

# Introducción a Ansible y AWS EC2

## Tabla de contenidos

1. Objetivo de este tutorial .....	2
2. ¿Qué es Ansible? .....	2
3. ¿Qué es AWS? .....	2
4. ¿Qué es Amazon EC2? .....	2
5. Descripción del escenario utilizado en este tutorial .....	2
6. Instalación de Ansible .....	3
7. Instalación de herramientas adicionales .....	4
7.1. Instalación y configuración de la extensión Ansible para Visual Studio Code .....	4
7.2. Instalación de la herramienta <code>ansible-lint</code> .....	4
8. Configuración del archivo de inventario de Ansible .....	4
9. Conexión SSH con las instancias EC2 de AWS .....	5
10. Módulos de Ansible .....	5
10.1. Módulo <code>ping</code> .....	6
10.2. Módulo <code>shell</code> .....	6
10.3. Módulo <code>command</code> .....	7
10.4. Módulo <code>apt</code> .....	7
11. ¿Qué es un <code>Playbook</code> ? .....	8
11.1. Ejemplo de instalación del servidor web <code>apache</code> .....	8
11.2. Ejecución de un <code>Playbook</code> .....	9
12. Ejemplos .....	11
12.1. Ejemplo 1 .....	12
12.2. Ejemplo 2 .....	13
12.3. Ejemplo 3 .....	14
12.4. Ejemplo 4 .....	15
12.5. Ejemplo 5 .....	16
12.6. Ejemplo 6 .....	17
12.7. Ejemplo 7 .....	18
12.8. Ejemplo 8 .....	19
12.9. Ejemplo 9 .....	20
12.10. Ejemplo 12 .....	22
12.11. Ejemplo 13 .....	23
Autor .....	26
Licencia .....	26
Referencias .....	26

# 1. Objetivo de este tutorial

El objetivo de este tutorial es entender algunos de los conceptos básicos de [Ansible](#) y ponerlos en práctica realizando tareas sencillas de administración sobre instancias [EC2 de AWS](#).

## 2. ¿Qué es Ansible?

[Ansible](#) es una herramienta que nos permite configurar, administrar y realizar instalaciones en sistemas cloud con múltiples nodos sin tener que instalar agentes software en ellos. Sólo es necesario instalar Ansible en la máquina principal desde la que vamos a realizar operaciones sobre el resto de nodos y ésta se conectará a los nodos a través de SSH.

[Ansible](#) utiliza archivos [YAML](#) para describir las configuraciones que queremos aplicar en cada uno de los nodos. Estos archivos de configuración se conocen como [Playbooks](#).



Si todavía no está familiarizado con [YAML](#) se recomienda la lectura del tutorial [Aprenda YAML en 5 minutos](#).

## 3. ¿Qué es AWS?

[Amazon Web Services \(AWS\)](#) es una plataforma de computación en la nube que ofrece gran variedad de servicios. En este tutorial haremos uso del servicio [Amazon Elastic Compute Cloud \(Amazon EC2\)](#).

## 4. ¿Qué es Amazon EC2?

[Amazon Elastic Compute Cloud \(Amazon EC2\)](#) es un servicio que permite a los usuarios crear máquinas virtuales de tamaño modificable para ejecutar sus aplicaciones informáticas. EC2 permite pagar únicamente por la capacidad utilizada, en lugar de comprar o alquilar una máquina para utilizarla durante varios meses o años, en EC2 se alquila la capacidad por horas.

## 5. Descripción del escenario utilizado en este tutorial

El escenario que vamos a utilizar en este tutorial estará formado por **un nodo principal** donde vamos a instalar [Ansible](#) y **dos instancias EC2 de AWS** que serán los nodos sobre los que vamos a realizar las tareas de configuración y administración.

El nodo principal no necesita ninguna característica especial, puede ser cualquier equipo. En este tutorial utilizaremos nuestro equipo de trabajo local con el sistema operativo Ubuntu.

Las instancias EC2 de AWS utilizarán la AMI con la última versión de Ubuntu Server disponible, pero podría ser cualquier otra AMI.

