

# Aprovisionamiento de Equipos - Ansible



Pablo Escaño Martín

Administración de Sistemas Informáticos en Red

IES Los Manantiales

Curso Académico: 2022/2023

<b>1. Introducción</b>	<b>3</b>
<b>2. Finalidad y Objetivos</b>	<b>3</b>
<b>3. ¿Qué es Ansible?</b>	<b>3</b>
<b>4. Recursos Hardware y Software</b>	<b>4</b>
<b>5. Infraestructura utilizada</b>	<b>4</b>
<b>6. Preparación del entorno</b>	<b>5</b>
6.1.Creación de las máquinas virtuales	5
6.2.Configuraciones de red de las máquinas virtuales	6
6.3. Configuración del servicio SSH	7
6.4. Intercambio de clave SSH	8
<b>7. Instalación de Ansible</b>	<b>8</b>
7.1. Instalación del programa	8
7.2. Configuración de Ansible	9
<b>8. Comandos ad hoc</b>	<b>9</b>
8.1. Comando Ping	10
8.2. Comando Copy	10
8.3. Comando Package	11
8.4. Comando Reboot	11
8.5. Comando Shell	12
8.6. Comando Service	12
8.7. Comando Script	13
<b>9. Instalación de módulos</b>	<b>13</b>
<b>10. PlayBooks</b>	<b>14</b>
10.1. Sintaxis de los Playbooks	15
10.2. Despliegue de servidor web	16
10.3. Despliegue de modo examen	17
<b>11. Conclusiones finales</b>	<b>19</b>
<b>12. Bibliografías</b>	<b>20</b>

# 1. Introducción

Hoy en día las empresas enfrentan una gran cantidad de desafíos a medida que crecen y evolucionan, especialmente en el ámbito de las tecnologías de la información. Esto va desde la configuración, gestión de servidores, dispositivos de red, hasta la implementación de aplicaciones.

Por eso es fundamental contar con herramientas que faciliten estas tareas y mejoren la eficiencia de la organización. Ante este panorama han surgido diversas herramientas que dan solución a este problema como “Chef”, “Puppet” o “SaltStack”. Aunque hay una que sobresale por encima por su sencillez y funcionalidad. Es aquí donde Ansible entra en juego.

## 2. Finalidad y Objetivos

La finalidad de este proyecto es mostrar el funcionamiento de la herramienta Ansible para poder cubrir las necesidades que hemos mencionado anteriormente además de mostrar todo su potencial con sus múltiples posibilidades.

Para mostrar el uso de esta herramienta se creará un entorno virtual que imita un entorno empresarial donde se requiera el aprovisionamiento, gestión y automatización de los mismos.

Los objetivos que se quieren alcanzar son el enseñar como instalar y utilizar esta herramienta en el entorno mencionado donde se probarán todas las funcionalidades que nos aporta Ansible además de ofrecer ejemplos prácticos utilizados en las empresas tecnológicas actuales siendo estos los comandos más utilizados en el día a día, playbooks para la instalación completa de un servidor web con una base de datos configurada y playbooks para desplegar un modo examen en un aula escolar.

## 3. ¿Qué es Ansible?

Antes de continuar vamos a hacer una explicación de Ansible empezando por como lo definen desde la página de su distribuidor oficial RedHat:

*“Ansible es una herramienta open source que automatiza los procesos informáticos para preparar la infraestructura, gestionar la configuración, implementar las aplicaciones y organizar los sistemas, entre otros procedimientos manuales de TI. A diferencia de las herramientas de gestión más simples, los usuarios de Ansible (como los administradores de sistemas, los desarrolladores y los arquitectos) pueden utilizar la automatización que ofrece esta herramienta para instalar sistemas de software, automatizar las tareas diarias, preparar la infraestructura, mejorar la seguridad y el cumplimiento, ejecutar parches en los sistemas y compartir la automatización en toda la empresa.”*

Yo definiría Ansible como un software que te permite realizar instalaciones y configuraciones de equipos de forma masiva siempre realizando las mismas operaciones de forma simultánea.

El funcionamiento de Ansible es simple: se pueden ejecutar comandos de un solo uso sobre el número de equipos que deseemos o podemos confeccionar una lista de tareas para poder ejecutarla en múltiples ocasiones.

Para que Ansible pueda funcionar solo necesita que en el equipo servidor esté instalado Python y que en los equipos donde se vaya a conectar tengan su clave pública para poder conectarse por ssh.

## 4. Recursos Hardware y Software

Para poder realizar el proyecto he optado por realizarlo todo desde un entorno virtualizado por el coste del hardware para poder ser implementado, además he utilizado preferentemente software libres para que este proyecto pueda ser implementado sin costes adicionales al hardware.

Los recursos hardware que se emplearán son: un portátil *Lenovo* con la capacidad de virtualizar los sistema que se van a utilizar en el proyecto y un router movistar inalámbrico para poder acceder a los recursos de la red.

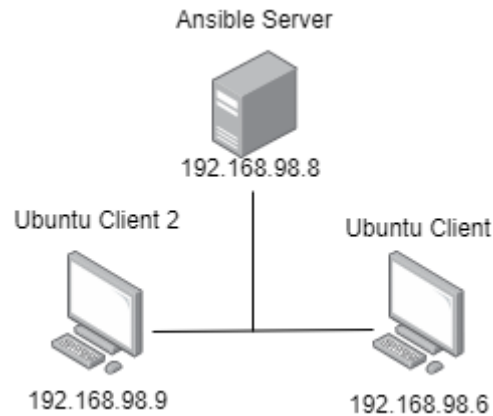
Los recursos software que utilizaré son: *Google Docs* para la redacción de la memoria, *bandicam* para la captura de video para la presentación, *prezi* para la realización de la presentación. *VirtualBox* para la creación y uso de las máquinas virtuales, *Ansible* para el aprovisionamiento de las máquinas además de su gestión, *Python* como dependencia de Ansible ya que está programado en ese lenguaje, *Apache2* como servidor web, *phpMyAdmin* como interfaz web para bases de datos y *MySQL* como sistema gestor de bases de datos.

