Ejemplo: Automatización con ANSIBLE

Entorno de pruebas

Script de instalación GNU/Linux: ejercicio-ansible.sh

- Puesta en marca

```
bash ejercicio-ansible.sh
```

Script de instalación Windows: ejercicio-ansible.ps1 [aportado por Adrián Otero]

- Puesta en marca

```
Powershell.exe -executionpolicy bypass -file ejercicio-ansible.ps1
```

Usuarios disponibles

Login	Password	
root	purple	
usuario	purple	con acceso sudo

Recursos

- Web de ansible: https://www.ansible.com/
- Documentación de los módulos: Módulos por defecto
- Repositorio de *roles*: Ansible Galaxy
- Funcionamiento: How Ansible works

Funcionamiento

Ansible es una herramienta de automatización de sistemas

- de tipo procedimental: en la configuración (*paybooks*) se indican las acciones/tareas a realizar sobre los distintos hosts administrados
- con una estrategía *push*: el "nodo de control" aloja la configuración global (*playbooks*, módulos, etc) y ejecuta las tareas de configuración conectándose a los hosts administrados
- sin agentes (*agenteless*): no requiere instalar software específico en los hosts adminstrados, únicamente es necesario disponer de un servidor *ssh* al que conectarse y una instalación básica de Python (puede ser conveniente disponer de herramientas locales como `facter` pero no es imprescindible)

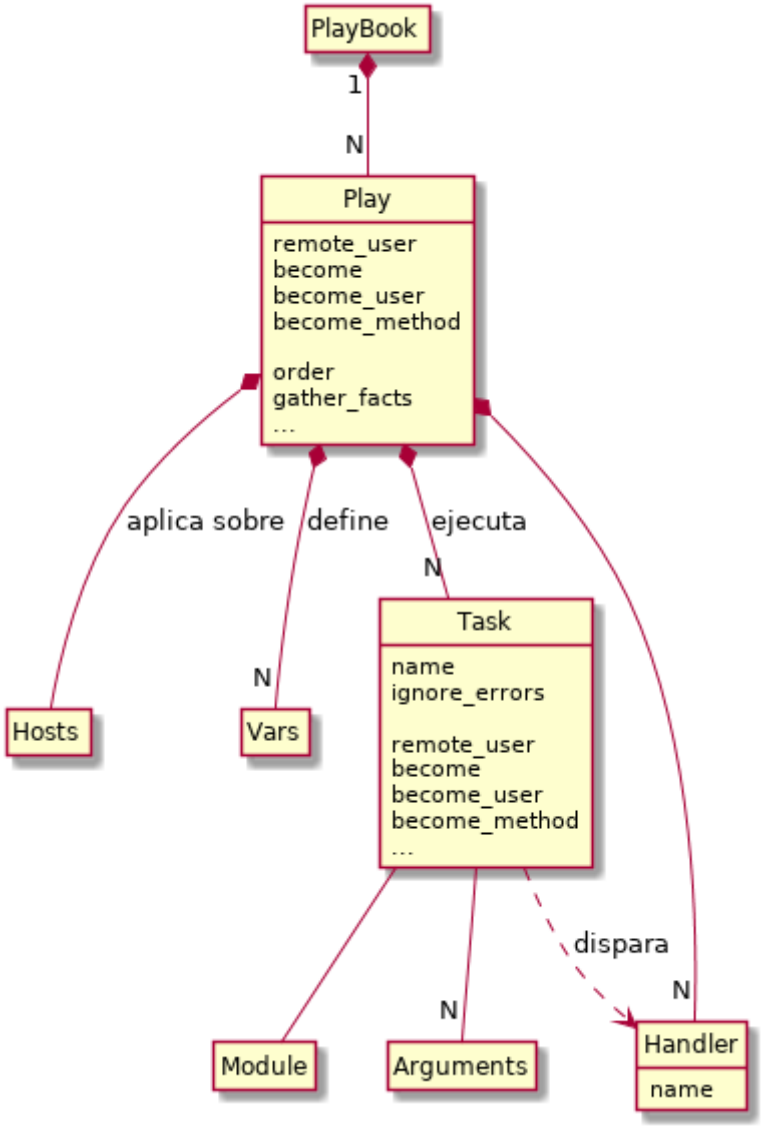
El "nodo de control" Ansible es el único participante donde debe estar instalado *Ansible* (paquete `ansible` en Debian/Ubuntu) junto con la colección de módulos.

