

# Ansible easy guide

|  |          |
|--|----------|
| <b>Introducción</b>                          | <b>1</b> |
| Instalación                                  | 1        |
| Añadir hosts a ansible                       | 2        |
| Configuración SSH                            | 2        |
| Creación de inventario                       | 3        |
| Creación de un playbook                      | 4        |
| Ejecución de un script en múltiples máquinas | 6        |
| Forma convencional                           | 6        |
| Mediante playbooks                           | 6        |

## Introducción

En este documento trataremos Ansible, este es un software diseñado para configurar y administrar ordenadores. También puede ser utilizado como una herramienta de orquestación.

Para realizar esta instalación será común editar archivos, de modo que para facilitar este trabajo, recomendación es la conexión desde una máquina principal (host si estás virtualizando) y el uso de algún terminal flexible, como terminator para facilitar nuestro trabajo. Para más información al respecto puedes consultar mi [guía fácil sobre ssh](#).

## Instalación

Podemos instalar Ansible de forma convencional mediante el paquete apt:

```
sudo apt install ansible
```

## Añadir hosts a ansible

Editando el fichero `/etc/ansible/hosts` añadiremos las ip de nuestros hosts.

```
Ubuntu-server (Pre-Ansible) [Corriendo] - Oracle VM VirtualBox
GNU nano 4.8 /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
# - Comments begin with the '#' character
# - Blank lines are ignored
# - Groups of hosts are delimited by [header] elements
# - You can enter hostnames or ip addresses
# - A hostname/ip can be a member of multiple groups
#
# Ex 1: Ungrouped hosts, specify before any group headers.
#green.example.com
#blue.example.com
#192.168.100.1
#192.168.100.10
#
# Ex 2: A collection of hosts belonging to the 'webservers' group
#[webservers]
#alpha.example.org
#beta.example.org
192.168.56.110 ansible_user=albertopm
192.168.56.103 ansible_user=albertopm_
#
# If you have multiple hosts following a pattern you can specify
# them like this:
#www[001:006].example.com
#
# Ex 3: A collection of database servers in the 'dbservers' group
#[dbservers]
```

## Configuración SSH

Ansible utiliza el intercambio de información mediante SSH, por lo que será necesario disponer de fingerprints de acceso sin contraseña entre el servidor en el que tenemos instalado ansible y los demás hosts. Para más información sobre ssh puedes consultar mi [guía fácil sobre ssh](#).

Para probar que esta conexión se ha realizado correctamente podemos conectarnos a los servidores con ping.

```
sudo ansible all -m ping
```

```
albertopm@ubuntui:~$ ansible all -m ping
192.168.56.110 | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: albertopm@192.168.56.110: Permission denied (publ
ickey,password).",
  "unreachable": true
}
[WARNING]: Platform linux on host 192.168.56.103 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
192.168.56.103 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

Podemos observar un error en el resultado, y si nos fijamos en la información es en el propio servidor. Eso es porque necesitamos añadir nuestra propia fingerprint ssh, para poder hacer así ssh sin contraseña a nuestro propio servidor. Una vez corregido esto, el resultado del ping será correcto.

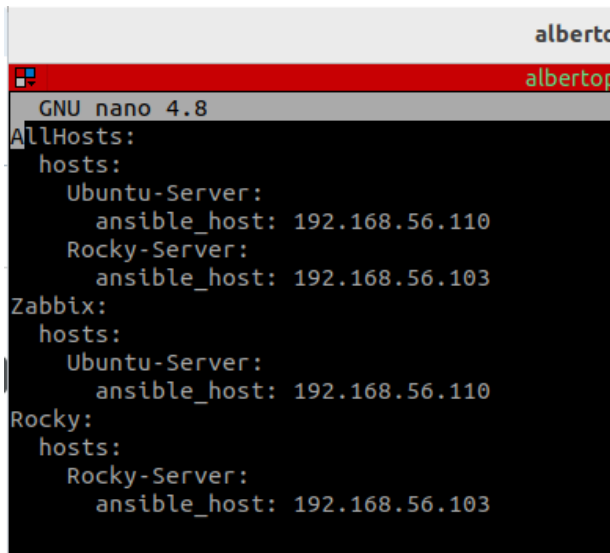
```
albertopm@ubuntui:~$ ansible all -m ping
[WARNING]: Platform linux on host 192.168.56.103 is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
192.168.56.103 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
192.168.56.110 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

## Creación de inventario

Los inventarios son una herramienta que nos permite agrupar nuestros hosts. Podemos usar distintos tipos de archivos de serialización. En este caso haremos un ejemplo en el formato .yaml

Puede encontrar más información en la [documentación oficial](#).