## 5. Infraestructura utilizada

La infraestructura para este proyecto se reducirá a tres equipos por la falta de capacidad del equipo que utilizaré para virtualizar los sistemas siendo posible escalar lo que se realizará en adelante.

El proyecto utilizará el siguiente esquema (Ilustración 1):

- Servidor Ansible: Esta máquina utilizará el sistema operativo Ubuntu Server 20.04. Tendrá instalada la aplicación de Ansible junto con los recursos, configuraciones y playbooks que se desarrollarán en el proyecto.
- Clientes Linux: Para los clientes Linux se utilizará una versión de Ubuntu Desktop 22.04. Tanto para el servidor como los clientes se utilizará esta distribución porque ya viene preinstalada la aplicación de python 3.0 necesaria para la ejecución de Ansible.



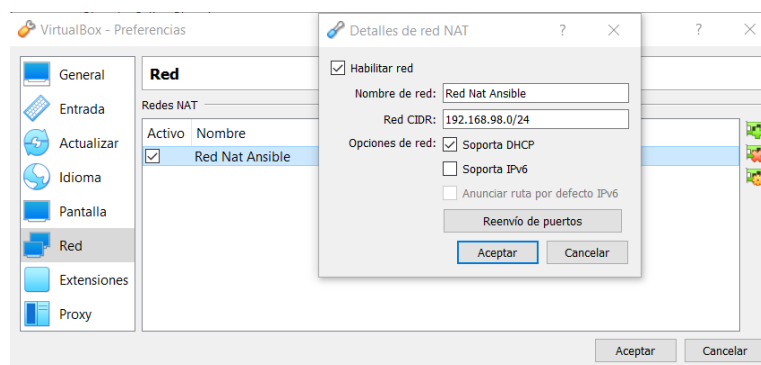
*Ilustración 1: Esquema del proyecto*

## 6. Preparación del entorno

### 6.1. Creación de las máquinas virtuales

Para que Ansible pueda funcionar debemos disponer de una red interna con conexión al exterior si queremos tener la posibilidad de instalar recursos en los equipos host o instalar nuevos módulos que permitan más opciones.

Para crear la red nat que necesitamos debemos acceder a las preferencias de virtualBox situándonos en el apartado de red, desde ahí añadimos la red con la dirección de red que definimos en el esquema de la infraestructura utilizada. Esto nos deberá quedar como en la ilustración 2.



*Ilustración 2: Configuración de la Red Nat*

Configurada la red interna procederemos a crear las máquinas virtuales con las características que se mostrarán en las siguiente ilustraciones.

General	
Nombre:	UbuntuServer20.04
Sistema operativo:	Ubuntu (64-bit)
Ubicación de archivo de preferencias:	C:\Users\Pablo Escaño Martín\VirtualBox VMs\UbuntuServer20.04
Sistema	
Memoria base:	1024 MB
Orden de arranque:	Disquete, Óptica, Disco duro
Aceleración:	VT-x/AMD-V, Paginación anidada, Paravirtualización KVM
Pantalla	
Almacenamiento	
Controlador:	IDE
IDE primario maestro:	[Unidad óptica] ubuntu-20.04.4-live-server-amd64.iso (1,24 GB)
Controlador:	SATA
Puerto SATA 0:	UbuntuServer.vdi (Normal, 20,00 GB)
Audio	
Red	
Adaptador 1:	Intel PRO/1000 MT Desktop (Red NAT, «Red Nat Ansible»)
Adaptador 2:	Intel PRO/1000 MT Desktop (NAT)

*Ilustración 3: Configuración de Ubuntu Server*

General	
Nombre:	UbuntuDesktop22.04
Sistema operativo:	Ubuntu (64-bit)
Ubicación de archivo de preferencias:	C:\Users\Pablo Escaño Martín\VirtualBox VMs\UbuntuDesktop22.04
Sistema	
Memoria base:	1024 MB
Orden de arranque:	Disquete, Óptica, Disco duro
Aceleración:	VT-x/AMD-V, Paginación anidada, Paravirtualización KVM
Pantalla	
Almacenamiento	
Controlador:	IDE
IDE secundario maestro:	[Unidad óptica] ubuntu-22.04.1-desktop-amd64.iso (3,56 GB)
Controlador:	SATA
Puerto SATA 0:	UbuntuDesktop.vdi (Normal, 20,00 GB)
Audio	
Red	
Adaptador 1:	Intel PRO/1000 MT Desktop (Red NAT, «Red Nat Ansible»)
Adaptador 2:	Intel PRO/1000 MT Desktop (NAT)

*Ilustración 4: Configuración de Ubuntu Desktop*

General	
Nombre:	ubuntuDesktop22.04_2
Sistema operativo:	Ubuntu (64-bit)
Ubicación de archivo de preferencias:	C:\Users\Pablo Escaño Martín\VirtualBox VMs\ubuntuDesktop22.04_2
Sistema	
Memoria base:	1024 MB
Orden de arranque:	Disquete, Óptica, Disco duro
Aceleración:	VT-x/AMD-V, Paginación anidada, Paravirtualización KVM
Pantalla	
Almacenamiento	
Controlador:	IDE
IDE secundario maestro:	[Unidad óptica] Vacío
Controlador:	SATA
Puerto SATA 0:	ubuntuDesktop22.04_2.vdi (Normal, 20,00 GB)
Audio	
Red	
Adaptador 1:	Intel PRO/1000 MT Desktop (Red NAT, «Red Nat Ansible»)
Adaptador 2:	Intel PRO/1000 MT Desktop (NAT)