- Desde el "nodo central" Ansible se conecta a los nodos administrados mediante SSH con un usuario existente en el host
    - Habilitualmente esa conexión SSH se configura para realizar autenticación via clave pública, evitando tener que introducir y gestionar passwords
- Ejecuta sobre cada nodo las tareas correspondientes (instalar componentes, modificar ficheros, crear usuarios, etc) según indique el *playbook*
    - De ser necesario, puede configurarse Ansible (*become*) para realizar esas tareas con privilegios de administrador mediante `sudo`, `su` o comandos similares
- Esas tareas las implementan los *módulos* Ansible interactuando con el hosts destino mediante la conexión SSH

Elementos

- **Módulos** Implementan las tareas a realizar sobre los hosts administrados a través de la conexión SSH
    - Pueden estar escritos en cualquier lenguaje (habitualmente python)
    - La distribución de Ansible incluye una collección de Módulos por defecto (en Debian ubicados en `/usr/lib/python3/dist-packages/ansible/modules/`)
- **Inventario** Listado de hosts sobre los que se realizan las tareas automatizadas con Ansible
    - Pueden usarse dos formatos: ficheros properties (INI), ficheros YAML
    - Es posible agrupar los hosts y vincularse variables propias
- **Playbooks** Colección de **Plays**, cada una de estas *Plays* indican los hosts del *Inventario* sobre los que realizar una lista de tareas (**Task**)
    - Cada *Task* se corresponde con la ejecución de un *Módulo* determinado (identificado por su nombre) con sus correspondientes argumentos específicos
    - Las *Task* son idempotentes, se garantiza que ejecuciones sucesivas dejarán el host en el mismo estado
    - Los *Playbooks* se definen en documentos YAML (*Ain't Markup Language _/*Yet Another Markup Language_) [(Sintaxis YAML)[https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html#yaml-syntax]]
- **Roles** Son una abstracción de los *Plays* que permiten reutilizarlos.
    - *Ansible Galaxy* es un repositorio de *Roles* open source del que se pueden descargar y utilizar en *Playbooks* locales

Estructura de los Playbooks (sin roles)



Nota: Listado de elementos de configuración en Playbooks

Comandos

- `ansible` Ejecución de tareas aisladas
- `ansible-playbook` Ejecución de *playbooks*
- `ansible-galaxy` Instalación (`ansible-galaxy install <nombre>`) y gestión de *roles* (por defecto se almacenan en `$HOME/.ansible/roles`)

Configuración

La configuración global de Ansible está en `/etc/ansible`

- `ansible.cfg` Configuración por defecto
- `hosts` Inventario de hosts por defecto
- `roles` Directorio con los roles disponibles (también `/usr/share/ansible/roles`)

En el directorio `$HOME/.ansible` está la configuración de usuario (con los correspondientes `ansible.cfg`, `hosts`, `roles`, etc)

Es posible "sobreescribir" la configuración por defecto (global o de usuario) mediante ficheros `ansible.cfg` y `hosts` en el directorio de trabajo.

- Tambien puede usarse un subdirectorio `roles` local para los *roles* particulares usados en cada caso.