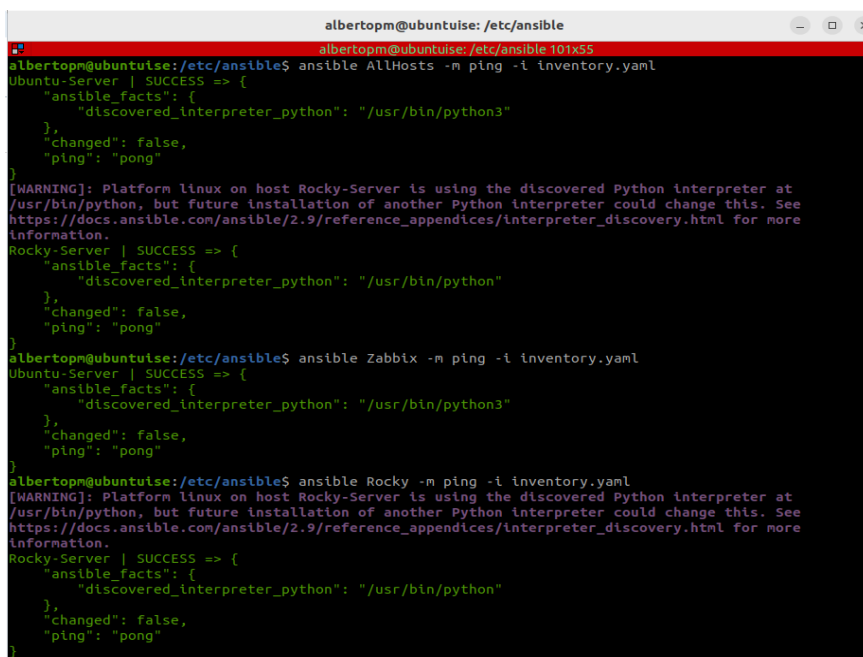
Para ello, en la dirección /etc/ansible crearemos un fichero de inventario con la extensión que vayamos a usar, en nuestro caso .yaml y utilizando el formato que designe la documentación oficial para cada archivo, crearemos los grupos y le asignaremos sus respectivos hosts.



```
alberto
albertopm
GNU nano 4.8
AllHosts:
  hosts:
    Ubuntu-Server:
      ansible_host: 192.168.56.110
    Rocky-Server:
      ansible_host: 192.168.56.103
Zabbix:
  hosts:
    Ubuntu-Server:
      ansible_host: 192.168.56.110
Rocky:
  hosts:
    Rocky-Server:
      ansible_host: 192.168.56.103
```

En este ejemplo hemos creado 3 grupos, uno con todos los hosts (AllHosts), otro exclusivamente con el servidor que gestiona Zabbix (Zabbix) y un tercero con el host de Rocky (Rocky).