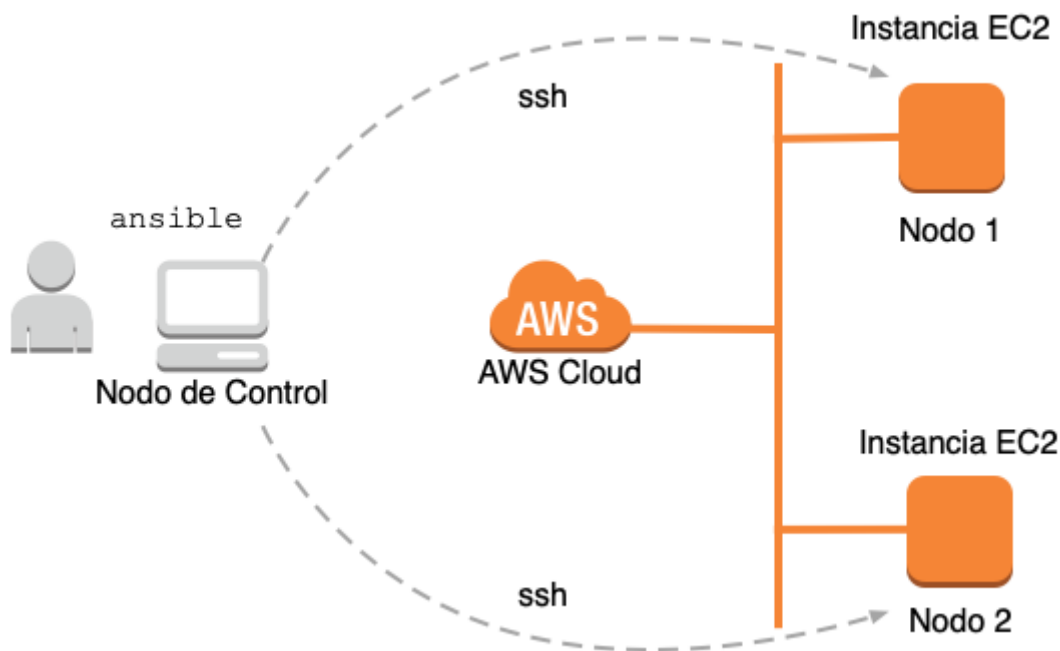


Figura 1. Escenario con el nodo principal y las dos instancias EC2 de AWS

Una vez creadas las instancias en AWS anotamos las direcciones IP públicas que se le han asignado en AWS, porque más adelante serán necesarias para crear el archivo de inventario de [Ansible](#). En mi caso he obtenido las siguientes direcciones.

Nodos	IP
Nodo 1	18.206.58.248
Nodo 2	52.5.11.241

## 6. Instalación de Ansible

La instalación de [Ansible](#) la vamos a realizar únicamente en el **nodo principal** y este nodo se conectará al resto de nodos por SSH para realizar las tareas de administración.

Una de las ventajas de Ansible es que **no es necesario instalar ningún agente** en los nodos que queremos gestionar, el único requisito es que el nodo principal tenga acceso a los nodos a través de SSH.

En primer lugar deberemos conectarnos al nodo principal y realizar la instalación de [Ansible](#). El sistema operativo que de nuestra máquina es Ubuntu, por lo tanto la instalación la realizaremos con `apt`.

```
$ sudo apt update
$ sudo apt install ansible -y
```

Para instalar Ansible en otro sistema operativo se recomienda consultar la [documentación oficial](#).

## 7. Instalación de herramientas adicionales

De forma opcional podemos instalar algunas herramientas adicionales que nos ayuden a trabajar con Ansible.

### 7.1. Instalación y configuración de la extensión Ansible para Visual Studio Code

Red Hat ha desarrollado un plugin de Ansible para Visual Studio Code. Algunas de las características que ofrece este plugin son resaltado sintáctico, validación y autocompletado, entre otras.

