etc/foo.conf
    notify:
       - restart memcached
       - restart apache
```

Disparar handlers con una sola llamada

```
tasks:
    - name: restart everything
      command: echo "this task will restart the web services"
      notify: "restart web services"
```

Trabajo con playbooks

Ejecutar un playbook:

```
ansible-playbook playbook.yaml
```

Por defecto se ejecutará sobre el fichero hosts predeterminado.

Para ejecutar sobre otro inventario:

```
ansible-playbook -i produccion deploy_apache.yaml
```

Si necesitamos solicitar contrasena de root:

```
ansible-playbook playbook.yaml -k
```

Ejecucion de comandos

Existe un modulo `shell` para ejecutar comandos de la shell directamente.

Actualizacioon del sistema

Para tareas de mantenimiento de paquetes, tenemos el modulo `apt`.

Podemos forzar un upgrade.

Ejemplo:

```
---
-   hosts: 'clients'
    tasks:
    -   name: 'ejemplo'
        become: true
        apt:
            update-cache: true
            upgrade: true
```

Ejecutamos:

```
ansible-playbook playbook.yaml -k
```

Instalar o desinstalar paquetes

Para desinstalar: absent

```
---
-   hosts: 'clients'
    tasks:
    -   name: 'ejemplo'
        become: true
        apt:
            name: 'aptitude'
            state: 'present'
```

Inventario

Podemos definir que nodos queremos que gestione Ansible

Los módulos se pueden agrupar en grupos

La lista de hosts representa el inventario Podemos asignar variables por host o por grupos Para ello, deberemos crear un archivo `hosts` que incluira la lista de nodos y sus direcciones

Ejemplo:

```
[webserver]
10.0.0.2    ansible_ssh_user=user   ansible_ssh_private_key_file=
[dbserver]
10.0.0.3
```

Imaginemos que queremos ejecutar el playbook siguiente:

```
- hosts: all
  roles:
    - webserver
```

Roles

La idea es incluir archivos y combinarlos para crear abstracciones limpias y reusables Es necesario crear una estructura de carpetas y subcarpetas Las carpetas se pueden crear de forma manual o a partir de ansible-galaxy. Ansible-galaxy es un sitio para buscar, reutilizar e intercambiar roles desarrollados por la comunidad

Creando roles

Podemos utilizar ansible-galaxy:

```
ansible-galaxy init webservers
```

Una vez ejecutado, se nos creará una estructura de carpetas y archivos

Reutilización de componentes

Dos modos para reutilizar contenido: dinámico y estático

- Import
  - Modo estático
  - Import_playbook, import_task
  - Los imports son pre procesados en el tiempo en que se parsean los playbooks
- Include
  - Modo dinámico
  - Los includes se procesan a medida que se encuentran en tiempo de ejecución del playbook