*Ilustración 5: Configuración de Ubuntu Desktop 2*

## 6.2. Configuraciones de red de las máquinas virtuales

Configuradas ya las máquinas procedemos a instalar cada uno de los sistemas operativos. Cuando hayamos terminado revisamos cada una de las configuraciones de red para llegar a tener el esquema de red que se planteó.

```

Network connections [ Help ]
Configure at least one interface this server can use to talk to other machines,
and which preferably provides sufficient access for updates.

NAME    TYPE  NOTES
[ enp0s3 eth - ]
DHCPv4  192.168.98.8/24
08:00:27:03:bf:8f / Intel Corporation / 82540EM Gigabit Ethernet Controller
(PRO/1000 MT Desktop Adapter)

[ enp0s8 eth - ]
DHCPv4  10.0.3.15/24
08:00:27:88:2c:73 / Intel Corporation / 82540EM Gigabit Ethernet Controller
(PRO/1000 MT Desktop Adapter)

```

*Ilustración 6: Configuración de red de Ubuntu Server*

```

pablo@pablo-ubuntu:~/Escritorio$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:5d:1d:e0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.98.6/24 brd 192.168.98.255 scope global dynamic noprefixroute enp0s3
        valid_lft 1160sec preferred_lft 1160sec
    inet6 fe80::ffff:b4150:91ee:fca5/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4e:7f:47 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 brd 10.0.3.255 scope global dynamic noprefixroute enp0s8
        valid_lft 86360sec preferred_lft 86360sec
    inet6 fe80::6e2e:c34a:113b:5d1c/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

```

*Ilustración 7: Configuración de red de Ubuntu Desktop*

```

pablo@cliente2Linux:~/Escritorio$ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:5a:35:80 brd ff:ff:ff:ff:ff:ff
    inet 192.168.98.9/24 brd 192.168.98.255 scope global dynamic noprefixroute enp0s3
        valid_lft 1116sec preferred_lft 1116sec
    inet6 fe80::8e7e:c883:2d82:1baa/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:58:91:3e brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 brd 10.0.3.255 scope global dynamic noprefixroute enp0s8
        valid_lft 86315sec preferred_lft 86315sec
    inet6 fe80::3e29:985a:c3e6:978f/64 scope link noprefixroute
        valid_lft forever preferred_lft forever

```

*Ilustración 8: Configuración de red Ubuntu Desktop 2*

### 6.3. Configuración del servicio SSH

Para poder habilitar el acceso root por SSH y utilizar Ansible debemos seguir los siguientes pasos en cada una de las máquinas:

- Accedemos a la máquina e instalamos ssh con “**sudo apt install ssh**”.
- Una vez instalado abrimos el archivo de configuración SSH utilizando un editor de texto. Ejecutamos el comando “**sudo nano /etc/ssh/sshd\_config**”.
- Buscamos la línea que contiene la configuración **PermitRootLogin**.
- Modificamos la línea a el valor “**PermitRootLogin yes**” y guardamos el archivo.
- Reiniciamos el servicio SSH para que los cambios surtan efecto ejecutando el comando “**sudo service ssh restart**”.
- Configuramos la contraseña de root para que sea sólida y poder acceder con ella para intercambiar la clave SSH con “**passwd root**”.

## 6.4. Intercambio de clave SSH

La clave SSH es lo que usará Ansible para poder acceder remotamente a cada equipo que configuremos en su lista de dispositivos sin necesidad de utilizar la contraseña de cada equipo ya que estará autorizado. Para crear esta clave y compartirla seguiremos los siguiente pasos:

- Desde el servidor ansible utilizamos el siguiente comando para generar la clave ***“ssh-keygen -t rsa -b 4096”***.
- Comprobamos que se haya generado yendo a la ubicación ***“/root/.ssh”***. El fichero con la clave pública que compartiremos con los host es ***“id\_rsa.pub”***.
- Generada la clave la podemos compartir con los equipos Linux mediante el comando ***“ssh-copy-id -i ~/.ssh/id\_rsa.pub root@ip-remoto”***. Tras introducir la contraseña del host al que le enviamos la clave mostrará el mensaje de la ilustración 9.

```
root@ansible:~/ssh# ssh-copy-id -i ~/.ssh/id_rsa.pub root@192.168.98.6
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are alr
eady installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to inst
all the new keys
root@192.168.98.6's password:
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'root@192.168.98.6'"
and check to make sure that only the key(s) you wanted were added.
```

*Ilustración 9: intercambio de claves SSH*

## 7. Instalación de Ansible

Como se mencionó anteriormente Ansible es un programa que utiliza python para poder funcionar, en las distribuciones de Ubuntu Python viene instalado por defecto pero en caso de utilizar otras distribuciones de linux debemos de instalarlo con el comando ***“apt install pip -y”***.

### 7.1. Instalación del programa

Con las dependencias instaladas instalar ansible requerirá los siguientes pasos que se van a detallar a continuación:

- Instalamos todos los software necesarios que utilizará Ansible para las ejecuciones de sus tareas, esto lo realizamos mediante el comando ***“apt install software-properties-common”***.
- Descargamos los repositorios y paquetes de ansible mediante el comando ***“apt-add-repository --yes --update ppa:ansible/ansible”***.
- Descargados los repositorios e instalados los Softwares necesarios actualizamos los paquetes e instalamos Ansible con ***“apt update | apt install ansible”***.

Para comprobar que toda la instalación se ha realizado correctamente y funciona podemos comprobarlo con el comando ***“ansible --version”*** obteniendo el resultado de la ilustración 10.



```
pablo@ansible:~$ ansible --version
ansible [core 2.14.5]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/pablo/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /home/pablo/.ansible/ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] (/usr/bin/python3)
  jinja version = 3.0.3
  libyaml = True
```

*Ilustración 10: Comprobación de la instalación de Ansible*

## 7.2. Configuración de Ansible

Una vez instalado tenemos que configurar Ansible para poder ejecutar comandos o listas de tareas. Para ello debemos de acceder a su fichero de inventario y modificarlo por lo siguiente.