Nota: Detalles sobre configuración

Uso básico de Ansible

(a) Desde la máquina **ansible** (logueado como `usuario/purple`), crear el directorio de trabajo `ejemplo_ansible` y fichero con el inventario de los equipos.

```
usuario@ansible:~/$ mkdir ejemplo_ansible
usuario@ansible:~/$ cd ejemplo_ansible
usuario@ansible:~/ejemplo_ansible$ nano inventario
```

```
[web]
web1.cda.net
web2.cda.net
web3.cda.net
web4.cda.net

[haproxy]
haproxy.cda.net

[bd]
bd.cda.net
```

Nota En lugar de enumerar los hosts `web1` `web2` `web3` y `web4` , otra alternativa al ternativa sería usar la notación `web[1:4].cda.net`

(b) Crear par de claves RSA para SSH (con `passphrase` vacía) y propogarlas a las máquinas del ejercicio (pedirá la contraseña de `usuario` [`purple`]).

```
usuario@ansible:~/ejemplo_ansible$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
9a:a0:d0:cc:b9:78:a8:24:15:ce:58:49:0f:5c:c8:33 usuario@ansible
The key's randomart image is:
+---[RSA 2048]----+
| ooo.            |
| .E+             |
|  +o.            |
| O o             |
|o O .   S        |
| = o . o         |
|+.+   o          |
|+.               |
|.                |
+-----------------+
```

```
usuario@ansible:~/ejemplo_ansible$ ssh-copy-id usuario@web1.cda.net
usuario@ansible:~/ejemplo_ansible$ ssh-copy-id usuario@web2.cda.net
usuario@ansible:~/ejemplo_ansible$ ssh-copy-id usuario@web3.cda.net
usuario@ansible:~/ejemplo_ansible$ ssh-copy-id usuario@web4.cda.net
usuario@ansible:~/ejemplo_ansible$ ssh-copy-id usuario@haproxy.cda.net
usuario@ansible:~/ejemplo_ansible$ ssh-copy-id usuario@bd.cda.net
```

(c) Comprobar la accesibilidad al usuario1 de esas máquinas con el módulo **ping** de Ansible.

```
usuario@ansible:~/ejemplo_ansible$ ansible all -i inventario -u usuario -m ping
haproxy.cda.net | success >> {
    "changed": false,
    "ping": "pong"
}

bd.cda.net | success >> {
    "changed": false,
    "ping": "pong"
}

web1.cda.net | success >> {
    "changed": false,
    "ping": "pong"
}

web3.cda.net | success >> {
    "changed": false,
    "ping": "pong"
}

web4.cda.net | success >> {
    "changed": false,
    "ping": "pong"
}

web2.cda.net | success >> {
    "changed": false,
    "ping": "pong"
}
```

(d) Para evitar indicar el inventario y el usuario en cada invocación, declarar esos datos en el fichero `ansible.cfg`.

```
usuario@ansible:~/ejemplo_ansible$ nano ansible.cfg
```

```
[defaults]
inventory = inventario
remote_user = usuario
```

(e) Ejecución de comandos en las máquinas administradas (módulo **command**)

```
usuario@ansible:~/ejemplo_ansible$ ansible all -m command -a "/usr/bin/free"
usuario@ansible:~/ejemplo_ansible$ ansible all -m command -a "/bin/df"
```

(f) Consulta de información en las máquinas administradas (serán las variables disponibles en los *playbooks*) (módulo **setup**)

```
usuario@ansible:~/ejemplo_ansible$ ansible all -m setup
```

- **Nota:** parte de la información extraida require tener instalada la herramienta `facter` en las máquinas administradas.

Playbooks sencillos: instalación de Apache2

(a) Crear el fichero `playbook_apache.yml` y lanzarlos sobre los servidores web con el comando `ansible-playbook` (pedirá la contraseña de *usuario*: `purple`).

```
usuario@ansible:~/ejemplo_ansible$ nano playbook_apache.yml
```

```
---
- name: instalacion de apache2 y php
  hosts: web
  become: yes
  tasks:
    - name: instalar apache2
      apt: name=apache2 update_cache=yes state=latest
```

```
usuario@ansible:~/ejemplo_ansible$ ansible-playbook playbook_apache.yml --ask-become-pass
```

- Pedirá la contraseña de *usuario* (`purple`) para que mediante *sudo* el módulo **apt** instale el paquete *apache2*.