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

### 7.2. Instalación de la herramienta `ansible-lint`

Esta herramienta nos permite detectar posibles errores en nuestros playbooks.

```
pip3 install "ansible-lint"
```

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

## 8. Configuración del archivo de inventario de Ansible

De momento la única configuración que vamos a realizar en Ansible será editar el **archivo de inventario** del **nodo principal**, para incluir la lista de hosts sobre los que vamos a realizar tareas con Ansible.

El archivo de inventario que utiliza Ansible por defecto está en la ruta `/etc/ansible/hosts`.

En los primeros ejemplos de este tutorial vamos a utilizar el archivo de inventario `/etc/ansible/hosts`, pero más adelante utilizaremos un archivo de inventario para ejemplo.

El contenido del archivo de inventario `/etc/ansible/hosts` será el siguiente:

```
[aws] ①  
18.206.58.248  
52.5.11.241
```

① `aws` es el nombre del grupo de hosts que hemos creado. Este archivo solo tiene un grupo de hosts, pero podemos crear tantos grupos como sean necesarios.

El archivo de inventario también se puede crear en otro directorio distinto a `/etc/ansible/hosts`, de

hecho es habitual tener un archivo de inventario dentro de cada proyecto. En este caso, cuando vayamos a utilizar los comandos de [Ansible](#) deberemos indicar el directorio donde se encuentra el archivo de inventario con el parámetro `-i`.

### Ejemplo:

En este ejemplo estamos utilizando un archivo de inventario que se llama `inventario` y se encuentra en el mismo directorio donde se está ejecutando el comando.

```
ansible-playbook -i inventario install_lamp.yml
```

## 9. Conexión SSH con las instancias EC2 de AWS

Para que el nodo principal pueda conectarse con las instancias EC2 de AWS vamos a necesitar una clave SSH privada que nos habrá proporcionado AWS. En nuestro caso como estamos utilizando **Amazon Academy** vamos a utilizar la clave SSH que se nos asigna por defecto en el entorno de laboratorio. El archivo con la clave privada SSH se llama `vockey.pem`.

La clave SSH debe tener permisos de sólo lectura para el propietario del archivo, por lo que será necesario aplicar el siguiente comando sobre el archivo que contiene la clave.

```
$ chmod 400 vockey.pem
```

Una vez aplicados los permisos podemos conectarnos a cualquier instancia de AWS utilizando la clave SSH y el usuario adecuado. El nombre de usuario en nuestro caso es `ubuntu` porque es el usuario que utiliza la AMI que hemos elegido para crear las instancias.

Antes de ejecutar algún comando de Ansible podemos hacer alguna prueba de conexión SSH desde el nodo principal a las instancias. Podemos conectar con el nodo 1 con el siguiente comando:

```
$ ssh -i vockey.pem ubuntu@18.206.58.248
```

Para conectarnos con el nodo 2 podemos ejecutar el siguiente comando:

```
$ ssh -i vockey.pem ubuntu@52.5.11.241
```

## 10. Módulos de Ansible

[Ansible](#) dispone de una gran variedad de módulos que pueden ser utilizados desde la línea de comandos o en las tareas de los *playbooks*.

Existen módulos para trabajar con *clouds* (Amazon, Azure, etc.), *clustering* (Kubernetes, Openshift,

etc.), bases de datos (Influxdb, Mongodb, MySQL, PostgreSQL, etc.), monitorización, mensajería, etc.

En la documentación oficial puede encontrar un listado de [todos los módulos disponibles en Ansible](#).

A continuación vamos a realizar una breve demostración de cómo utilizar algunos módulos con los comandos *ad-hoc* de Ansible:

- `ping`
- `shell`
- `command`
- `apt`

## 10.1. Módulo `ping`

El módulo `ping` nos permite realizar un `ping` sobre todos los nodos del inventario o sobre algún nodo específico.



Para indicar a [Ansible](#) el módulo que queremos utilizar usamos el modificador `-m`.