El fichero de inventario está ubicado en “*/etc/ansible/*”. Por defecto se llama “*hosts*”, en el añadimos la línea “*[Linux]*”, esto delimitará en adelante un grupo con el nombre Linux, en las siguiente líneas se deberán de agregar cada uno de los hosts siguiendo la sintaxis “*nombre ansible\_ssh\_host=direccion\_ip*”.

Con el fichero modificado podemos ejecutar un comando de ping para verificar que ansible ha podido conectarse a los equipos del fichero hosts que previamente se les ha debido compartir su clave ssh. El comando y su resultado se muestran en la ilustración 11.

```
root@ansible:~# ansible -m ping Linux
ubuntu | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python3"
  },
  "changed": false,
  "ping": "pong"
}
```

*Ilustración 11: Ejecución del módulo ping a los equipos del grupo Linux*

## 8. Comandos ad hoc

Instalada el servicio de ansible vamos a hablar de una funcionalidad esencial que hace de esta herramienta un sistema imprescindible a la hora de manejar varios sistemas simultáneamente, los comandos **ad-hoc**.

Un comando ad hoc en Ansible es una manera de ejecutar una tarea de un solo uso, sin necesidad de crear y ejecutar un playbook. Estos comandos permiten realizar acciones o cambios en uno o varios hosts de forma rápida y sencilla.

Los comandos ad hoc de Ansible se ejecutan utilizando el terminal del sistema utilizando la sintaxis “*ansible*” seguida de opciones y argumentos que especifican el grupo de hosts, el módulo a ejecutar y los parámetros asociados. Estos comandos ad hoc pueden ir desde tareas sencillas, como ejecutar comandos, copiar archivos, reiniciar servicios o tareas complejas, como la instalación de utilidades o la ejecución de scripts.

Para mostrar la capacidad de estos comandos voy a mostrar los comandos más utilizados en el día a día. En ellos se enseñará su sintaxis y funcionamiento.

## 8.1. Comando Ping

Este comando ya fue mostrado pero no lo explicamos debidamente durante la instalación de Ansible. Este comando nos permite verificar la conectividad entre el controlador y los hosts remotos. Esto puede ayudar a identificar hosts inaccesibles o problemas de conectividad, muy útil en entornos de redes grandes.

La sintaxis de comando es “*ansible <nombre\_del\_grupo> -m ping*” donde **-m** se refiere al tipo de módulo que vamos a emplear que es **ping** que es un módulo que está incluido en el core de Ansible. Si queremos limitar el número de dispositivos del grupo, podemos añadir el flag “*-l <nombre\_del\_host>*”.

Si el resultado del ping es correcto el resultado será el mismo que se mostró en la ilustración 11.

## 8.2. Comando Copy

El comando **copy** como su nombre indica sirve para copiar archivos desde el sistema Ansible hacia los hosts remotos. Proporciona una forma sencilla de transferir archivos y configuraciones necesarios para la administración y manejo de los hosts.

La sintaxis de comando es “*ansible <nombre\_del\_grupo> -m copy -a "src=<ruta\_origen> dest=<ruta\_destino>"*” donde **-m** se refiere al tipo de módulo que vamos a emplear que es **copy** que es un módulo que está incluido en el core de Ansible, **-a** se utiliza para especificar las variables de entorno de los comandos, en este caso se utiliza para especificar la ruta de origen en el sistema Ansible y la ruta de destino donde se almacenarán en los hosts.

Con este comando también podemos personalizar los permisos de los ficheros que queremos enviar añadiendo los siguientes parámetros adyacentes a las rutas. Si deseamos cambiar los permisos de los ficheros utilizaremos “*mode=644*” (lectura y escritura para el propietario, lectura para el grupo y otros), si deseamos cambiar también el propietario utilizamos “*owner=user*” y también si queremos limitar el número de dispositivos del grupo, podemos añadir el flag “*-l <nombre\_del\_host>*”.

Un ejemplo del comando completo para enviar el fichero prueba.txt con permisos 664 y el usuario pablo al host ubuntu sería “*ansible Linux -m copy -a "src=prueba.txt dest=/etc/pruebas/ mode=644 owner=pablo" -l ubuntu*”.

### 8.3. Comando Package

El comando **package** sirve para gestionar paquetes en los sistemas operativos de los hosts. Permite instalar, actualizar o eliminar paquetes de software de manera eficiente y consistente en múltiples hosts de forma automatizada. con él además puedes especificar el estado deseado de los paquetes y dejar que Ansible se encargue de realizar las acciones necesarias para cumplir con ese estado en cada host

La sintaxis de comando es “*ansible <nombre\_del\_grupo> -m package -a "name=paquete state=estado"*” donde **-m** se refiere al tipo de módulo que vamos a emplear que es **package** que es un módulo que está incluido en el core de Ansible, **-a** se utiliza para especificar las variables de ejecución de los comandos, en este caso se utiliza para especificar el nombre del paquete con “*name=paquete*” y la acción que vamos realizar con “*state=estado*” que puede ser varias opciones desde instalarlo o eliminarlo.

Los estados que pueden ser utilizados con este comando son los siguientes:

- **Present:** El paquete deberá estar instalado y si no lo está ansible lo instalará.
- **Absent:** El paquete deberá estar ausente y si no lo está ansible lo eliminará.
- **Latest:** El paquete deberá estar instalado con la versión más reciente. Si el paquete no está instalado se encargará de instalarlo y si se encuentra pero no está en la versión más reciente este lo actualizará.
- **Held:** El paquete se bloqueará y no se actualizará. Puede que dependiendo del gestor de paquetes este no funcione.
- **Purged:** El paquete junto con sus ficheros de configuración relacionados del sistema han de ser eliminados. Puede que dependiendo del gestor de paquetes este no funcione.

Un ejemplo del comando para la instalación del módulo de apache2 limitado a ubuntu es: “*ansible Linux -m package -a "name=apache2 state=present" -l ubuntu*”.