Ahora podemos probar el funcionamiento del inventario con los 3 grupos, mediante el comando



```
albertopm@ubuntu: /etc/ansible
albertopm@ubuntu: /etc/ansible 101x55
albertopm@ubuntu: /etc/ansible$ ansible AllHosts -m ping -i inventory.yaml
Ubuntu-Server | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
[WARNING]: Platform linux on host Rocky-Server is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
Rocky-Server | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
albertopm@ubuntu: /etc/ansible$ ansible Zabbix -m ping -i inventory.yaml
Ubuntu-Server | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
albertopm@ubuntu: /etc/ansible$ ansible Rocky -m ping -i inventory.yaml
[WARNING]: Platform linux on host Rocky-Server is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
Rocky-Server | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

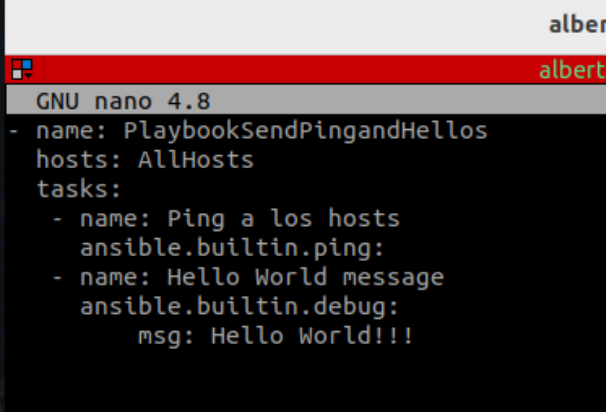
*ansible <grupo> -m  
<comando> -i  
<archivoInventario>*

## Creación de un playbook

Un playbook es una lista de tareas que ansible ejecutará en los servidores que especifiquemos. Al igual que en los inventarios, Ansible nos permite utilizar distintos modelos de archivo organizador. Nosotros volveremos a usar `.yaml`.

Para ello, en la dirección `/etc/ansible` crearemos un fichero de playbook con la extensión que vayamos a usar y en el que introduciremos los códigos correspondientes a lo que queramos ejecutar.

Ejemplo en yaml para los ping y el Hello World en modo debug.

A screenshot of a terminal window. The top bar shows the username 'alber' on the right. Below it, a red bar shows 'albert@alber:~\$'. The terminal content shows the GNU nano 4.8 editor with a YAML playbook named 'PlaybookSendPingandHellos'. The playbook is configured for 'AllHosts' and contains two tasks: a ping task and a debug task that prints 'Hello World!!!'.

```
alber
albert@alber:~$
GNU nano 4.8
- name: PlaybookSendPingandHellos
  hosts: AllHosts
  tasks:
    - name: Ping a los hosts
      ansible.builtin.ping:
    - name: Hello World message
      ansible.builtin.debug:
        msg: Hello World!!!
```

Para la ejecución del playbook usaremos

*`ansible-playbook -i <inventario> <playbook>`*

```
albertopm@ubuntu: /etc/ansible
albertopm@ubuntu: /etc/ansible 203x56
albertopm@ubuntu: /etc/ansible$ sudo nano playbook.yaml
albertopm@ubuntu: /etc/ansible$ ansible-playbook -i inventory.yaml playbook.yaml

PLAY [PlaybookSendPingandHellos] *****

TASK [Gathering Facts] *****
[WARNING]: Platform linux on host Rocky-Server is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
ok: [Rocky-Server]
ok: [Ubuntu-Server]

TASK [Ping a los hosts] *****
ok: [Ubuntu-Server]
ok: [Rocky-Server]

TASK [Hello World message] *****
ok: [Ubuntu-Server] => {
  "changed": false,
  "msg": "Hello World!!!"
}
ok: [Rocky-Server] => {
  "changed": false,
  "msg": "Hello World!!!"
}

PLAY RECAP *****
Rocky-Server      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
Ubuntu-Server    : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
albertopm@ubuntu: /etc/ansible$
```

## Ejecución de un script en múltiples máquinas

### Forma convencional

Ansible nos permitirá ejecutar un script de las máquinas que tengamos agregadas como hosts a ansible. Para ello necesitaremos tener el script accesible **en la misma ruta** en las máquinas en las que se vaya a utilizar. Para ello podríamos o copiar el script a mano en todas las máquinas, o copiarlo mediante algún mecanismo de traspaso de información seguro como es scp.

Una vez lo tengamos en la misma dirección, ejecutaremos un comando con esta estructura:

*ansible <destino> -m <módulo> -a <argumentos del módulo> -i <inventario>*

siendo argumentos del módulo el comando que queremos ejecutar, en este caso, la ejecución de un script en python.