Para realizar hacer un `ping` sobre todos los nodos utilizaríamos el modificador `all`.

```
ansible all -m ping
```

Si sólo queremos hacer `ping` sobre uno de los nodos en lugar de usar `all`, indicaremos el nodo sobre el que queremos realizar la operación. Por ejemplo, en este caso realizaríamos un `ping` únicamente sobre el nodo que tiene la dirección IP `18.206.58.248`.

```
ansible 18.206.58.248 -m ping
```

## 10.2. Módulo `shell`

El módulo `shell` nos permite ejecutar comando a través de una shell sobre cada uno de los nodos.



En este caso el modificador `-m` nos permite indicar el módulo que queremos utilizar y el modificador `-a` nos permite indicar el comando.

El siguiente ejemplo ejecuta el comando `uname -a` sobre todos los nodos del inventario.

```
ansible all -m shell -a "uname -a"
```

Si no indicamos ningún módulo, por defecto [Ansible](#) utilizará el módulo `shell`. Por lo tanto, el siguiente ejemplo realizaría la misma acción que el comando anterior.

```
ansible all -a "uname -a"
```

#### Referencia:

- [Documentación oficial del módulo shell](#)

## 10.3. Módulo **command**

El módulo **command** también nos permite ejecutar comandos sobre cada uno de los nodos, pero a diferencia del módulo **shell**, los comandos no se ejecutan a través de una shell. Por lo tanto, no podremos hacer uso de variables de entorno ni de las operaciones: `<`, `>`, `|`, `;` y `&`.

#### Referencia:

- [Documentación oficial del módulo command](#)

## 10.4. Módulo **apt**

El módulo **apt** nos permite utilizar el sistema de gestión de paquetes APT para los sistemas operativos Debian y Ubuntu.

En la [documentación oficial](#) tenemos disponible todos los parámetros que podemos usar en este módulo.

Por ejemplo, el parámetro **update\_cache** nos permite realizar la operación **apt-get update**. Si quisiéramos realizar un **apt-get update** sobre cada uno de los nodos ejecutaríamos el siguiente comando.

```
ansible all -m apt -a "update_cache=yes" -b
```



El modificador **-b** es para indicar que queremos realizar un escalado de privilegios (**become**) para poder ejecutar comandos como root.

Para poder realizar la instalación de un paquete será necesario hacer uso de los parámetros **name** y **state**.

- El parámetro **name** nos permite indicar el nombre del paquete que queremos instalar.
- El parámetro **state** nos permite indicar en qué estado queremos que se encuentre el paquete. En este caso vamos a indicar **present**, para indicar que queremos que esté instalado.

Ejemplo de cómo instalar **git** en cada uno de los nodos.

```
ansible all -m apt -a "name=git state=present" -b
```

#### Referencia:

- [Documentación oficial del módulo apt](#)

# 11. ¿Qué es un **Playbook**?

Un *Playbook* es un archivo escrito en [YAML](#) donde se describen las tareas de configuración y administración que queremos realizar en cada uno de los nodos.

Un *Playbook* está formado por uno a varios *Plays* y un *Play* está formado por un conjunto de operaciones o *Tasks* que vamos a realizar en los nodos.

## 11.1. Ejemplo de instalación del servidor web **apache**

En primer lugar vamos a crear un *playbook* en el nodo principal. El nombre del archivo del archivo será `apache.yml` y las tareas que vamos a realizar en él serán la instalación del servidor web Apache y la activación del módulo `rewrite`.

El contenido del archivo de ejemplo `apache.yml` se describe a continuación.



Tenga en cuenta que los archivos YAML no permiten el uso de caracteres de tabulación para indentar. Se recomienda el uso de espacios.

```
--- ①
- hosts: aws ②
  become: yes ③
  tasks: ④
    - name: Install apache2 ⑤
      apt:
        name: apache2
        update_cache: yes
        state: latest ⑥

    - name: Enable mod_rewrite ⑦
      apache2_module:
        name: rewrite
        state: present ⑧
      notify: ⑨
        - Restart apache2

  handlers: ⑩
    - name: Restart apache2 ⑪
      service:
        name: apache2
        state: restarted
```

① Los archivos YAML empiezan con tres guiones.