### 8.4. Comando Reboot

El comando **reboot** sirve para reiniciar los hosts de manera controlada. Permite programar reinicios de forma automatizada y proporciona opciones para gestionar el reinicio. Este comando es muy útil para aplicar los cambios de configuración o las instalaciones de paquetes en los hosts y hacer que el sistema se inicie en un estado óptimo.

La sintaxis de comando es “*ansible <nombre\_del\_grupo> -m reboot*” donde **-m** se refiere al tipo de módulo que vamos a emplear que es **reboot** que es un módulo que está incluido en el core de Ansible.

Este comando también tiene la posibilidad de ejecutarse con argumentos permiten desde enviar mensajes e incluso aplicar un delay antes de ejecutarse. Un ejemplo del

comando con un delay de 100 segundo limitado a ubuntu es: “***ansible Linux -m reboot -a "delay=100" -l ubuntu***”.

## 8.5. Comando Shell

El comando **shell** sirve para ejecutar comandos en los hosts. Estos comandos son ejecutados en el shell pudiendo trabajar con diferentes usuario y privilegios. Existe el comando **command** que es similar pero se diferencia en que este último se ejecuta a nivel del sistema y no puede utilizar las tuberías.

La sintaxis de comando es “***ansible <nombre\_del\_grupo> -m shell -a "<comando>"***” donde **-m** se refiere al tipo de módulo que vamos a emplear que es **shell** que es un módulo que está incluido en el core de Ansible, **-a** se utiliza para especificar las variables de ejecución de los comandos, en este caso se utiliza para especificar el comando que se quiere ejecutar en el sistema. Este comando cuenta con argumentos para que este se ejecute con privilegios de superusuario con “***--become***” o que se ejecute a través de un usuario con privilegios con “***--become-user=<usuario>***”.

El ejemplo que se muestra a continuación es un comando que enviará un comando sleep que esperara ese tiempo antes de reiniciar el equipo ubuntu mediante el superusuario root “***ansible Linux -m shell -a "sleep 20 && reboot" --become --become-user=root -l ubuntu***”.

## 8.6. Comando Service

El comando **service** sirve para administrar servicios en los hosts. Permite controlar el estado de los servicios, como iniciarlos, detenerlos, reiniciarlos o verificar si están en ejecución. Esto nos ayuda a garantizar el funcionamiento de los servicios y su gestión.

La sintaxis de comando es “***ansible <nombre\_del\_grupo> -m service -a "name=servicio state=estado"***” donde **-m** se refiere al tipo de módulo que vamos a emplear que es **service** que es un módulo que está incluido en el core de Ansible, **-a** se utiliza para especificar las variables de ejecución de los comandos, en este caso se utiliza para especificar el servicio con “***name=servicio***” y el estado deseado para el servicio con “***state***”. Este comando cuenta con argumentos para que este se ejecute con privilegios de superusuario con “***--become***” o que se ejecute a través de un usuario con privilegios con “***--become-user=<usuario>***”.

Los estados que pueden ser utilizados con este comando son los siguientes:

- **Started**: El servicio deberá estar iniciado.
- **Stopped**: El servicio deberá estar detenido.
- **Restarted**: El servicio deberá reiniciarse.
- **Reloaded**: El Servicio deberá recargarse.

El ejemplo que se muestra a continuación se iniciará el servicio de apache2 desde el usuario root con privilegios de superusuario en el host ubuntu “***ansible Linux -m service -a "name=apache2 state=started" --become --become-user=root -l ubuntu***”.

## 8.7. Comando Script

El comando **script** sirve para ejecutar ficheros de comandos o código ejecutable en lenguajes como Shell, Python, Perl, etc. Proporciona la opción de recibir los resultados.

La sintaxis de comando es “***ansible <nombre\_del\_grupo> -m script -a "ruta\_del\_script"***” donde **-m** se refiere al tipo de módulo que vamos a emplear que es **script** que es un módulo que está incluido en el core de Ansible, **-a** se utiliza para especificar las variables de ejecución de los comandos, en este caso se utiliza para especificar la ruta donde se encuentra el script que deseamos ejecutar.

El ejemplo que se muestra a continuación ejecutará un script que cree varios directorios en el hosts, el script está ubicado en el directorio “/root/”, “***ansible Linux -m script -a "/root/mkdir.sh"***”.

## 9. Instalación de módulos

Ansible cuenta con muchas funcionalidades de base pero esto no es todo el potencial que tiene la herramienta, existe toda una comunidad de desarrolladores que han reunido una colección de módulos disponibles de forma gratuita junto con documentación en [Ansible Galaxy](#).

Estos módulos aportan funcionalidades ampliadas para interactuar con diferentes sistemas, servicios y recursos. Esto nos ayudará a mejorar la automatización y la interacción con sistemas como **CentOS** cuyo gestor de paquetes no es compatible con Ansible, pero gracias a colecciones como “***community.general***” que incluye el módulo “***yum***” podremos trabajar con él.

Para instalar las colecciones ansible cuenta con el siguiente comando cuya sintaxis es “***ansible-galaxy collection install <nombre\_de\_la\_colección>***”. En el caso de no necesitar una colección también podemos desinstalar una colección con “***ansible-galaxy collection remove <nombre\_de\_la\_colección>***”.