```
albertopm@ubuntu: /etc/ansible 131x56
albertopm@ubuntu: /etc/ansible$ ansible AllHosts -m shell -a 'python3 /home/albertopm/scripts/monitorRaids.py' -i inventory.yaml
Ubuntu-Server | CHANGED | rc=0 >>
---OK_Script---
[WARNING]: Platform linux on host Rocky-Server is using the discovered Python interpreter at
/usr/bin/python, but future installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more
information.
Rocky-Server | CHANGED | rc=0 >>
---OK_Script---
albertopm@ubuntu: /etc/ansible$
```

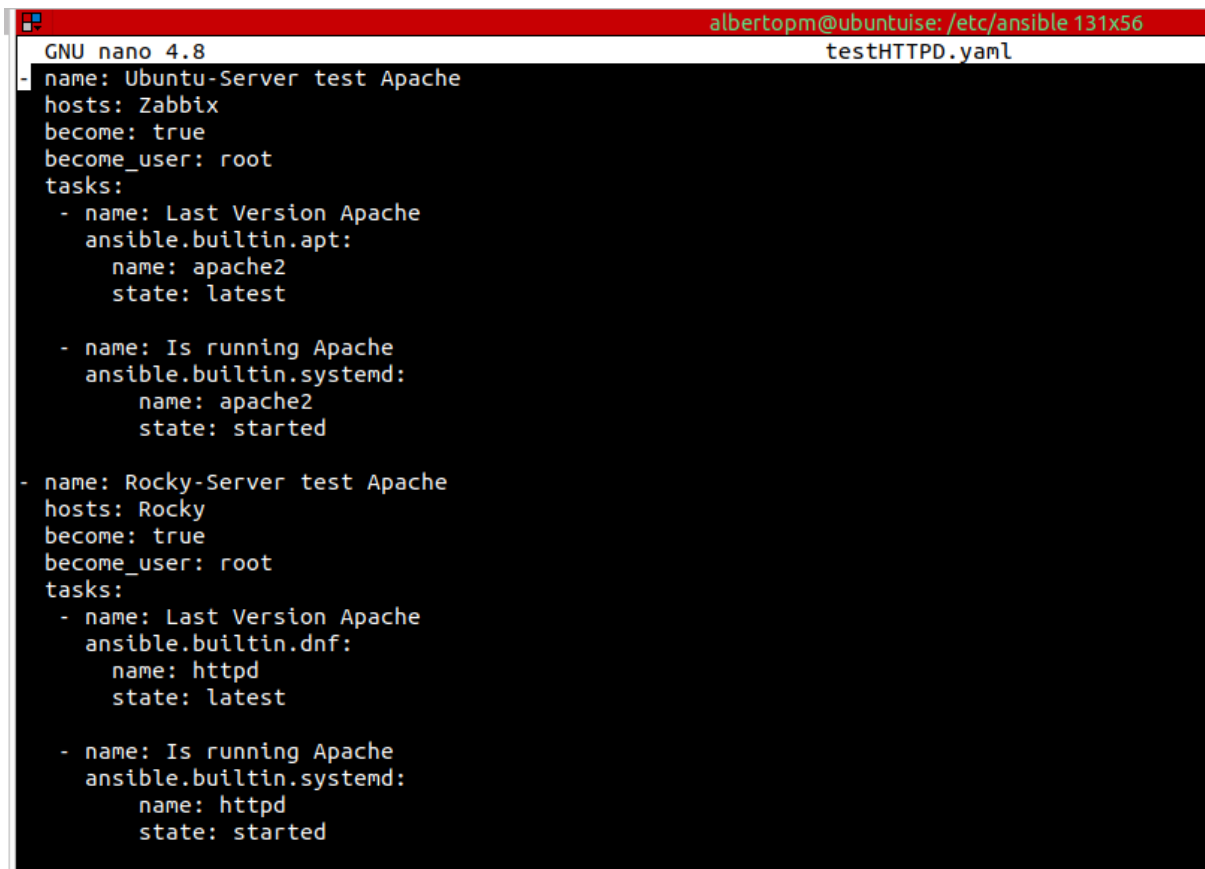
## Mediante playbooks

En el caso del playbook, no es necesario que el script se encuentre en todos los hosts, si no que es suficiente con que se encuentre en la máquina que gestione ansible.

Una vez hecho el playbook, lo ejecutaremos con un comando con la siguiente estructura:

```
ansible-playbook -i <inventario> <playbook>
```

Ejemplo, playbook que se asegura de que tengamos la última versión de httpd:



```
albertopm@ubuntuise: /etc/ansible 131x56
GNU nano 4.8 testHTTPD.yaml
- name: Ubuntu-Server test Apache
  hosts: Zabbix
  become: true
  become_user: root
  tasks:
    - name: Last Version Apache
      ansible.builtin.apt:
        name: apache2
        state: latest

    - name: Is running Apache
      ansible.builtin.systemd:
        name: apache2
        state: started

- name: Rocky-Server test Apache
  hosts: Rocky
  become: true
  become_user: root
  tasks:
    - name: Last Version Apache
      ansible.builtin.dnf:
        name: httpd
        state: latest

    - name: Is running Apache
      ansible.builtin.systemd:
        name: httpd
        state: started
```

En este playbook creamos dos apartados, ya que entre Ubuntu y Rocky hay pequeñas diferencias, como el paquete (apt y dnf) y el nombre de apache (apache2 y httpd). Si compartieran nombres, podríamos haber hecho un solo apartado; sin embargo en estructura, son exactamente iguales ya que realmente hacen lo mismo.

Tal y como explicamos en el apartado de creación de playbooks, le asignaremos un nombre y los hosts a los que van dirigido, para después asignar las tareas que tienen que realizar, en este caso, comprobar mediante su correspondiente gestor de paquetes si se encuentra en la última versión, y después comprobar mediante el systemd si su estado es “iniciado”.

Algunos errores que podemos encontrarnos:

```
albertopm@ubuntu19:~$ ansible-playbook -i inventory.yaml testHTTPD.yaml -k
SSH password:

PLAY [Ubuntu-Server test Apache] *****

TASK [Gathering Facts] *****
Fatal: [Ubuntu-Server]: FAILED! => {"msg": "to use the 'ssh' connection type with passwords, you must install the sshpass program"}

PLAY RECAP *****
Ubuntu-Server      : ok=0    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0

albertopm@ubuntu19:~$
```

En este caso nos dice claramente el error, bastaría con instalar el paquete sshpass.

```
Rocky (Zabbix agent Working) [Corriendo] - Oracle VM VirtualBox

Rocky Linux 9.0 (Blue Onyx)
Kernel 5.14.0-70.30.1.el9_0.x86_64 on an x86_64

localhost login: albertopm
Password:
Last login: Thu Nov 24 11:49:45 on tty1
[albertopm@localhost ~]$ [ 3120.095710] systemd-rc-local-generator[2352]: /etc/rc.d/rc.local is not
marked executable, skipping.
```

(esta nos aparece en el servidor de Rocky-Server, tras ejecutar el playbook en Ubuntu).

De nuevo claramente nos dice que no tiene permisos de ejecución sobre un archivo, de modo que le damos los permisos correspondientes al archivo.

Tras esto, podemos ver como el playbook se ha ejecutado correctamente, y nos indica que se encuentra el servicio de apache actualizado y funcionando en ambos servidores.

```
albertopm@ubuntu19:~$ ansible-playbook -i inventory.yaml testHTTPD.yaml -k
BECOME password:

PLAY [Ubuntu-Server test Apache] *****

TASK [Gathering Facts] *****
ok: [Ubuntu-Server]

TASK [Last Version Apache] *****
ok: [Ubuntu-Server]

TASK [Is running Apache] *****
ok: [Ubuntu-Server]

PLAY [Rocky-Server test Apache] *****

TASK [Gathering Facts] *****
[WARNING]: Platform linux on host Rocky-Server is using the discovered Python interpreter at /usr/bin/python, but future
installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information.
ok: [Rocky-Server]

TASK [Last Version Apache] *****
ok: [Rocky-Server]

TASK [Is running Apache] *****
ok: [Rocky-Server]

PLAY RECAP *****
Rocky-Server      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
Ubuntu-Server     : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

albertopm@ubuntu19:~$
```