② En la sección `hosts` indicamos sobre qué grupo de nuestro inventario vamos a realizar las tareas.

③ La directiva `become` se utiliza para indicar que es necesario realizar un escalado de privilegios,

por lo tanto, las tareas de este *play* se ejecutarán como *root*.

- ④ Cada play contiene una lista de *tasks*. En este caso tenemos dos *tasks*.
- ⑤ El nombre de la primera *task* es *Install apache2*.
- ⑥ Con el módulo *apt* vamos a realizar una operación de *apt update* y la instalación de la última versión del paquete *apache2*.
- ⑦ El nombre de la segunda *task* es *Enable mod\_rewrite*.
- ⑧ Con el módulo *apache2\_module* indicamos que queremos habilitar el módulo *rewrite* de Apache.
- ⑨ Las acciones que se indican en la sección de *notify* se ejecutan una vez que se han finalizado con éxito todas las acciones del bloque. En este caso, si el módulo *rewrite* de Apache se puede habilitar con éxito entonces se ejecutará el handler *Restart apache2*.
- ⑩ Los *handlers* son una lista de *tasks* que sólo se ejecutarán si son activados explícitamente desde una sección *notify*.
- ⑪ Utilizamos el módulo *service* para indicar el estado que queremos tener en el servicio *apache2*. El estado *restarted* hará que se reinicie el servicio.

## 11.2. Ejecución de un *Playbook*

Para ejecutar las tareas que hemos definido en nuestro archivo *Playbook* usamos el siguiente comando:

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

Si no ocurre ningún error deberíamos tener una respuesta similar a esta.

```
PLAY [aws] *****

TASK [Gathering Facts] *****
ok: [52.5.11.241]
ok: [18.206.58.248]

TASK [Install apache2] *****
ok: [18.206.58.248]
ok: [52.5.11.241]

TASK [Enable mod_rewrite] *****
ok: [18.206.58.248]
ok: [52.5.11.241]

PLAY RECAP *****
18.206.58.248      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
52.5.11.241      : ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Una vez que ha finalizado la ejecución de nuestro *Playbook* tendríamos el servidor web Apache instalado sobre cada uno de los nodos del grupo *aws* con el módulo *rewrite* de Apache habilitado.

## Cómo indicar el usuario y la clave privada SSH

### Desde la línea de comandos

Podemos utilizar los parámetros `--user` y `--private-key`.

Ejemplo:

```
ansible-playbook -i inventario install_lamp.yaml --user ubuntu --private-key /home/josejuan/Lab/vockey.pem
```

### Desde el archivo de inventario

Podemos utilizar las variables `ansible_user` y `ansible_ssh_private_key_file`.

Ejemplo:

```
[aws]
34.226.122.155
44.206.245.114

[all:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=/home/josejuan/Lab/vockey.pem
```

## Cómo gestionar la validación del *fingerprint* de la clave pública SSH

Al conectar por SSH por primera vez con una instancia tendremos que aceptar el *fingerprint* de la clave SSH pública de la instancia remota.

Si en el archivo de inventario tiene varias instancias, al ejecutar nuestro playbook tendremos un error porque no podemos aceptar el *fingerprint* de todas las instancias.

En el siguiente ejemplo se muestra lo que ocurre cuando ejecutamos un playbook sobre un archivo de inventario que tiene dos instancias. En este ejemplo, solo podemos aceptar el *fingerprint* de la última instancia del inventario.

```
TASK [Gathering Facts]
*****
The authenticity of host '34.226.122.155 (34.226.122.155)' can't be established.
ECDSA key fingerprint is SHA256:AjA3b6U0HwxjtJ3Phhelcp10f0u5xoY8fiEuSATMAJg.

The authenticity of host '44.206.245.114 (44.206.245.114)' can't be established.
ECDSA key fingerprint is SHA256:FJS7iV7GpDIuZPykQ9VQfGRj810dfLU1FiZVYJ+gIjI.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

Existen varias soluciones para solucionar este problema, veamos algunas.