En la siguiente ilustración se muestra la instalación de la colección “***community.general***” ya que sus módulos proporcionan funcionalidades adicionales y tratan casos específicos que pueden no estar soportados por los módulos estándar de Ansible.

```
root@ansible:~# ansible-galaxy collection install community.general
Starting galaxy collection install process
Process install dependency map
Starting collection install process
Downloading https://galaxy.ansible.com/download/community-general-7.0.1.tar.gz to /root/.ansible/tmp/
/ansible-local-984nnkkehho/tmpv78thb9e/community-general-7.0.1-m2izafe0
Installing 'community.general:7.0.1' to '/root/.ansible/collections/ansible_collections/community/ge
neral'
community.general:7.0.1 was installed successfully
```

*Ilustración 12: Instalación de la colección community.general.*

Un ejemplo de la utilidad de esta colección es el módulo para el apagado de un equipo que utilizando las herramientas del core de Ansible arroja un resultado que no podemos distinguir de un error. En las ilustraciones siguientes veremos la diferencia de utilizar estos módulos.

```
root@ansible:~# ansible -m shell -a "poweroff" Linux
ubuntu | UNREACHABLE! => {
  "changed": false,
  "msg": "Failed to connect to the host via ssh: Shared connection to 192.168.98.6 closed.",
  "unreachable": true
}
```

*Ilustración 13: Apagado del sistema con shell de Ansible core.*

```
root@ansible:~# ansible Linux -m community.general.shutdown
ubuntu | CHANGED => {
  "changed": true,
  "shutdown": true,
  "shutdown_command": "/sbin/shutdown -h 0 \"Shut down initiated by Ansible\""
```

*Ilustración 14: Apagado del sistema con el módulo community.general.shutdown.*

## 10. PlayBooks

Ya hemos visto parte del potencial de Ansible mediante los comandos **ad hoc** con varios de sus módulos pero su principal funcionalidad son los **Playbooks**, esto es la manera de definir tareas, organizarlas y ejecutarlas de forma automática en nuestros hosts.

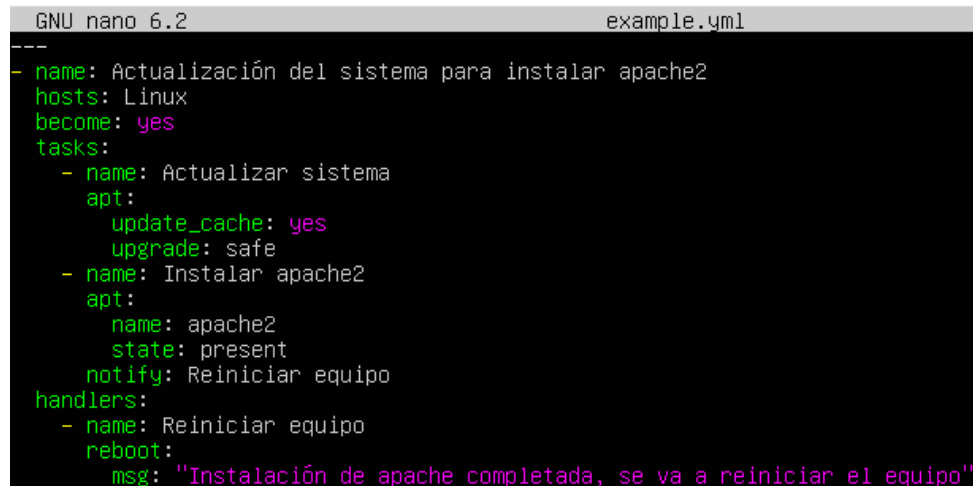
Los playbooks en Ansible proporcionan una estructura flexible y legible en formato **YAML** que permite describir cada tarea con el estado al que se desea llegar. La diferencia principal que tienen estas listas de tareas frente a los comandos ad hoc es que estos últimos son para un uso más simple y para tareas repetitivas son un inconveniente al tener que redactarlos nuevamente.

Además de las tareas, los playbooks también pueden incluir variables para personalizar la configuración en función del hosts, condiciones que deben cumplirse para la ejecución y se pueden agrupar las tareas en bloques para poder controlar fácilmente su ejecución. Gracias a esto nos permite automatizar tareas repetitivas, desde simples configuraciones hasta despliegues complejos y la gestión de infraestructuras. Al tener todo esto como código facilitará la colaboración de las personas o equipos responsables de la infraestructura mejorando la calidad de la misma.

## 10.1. Sintaxis de los Playbooks

Como se ha mencionado en la explicación de los playbooks estos son ficheros YAML por lo que los ficheros tendrán que crearse con el nombre que deseemos y la extensión .yaml.

Para facilitar la comprensión de la sintaxis de estos ficheros voy a explicar un ejemplo sencillo para actualizar los paquetes para posteriormente instalar apache2 y reiniciar el sistema. El playbook se muestra en la ilustración 15.



```
GNU nano 6.2                                example.yaml
---
- name: Actualización del sistema para instalar apache2
  hosts: Linux
  become: yes
  tasks:
    - name: Actualizar sistema
      apt:
        update_cache: yes
        upgrade: safe
    - name: Instalar apache2
      apt:
        name: apache2
        state: present
      notify: Reiniciar equipo
  handlers:
    - name: Reiniciar equipo
      reboot:
        msg: "Instalación de apache completada, se va a reiniciar el equipo"
```

*Ilustración 15: Playbook de ejemplo.*

A continuación se va a detallar todos los componentes de esta playbook que contiene muchos de los componentes que se emplean en el desarrollo de estas listas, antes aclarar que estos fichero empiezan con (---) y para las **indentaciones** se utilizan **2 espacios**. Explicadas estas reglas sobre los ficheros yaml los componentes que se emplean en la playbook son los siguiente:

- **Nombre del playbook (name):** Especifica un nombre para el playbook. En este caso, el nombre es "Instalar Apache y reiniciar si tiene éxito".
- **Hosts (hosts):** Indica el grupo de hosts que se ejecutará el playbook.
- **Privilegios de superusuario (become):** Establece que las tareas se ejecuten con privilegios de superusuario (root). Al tener el valor **yes** indica que se ejecuten todas las tareas con estos privilegios.
- **Tareas (tasks):** Es lista de acciones que se ejecutarán en los hosts. Cada tarea debe de tener un nombre (name) y el módulo que utilizará que será la acción.
- **Tarea de actualizar el sistema:** Esta tarea utiliza el módulo **apt** que es el utilizado para sistemas Debian/Ubuntu. Los argumentos del módulo son "**update\_cache**" con valor **yes** para actualizar la caché de paquetes y "**upgrade**" con valor **safe** para ejecutar solo las actualizaciones seguras.
- **Tarea de instalar Apache:** emplea también el módulo **apt** con los siguientes argumentos "**name**" para indicar el nombre del módulo que se instalará siendo este "**apache2**" y el estado "**state**" con valor "**present**" que indica que deseamos que este activo e instalado.
- **Condición de ejecución (notify):** Indica las tareas adicionales que se deben ejecutar en caso de que la ejecución de una tarea realice cambios en el sistema. En

este caso si hay cambios al instalar apache ejecutará la tarea especial “*Reiniciar equipo*”.