(b) Crear los directorios `plantillas` y `ficheros`.

```
usuario@ansible:~/ejemplo_ansible$ mkdir plantillas
usuario@ansible:~/ejemplo_ansible$ mkdir ficheros
```

(c) Crear el fichero `index.html.j2` con la plantilla parametrizable del *home* por defecto.

```
usuario@ansible:~/ejemplo_ansible$ nano plantillas/index.html.j2
```

```
<html>
<body>
<h1>Pagina por defecto de apache</h1>

<p>Contenido estático, contenido estático</p>

<h1>Servido por {{ansible_hostname}} </h2>
</body>
</html>
```

Nota Sintaxis del lenguaje de plantillas Jinja2

(d) Crear el fichero `phpinfo.php` con un ejemplo de página PHP.

```
usuario@ansible:~/ejemplo_ansible$ nano ficheros/phpinfo.php
```

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

(d) Completar el playbook de Apache con las siguientes **tareas**.

- Añadir la instalación de los paquetes necesarios el soporte de *php* (módulo **apt** iterando con la opción *with_items*)
- Añadir la creación de un *index.html* personalizado para cada servidor sobre la base de una plantilla Jinga2 (módulo **template** reemplando la variable *ansible_hostname*)
- Copiar un archivo *php* de ejemplo a los servidores (módulo **copy**)
- Deshabilitar la opción_keepalive_ en la configuración del servidor Apache (sobreescribíendosu valor con el módulo **replace**)

El fichero `playbook_apache.yml` resultante debe de quedar como:

```
---
- name: instalacion de apache2 y php
  hosts: web
  become: yes
  tasks:
    - name: instalar apache2
      apt: name=apache2 update_cache=yes state=latest

    - name: instalar php
      apt:
        name:
          - libapache2-mod-php
          - php-mysql
          - php-pear
        state: latest
      notify:
        - restart apache2

    - name: crear index.html
      template: src=plantillas/index.html.j2
         dest=/var/www/html/index.html
         backup=yes
```

```
    - name: copiar phpinfo.php
      copy: src=ficheros/phpinfo.php
         dest=/var/www/html/phpinfo.php

  handlers:
    - name: restart apache2
      service: name=apache2 state=restarted
```

(e) Volver a lanzar el playbook y comprobar con un navegador web desde la máquina **ansible** que se accede a los ficheros *index.html* y *phpinfo.php*.

```
usuario@ansible:~/ejemplo_ansible$ ansible-playbook playbook_apache.yml --ask-become-pass
```

Desde la máquina `ansible`, comprobar con el comando `lynx` el acceso a las páginas web de `web1.cda.net`, `web2.cda.net`,`web3.cda.net` y `web4.cda.net`

```
usuario@ansible:~/ejemplo$ lynx web1.cda.net
usuario@ansible:~/ejemplo$ lynx web1.cda.net/phpinfo.php

usuario@ansible:~/ejemplo$ lynx web2.cda.net
usuario@ansible:~/ejemplo$ lynx web2.cda.net/phpinfo.php

usuario@ansible:~/ejemplo$ lynx web3.cda.net
usuario@ansible:~/ejemplo$ lynx web3.cda.net/phpinfo.php

usuario@ansible:~/ejemplo$ lynx web4.cda.net
usuario@ansible:~/ejemplo$ lynx web4.cda.net/phpinfo.php
```

Playbooks sencillos: instalación de HAProxy

(a) Crear el fichero `playbook_haproxy.yml`

```
---
- name: instalacion y configuracion de haproxy
  hosts: haproxy
  become: yes
  vars:
    haproxy_bind_address:  10.10.10.10
    haproxy_bind_port:     80
    haproxy_balance:       roundrobin
    haproxy_cluster_name:  cluster_ansible
    haproxy_servers:
      - name: web1
        address: web1.cda.net:80
      - name: web2
        address: web2.cda.net:80
      - name: web3
        address: web3.cda.net:80
      - name: web4
        address: web4.cda.net:80
  tasks:
```

```
    - name: instalar paquete haproxy
      apt: name=haproxy state=present

    - name: habilitar arranque haproxy
      lineinfile:
        dest: /etc/default/haproxy
        regexp: "^ENABLED.+$"
        line: "ENABLED=1"
        state: present

    - name: crear fichero configuracion
      template:
        src: plantillas/haproxy.cfg.j2
        dest: /etc/haproxy/haproxy.cfg
        mode: 0644
      notify: restart haproxy

  handlers:
    - name: restart haproxy
      service: name=haproxy state=restarted

- name: ajuste de KeepAlive
  hosts: web
  become: yes
  tasks:
    - name: deshabilitar KeepAlive
      replace: dest='/etc/apache2/apache2.conf'
         regexp='KeepAlive On'
         replace='KeepAlive Off'
      notify:
        - restart apache2
  handlers:
    - name: restart apache2
      service: name=apache2 state=restarted
```