## Configurar la variable de entorno `ANSIBLE_HOST_KEY_CHECKING`

La primera solución es configurar la variable de entorno `ANSIBLE_HOST_KEY_CHECKING` a `False` para que no saltarnos el paso de aceptar el *fingerprint* de las instancias remotas.

Para ejecutar nuestro playbook realizaríamos los siguientes pasos:

```
export ANSIBLE_HOST_KEY_CHECKING=False
```

```
ansible-playbook -i inventario install_lamp.yaml --user ubuntu --private-key /home/josejuan/Lab/vockey.pem
```

## Configurar SSH para que acepte por defecto el *fingerprint* de las nuevas instancias

Podemos utilizar un parámetro en la conexión SSH para aceptar por defecto el *fingerprint* de las nuevas instancias a las que vamos a conectarnos.

### Desde la línea de comandos

Si lo hacemos desde la línea de comandos tendríamos que añadir el siguiente parámetro `--ssh-common-args '-o StrictHostKeyChecking=accept-new'` al comando de ejecución del playbook.

Por ejemplo:

```
ansible-playbook -i inventario install_lamp.yaml --user ubuntu --private-key /home/josejuan/Lab/vockey.pem --ssh-common-args '-o StrictHostKeyChecking=accept-new'
```

### Desde el archivo de inventario

En el archivo de inventario podemos utilizar el parámetro `ansible_ssh_common_args` para configurar SSH y aceptar por defecto el *fingerprint* de las nuevas instancias a las que vamos a conectarnos.

```
[aws]
34.226.122.155
44.206.245.114

[all:vars]
ansible_user=ubuntu
ansible_ssh_private_key_file=/home/josejuan/Lab/vockey.pem
ansible_ssh_common_args='-o StrictHostKeyChecking=accept-new'
```

## 12. Ejemplos

Todos los ejemplos que se muestran a continuación están disponibles en el siguiente repositorio de GitHub:

- <https://github.com/josejuansanchez/ansible-playground>

Los ejemplos muestran diferentes *playbooks* de Ansible que hacen uso de los siguientes módulos:

Módulo	Descripción
shell	Módulo para ejecutar comandos.
command	Módulo para ejecutar comandos.
apt	Módulo para utilizar el gestor de paquetes <code>apt</code> .
service	Módulo para gestionar los servicios.
copy	Módulo para copiar archivos.
pip	Módulo para instalar paquetes de Python.
mysql_db	Módulo para gestionar las bases de datos de MySQL.
mysql_user	Módulo para gestionar los usuarios de MySQL.
get_url	Módulo para descargar archivos desde una URL.
unarchive	Módulo para descomprimir archivos.
file	Módulo para gestionar archivos y directorios.
template	Módulo para gestionar plantillas.
git	Módulo para gestionar repositorios Git.
replace	Módulo para reemplazar cadenas de texto en archivos.

## 12.1. Ejemplo 1

Este ejemplo muestra la **instalación de la pila LAMP** utilizando el módulo `command`.

**Archivo *Playbook*: `install_lamp.yml`**

```
---
- name: Playbook para instalar la pila LAMP
  hosts: aws
  become: yes

  tasks:
    - name: Actualizar los repositorios
      command: apt update

    - name: Instalar el servidor web Apache
      command: apt install apache2 -y

    - name: Instalar el sistema gestor de bases de datos MySQL
      command: apt install mysql-server -y
```

- **name:** Instalar PHP  
**command:** apt install php libapache2-mod-php php-mysql -y
- **name:** Reiniciar el servicio de Apache  
**command:** systemctl restart apache2



Recuerde que la diferencia que existe entre los módulos **command** y **shell** es que en el módulo **command** los comandos no se ejecutan a través de una shell, por lo tanto, no podremos hacer uso de variables de entorno ni de las operaciones: **<**, **>**, **|**, **;** y **&**.