- **Tareas especiales (handlers):** Son las tareas especiales que serán ejecutadas cuando un **notify** lo solicite, es este caso solo hay una tarea llamada “*Reiniciar equipo*” que usa el módulo reiniciar con un mensaje.

```
root@ansible:~/ejemplo_sintaxis# ansible-playbook example.yml
PLAY [Actualización del sistema para instalar apache2] *****
TASK [Gathering Facts] *****
ok: [ubuntu]
TASK [Actualizar sistema] *****
ok: [ubuntu]
TASK [Instalar apache2] *****
changed: [ubuntu]
RUNNING HANDLER [Reiniciar equipo] *****
changed: [ubuntu]
PLAY RECAP *****
ubuntu : ok=4  changed=2  unreachable=0  failed=0  skipped=0  rescued=
0  ignored=0
```

*Ilustración 16: ejecución de la playbook de ejemplo.*

Para ejecutar cualquier playbook podemos hacerlo con el comando “*ansible-playbook <nombre\_del\_playbook>*” en caso de querer limitar los hosts a los que les afecta podemos utilizar como en los comandos ad hoc el flag **-l**. En la ilustración 16 se muestra la ejecución completa del playbook de ejemplo el cual aplicó correctamente el playbook.

## 10.2. Despliegue de servidor web

Para este playbooks se instalará un servidor web apache2 con una base de datos MySQL junto con phpMyAdmin. El objetivo de esta playbook es el alta de un servidor web con una base de datos pasada como parámetro junto con un usuario para su gestión a través del interfaz web phpMyAdmin.

Para ello se utilizará la serie de playbooks ubicada en la ruta “*/root/servidor\_web/*”, está playbook resume gran parte de lo realizado en la asignatura de seguridad y alta disponibilidad ya que procuraremos que el despliegue sea con las condiciones de seguridad apropiadas.

Como requisitos para el despliegue tendremos que instalar el paquete de módulos “*community.mysql*” ya que para interactuar con MySQL necesitaremos dos módulos que contiene esta colección además de instalar Python en el cliente.

Para su ejecución contamos con 3 playbooks diferenciadas que deben de ejecutarse en el siguiente orden “*dependencias.yml*” → “*instalar\_servidor\_web.yml*” → “*config\_mysql.yml*”. Cada una de estas listas de tareas contienen toda la información necesaria en su código para entender su funcionamiento.

A continuación se mostrarán unas ilustraciones del resultado de su ejecución, los resultados con la respuesta “**Changed: [nombre\_del\_hosts]**” en amarillo indican que Ansible ha ejecutado cambios en el sistema, los resultados en verde con “**Ok:**



[nombre\_del\_hosts]” indican que Ansible ha ejecutado la tarea pero no ha modificado el sistema porque ha comprobado que la acción que iba a realizar ya ha sido completada.

```
root@ansible:~/servidor_web/playbooks# ansible-playbook dependencias.yml -i ubu
PLAY [Instalación de dependencias y actualización del sistema] *****
TASK [Gathering Facts] *****
ok: [ubu]
TASK [Actualizar el sistema] *****
changed: [ubu]
TASK [Instalar Python3.0] *****
changed: [ubu]
TASK [Instalar PyMySQL] *****
changed: [ubu]
PLAY RECAP *****
ubu : ok=4 changed=3 unreachable=0 failed=0 skipped=0 rescued=
0 ignored=0
```

*Ilustración 17: Instalación de las dependencias.*

```
root@ansible:~/servidor_web/playbooks# ansible-playbook instalar_servidor_web.yml -i ubu
PLAY [Instalación del servidor Web] *****
TASK [Gathering Facts] *****
ok: [ubu]
TASK [Instalación de Apache] *****
changed: [ubu]
TASK [Instalación de MySQL server] *****
changed: [ubu]
TASK [Instalación de PHP con los módulos necesarios] *****
changed: [ubu]
TASK [Reiniciar Apache] *****
changed: [ubu]
PLAY RECAP *****
ubu : ok=5 changed=4 unreachable=0 failed=0 skipped=0 rescued=
0 ignored=0
```

*Ilustración 18: Instalación del servidor web.*

```
root@ansible:~/servidor_web/playbooks# ansible-playbook config_mysql.yml -i ubu
PLAY [Playbook para instalar herramientas adicionales] *****
TASK [Gathering Facts] *****
ok: [ubu]
TASK [Descargar phpMyAdmin] *****
changed: [ubu]
TASK [Instalar unzip] *****
ok: [ubu]
TASK [Descomprimir phpMyAdmin] *****
changed: [ubu]
TASK [Copiar phpMyAdmin] *****
changed: [ubu]
TASK [Cambiar el propietario y el grupo de phpMyAdmin] *****
changed: [ubu]
TASK [Crear la base de datos para phpMyAdmin] *****
changed: [ubu]
TASK [Importar la base de datos de phpMyAdmin] *****
changed: [ubu]
TASK [Crear el usuario de phpMyAdmin] *****
changed: [ubu]
PLAY RECAP *****
ubu : ok=9 changed=7 unreachable=0 failed=0 skipped=0 rescued=
0 ignored=0
```

*Ilustración 19: Instalación de phpMyAdmin y configuración del servidor web.*

### 10.3. Despliegue de modo examen

El objetivo de este playbook es desplegar un modo examen en un entorno controlado donde los usuarios accedan a equipos previamente configurados para poder realizar en ellos cualquier prueba enviando un enunciado a su escritorio y deshabilitando las aplicaciones como navegadores web etc evitando trampas.