(b) Crear el fichero `plantillas/haproxy.cfg.j2`

```
global
        daemon
        user    haproxy
        group   haproxy

defaults
        mode    http
        log     global
        timeout connect 10000ms
        timeout client  50000ms
        timeout server  50000ms

frontend {{haproxy_cluster_name}}
        bind {{haproxy_bind_address}}:{{haproxy_bind_port}}
        mode http
        stats enable
        stats auth  cda:cda
        default_backend {{haproxy_cluster_name}}_servers

backend {{haproxy_cluster_name}}_servers
```

```
        balance {{haproxy_balance}}
        cookie PHPSESSID prefix
{% for backend in haproxy_servers %}
        server {{ backend.name }} {{ backend.address }} cookie {{ backend.name }}
check
{% endfor %}
```

(c) Lanzar el *playbook* (pedirá la contraseña de SUDO para *usuario* [`purple`])

```
usuario@ansible:~/ejemplo# ansible-playbook playbook_haproxy.yml --ask-become-
pass
```

(d) Desde un navegador web comprobar el funcionamiento del balanceador de carga accediendo a la URL **haproxy.cda.net** (comprobar la página de estadísticas en la URL `http://haproxy.cda.net/haproxy?stats`)

```
usuario@ansible:~/ejemplo$ lynx haproxy.cda.net
< recarga con [CTRL]+R >
```

Comprobar en la máquina **haproxy** la configuración de HAProxy creada.

Uso de Roles

Los **Roles** son un mecanismo que permite reutilizar *Plays* (tareas, variables, handler, etc).

Los **Roles** "empaquetan" en un directorio los elementos de un *Play* (tasks, handlers, variables, etc) de modo que puedan ser reutilizados en diferentes *Playbooks*

- Por defecto se buscan directorios con *Roles* en la ubicación global por defecto (`/etc/ansible/roles`), la ubicación por defecto del usuario (`$HOME/.ansible/roles`) o en el directorio `roles` del directorio de trabajo del *Playbook*.
- El nombre que identifica al *Role* coincide con el nombre del directorio donde se declara

Nota: Documentación Roles

Para invocar los *Roles* disponibles desde un *Playbook*, se enumeran sus nombres dentro de la sección `roles`

Es posible instalar nuevos *Roles* usando el comando `ansible-galaxy install <nombre>`, seleccionando uno de los *Roles* disponibles en Ansible Galaxy

Nota: Otro mecanismo (menos potente) para reutilizar "código" Ansible son las sentencias `include`. Por ejemplo:

```
---
- name: un play
  hosts: todos
  tasks:
    - name: instalar cosas
      apt: name=cosas state:latest
    - inclue: otro_fichero.yml
```

Estructura de los Roles

```
roles
├── rol_ejemplo1
│   ├── defaults
│   │   └── main.yml
│   ├── files
│   ├── handlers
│   │   └── main.yml
│   ├── meta
│   │   └── main.yml
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   └── vars
│       └── main.yml
├── rol_ejemplo2
│   ├── defaults
│   │   └── main.yml
... ...
```

Para utilizar los *roles* en el *Playbook* se vincula en la sección `roles` de cada *Play* la lista con los nombre de los *roles* a aplicar en dicha *Play*.

- Ansible usará el `main.yml` de las diferentes secciones del directorio de cada *role* (si está presente) como contenido de las secciones `tasks`, `handlers`, `vars` etc

Ejemplo de definición y uso de Roles

`<pendiente>`