[Ver en GitHub](#)

## 12.2. Ejemplo 2

Este ejemplo muestra la **instalación de la pila LAMP** utilizando el módulo **shell**.

**Archivo *Playbook*: `install_lamp.yml`**

```
---  
- name: Playbook para instalar la pila LAMP  
  hosts: aws  
  become: yes  
  
  tasks:  
  
    - name: Actualizar los repositorios  
      shell: apt update  
  
    - name: Instalar el servidor web Apache  
      shell: apt install apache2 -y  
  
    - name: Instalar el sistema gestor de bases de datos MySQL  
      shell: apt install mysql-server -y  
  
    - name: Instalar PHP  
      shell: apt install php libapache2-mod-php php-mysql -y  
  
    - name: Reiniciar el servicio de Apache  
      shell: systemctl restart apache2
```



Las buenas prácticas de Ansible recomiendan evitar el uso de los módulos **command** y **shell** en el caso de que existan módulos específicos que permitan realizar la misma tarea que el comando.

[Ver en GitHub](#)

## 12.3. Ejemplo 3

Este ejemplo muestra la **instalación de la pila LAMP** utilizando los módulos `apt`, `service` y `copy`.

Este playbook copia dos archivos en los hosts de destino:

- Un archivo de configuración de Apache que está en la ruta local `apache/000-default.conf`.
- Un archivo PHP para verificar que PHP funciona correctamente, que está en la ruta `php/index.php`.

### Archivo *Playbook*: `install_lamp.yml`

```
---
- name: Playbook para instalar la pila LAMP
  hosts: aws
  become: yes

  tasks:

    - name: Actualizar los repositorios
      apt:
        update_cache: yes

    - name: Instalar el servidor web Apache
      apt:
        name: apache2
        state: present

    - name: Instalar el sistema gestor de bases de datos MySQL
      apt:
        name: mysql-server
        state: present

    - name: Instalar PHP y los módulos necesarios
      apt:
        name:
          - php
          - php-mysql
          - libapache2-mod-php
        state: present

    - name: Copiar el archivo de configuración de Apache
      copy:
        src: apache/000-default.conf
        dest: /etc/apache2/sites-available/

    - name: Reiniciar el servidor web Apache
      service:
        name: apache2
```

```
state: restarted

- name: Copiar el archivo index.php
  copy:
    src: php/index.php
    dest: /var/www/html/
    mode: 0755
    owner: www-data
    group: www-data
```

[Ver en GitHub](#)

## 12.4. Ejemplo 4

Este ejemplo muestra cómo instalar la pila LAMP haciendo uso de variables definidas en un playbook.

**Archivo *Playbook*: `install_lamp.yml`**

```
---
- name: Playbook para instalar la pila LAMP
  hosts: aws
  become: yes

  # Variables definidas en el playbook
  vars:
    php_packages:
      - php
      - php-mysql
      - libapache2-mod-php

  tasks:

    - name: Actualizar los repositorios
      apt:
        update_cache: yes

    - name: Instalar el servidor web Apache
      apt:
        name: apache2
        state: present

    - name: Instalar el sistema gestor de bases de datos MySQL
      apt:
        name: mysql-server
        state: present

    - name: Instalar PHP y los módulos necesarios
      apt:
```

```
name: "{{ php_packages }}"
state: present
```

- **name:** Reiniciar el servidor web Apache
- service:**
  - name:** apache2
  - state:** restarted

[Ver en GitHub](#)

## 12.5. Ejemplo 5

Este ejemplo muestra cómo instalar la pila LAMP haciendo uso de variables definidas en un archivo externo.