Este playbook está inspirado en el modo examen que se despliega en el entorno donde he realizado mi fct donde ha sido muy útil contar con ello para poder realizar pruebas de forma rápida y sencilla.

Se compone de tres listas de tareas realizando la instalación de *Visual Studio Code* como entorno de desarrollo, el despliegue del **modo exam** creando un usuario con el examen en su escritorio, aplicaciones como *Firefox* deshabilitadas para evitar trampas y el despliegue de el **modo site** que revierte esto cambios cerrando las sesiones abiertas además de revertir los cambios.

Este playbook cuenta con el inconveniente de que no deben existir usuario previos con el inicio de sesión automático ya que al cargar el modo exam no nos deja la posibilidad de cambiar de usuarios teniendo que volver al modo site y cambiar esta configuración.

A continuación se mostrarán unas ilustraciones del resultado de las ejecuciones de las listas mencionadas anteriormente:

```
root@ansible:~/modo_examen/playbooks# ansible-playbook exam.yml -i ubu
PLAY [Despliegue del modo examen] *****
TASK [Gathering Facts] *****
ok: [ubu]

TASK [Obtener lista de usuarios] *****
changed: [ubu]

TASK [Bloquear usuarios] *****
changed: [ubu] => (item=nobody)
changed: [ubu] => (item=pablo)

TASK [Crear el usuario 'exam'] *****
[DEPRECATION WARNING]: Encryption using the Python crypt module is deprecated. The Python crypt
module is deprecated and will be removed from Python 3.13. Install the passlib library for
continued encryption functionality. This feature will be removed in version 2.17. Deprecation
warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [ubu]

TASK [Copiar archivo PDF al escritorio de 'exam'] *****
ok: [ubu]

TASK [Deshabilitar navegador Firefox] *****
changed: [ubu]

TASK [Reiniciar el sistema para aplicar la configuración] *****
changed: [ubu]

PLAY RECAP *****
ubu          : ok=7  changed=5  unreachable=0  failed=0  skipped=0  rescued=
0  ignored=0
```

*Ilustración 20: despliegue del modo exam.*

```
root@ansible:~/modo_examen/playbooks# ansible-playbook site.yml -i ubu
PLAY [Despliegue del modo site para volver a la configuración anterior] *****
TASK [Gathering Facts] *****
ok: [ubu]

TASK [Obtener el ID del usuario exam] *****
changed: [ubu]

TASK [Cerrar sesión del usuario] *****
changed: [ubu]

TASK [Obtener lista de usuarios] *****
changed: [ubu]

TASK [Desbloquear usuarios] *****
changed: [ubu] => (item=nobody)
changed: [ubu] => (item=pablo)
changed: [ubu] => (item=exam)

TASK [Eliminar el usuario 'exam'] *****
changed: [ubu]

TASK [Habilitar el navegador Firefox] *****
changed: [ubu]

TASK [Reiniciar el sistema para aplicar la configuración] *****
changed: [ubu]

PLAY RECAP *****
ubu          : ok=8  changed=7  unreachable=0  failed=0  skipped=0  rescued=
0  ignored=0
```

*Ilustración 21: Despliegue del modo site.*

```
root@ansible:~/modo_examen/playbooks# ansible-playbook aplicaciones.yml -l ubu
PLAY [Intalación de aplicaciones para realizar el examen] *****
TASK [Gathering Facts] *****
ok: [ubu]
TASK [Descargar Visual Studio Code] *****
changed: [ubu]
TASK [Instalar Visual Studio Code] *****
ok: [ubu]
PLAY RECAP *****
ubu : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=
0 ignored=0
```

*Ilustración 22: Instalación de Visual Studio Code.*

## 11. Conclusiones finales

En conclusión, en este proyecto de fin de grado se ha llevado a cabo la configuración e instalación de varios sistemas operativos como servidores, centrándonos en Ubuntu Server 20.04 y Ubuntu Desktop 22.04. Utilizando la potente herramienta de automatización Ansible.

A lo largo del proyecto, se ha abordado la configuración de la red, la instalación de diversos servicios y la creación de una jerarquía de sistemas, con un servidor padre encargado de administrar el automatizador de procesos. Además, se ha implementado un servidor web con una base de datos para las implementaciones de aplicaciones web que requieren las empresas de software aprovechando la automatización de Ansible.

La simulación de los servidores se ha realizado mediante la virtualización, permitiendo crear una macro red de forma virtualizada. Se ha puesto especial énfasis en la seguridad y privacidad, implementando protocolos seguros para prevenir posibles ataques y garantizar la integridad de los sistemas.

Este proyecto ha permitido adquirir conocimientos valiosos sobre la configuración y administración de servidores utilizando Ansible, así como sobre la implementación de redes virtuales y la gestión de la seguridad. Gracias al apoyo de mi tutor laboral Alexander Bello he podido aprender bien esta herramienta por sus consejos e indicaciones para realizarlo con las medidas de seguridad óptimas.

Este proyecto originalmente también contaría con el soporte en Windows pero debido a dificultades a la hora de su implementación es una mejora que se puede implementar a futuro. También se puede mejorar el despliegue del modo examen para poder adaptarlo con un servidor proxy como Nginx para poder utilizar el navegador web solo con las páginas que autoricemos por ejemplo.

En resumen, este proyecto ha sido una gran experiencia personal donde he podido observar la necesidad de este tipo de herramientas para la gestión de servidores de forma eficiente y segura. Así como la protección de las infraestructuras de manera robusta.

## 12. Bibliografías

Recursos web empleados en el desarrollo del tfg:

- [Página de la empresa desarrolladora de Ansible Red Hat](#)
- [Documentación oficial de Ansible](#)
- [Página oficial de Ansible Galaxy](#)
- [Tutorial de sintaxis YAML](#)
- [Keywords para Playbooks](#)