### Archivo de variables: `vars/variables.yml`

```
php_packages:
- php
- php-mysql
- libapache2-mod-php
```

### Archivo *Playbook*: `install_lamp.yml`

```
---
- name: Playbook para instalar la pila LAMP
  hosts: aws
  become: yes

  # Variables definidas en un archivo externo
  vars_files:
    - vars/variables.yml

  tasks:

    - name: Actualizar los repositorios
      apt:
        update_cache: yes

    - name: Instalar el servidor web Apache
      apt:
        name: apache2
        state: present

    - name: Instalar el sistema gestor de bases de datos MySQL
      apt:
        name: mysql-server
```

```

    state: present

- name: Instalar PHP y los módulos necesarios
  apt:
    name: "{{ php_packages }}"
    state: present

- name: Reiniciar el servidor web Apache
  service:
    name: apache2
    state: restarted

```

[Ver en GitHub](#)

## 12.6. Ejemplo 6

Este ejemplo muestra el uso de **handlers** en un playbook.

**Archivo *Playbook*: `install_lamp.yml`**

```

---
- name: Playbook para instalar la pila LAMP
  hosts: aws
  become: yes

  tasks:
    - name: Actualizar los repositorios
      apt:
        update_cache: yes

    - name: Instalar el servidor web Apache
      apt:
        name: apache2
        state: present

    - name: Instalar el sistema gestor de bases de datos MySQL
      apt:
        name: mysql-server
        state: present

    - name: Instalar PHP y los módulos necesarios
      apt:
        name:
          - php
          - php-mysql
          - libapache2-mod-php
        state: present
      notify:
        - Reiniciar el servidor web Apache

```

```
handlers:
- name: Reiniciar el servidor web Apache
  service:
    name: apache2
    state: restarted
```

[Ver en GitHub](#)

## 12.7. Ejemplo 7

Este ejemplo muestra un playbook sobre cómo configurar MySQL Server haciendo uso de los módulos `mysql_db` y `mysql_user`.

### Archivo *Playbook*: `config_mysql.yml`

```
---
- name: Playbook para configurar MySQL
  hosts: aws
  become: yes

  vars:
    db:
      name: db_name
      user: db_user
      password: db_password

  tasks:

    - name: Instalamos el módulo de pymysql
      apt:
        name: python3-pymysql
        state: present

    - name: Crear una base de datos
      mysql_db:
        name: "{{ db.name }}"
        state: present
        login_unix_socket: /var/run/mysqld/mysqld.sock

# El parámetro 'no_log: true' lo utilizamos para ocultar información de salida
- name: Crear el usuario de la base de datos
  no_log: true
  mysql_user:
    name: "{{ db.user }}"
    password: "{{ db.password }}"
    priv: "{{ db.name }}.*:ALL"
    host: "%"
    state: present
```

```
login_unix_socket: /var/run/mysqld/mysqld.sock
```

[Ver en GitHub](#)

## 12.8. Ejemplo 8

Este ejemplo muestra la cómo hacer el despliegue de phpMyAdmin.

Archivo *Playbook*: `install_tools.yml`

```
---
- name: Playbook para instalar herramientas adicionales
  hosts: aws
  become: yes

  vars:
    - phpmyadmin_user: pma_user
    - phpmyadmin_password: pma_password
    - phpmyadmin_db_name: db_name

  tasks:

    - name: Descargar phpMyAdmin
      get_url:
        url: https://files.phpmyadmin.net/phpMyAdmin/5.2.0/phpMyAdmin-5.2.0-all-languages.zip
        dest: /tmp/phpMyAdmin-5.2.0-all-languages.zip

    - name: Instalar unzip
      apt:
        name: unzip
        state: present

    - name: Descomprimir phpMyAdmin
      unarchive:
        src: /tmp/phpMyAdmin-5.2.0-all-languages.zip
        dest: /tmp/
        remote_src: yes

    - name: Copiar phpMyAdmin
      copy:
        src: /tmp/phpMyAdmin-5.2.0-all-languages/
        dest: /var/www/html/phpmyadmin
        remote_src: yes

    - name: Cambiar el propietario y el grupo de phpMyAdmin
      file:
        path: /var/www/html/phpmyadmin
        state: directory
        owner: www-data
        group: www-data
        recurse: yes
```

```

- name: Instalar PyMySQL
  apt:
    name: python3-pymysql
    state: present

- name: Crear la base de datos para phpMyAdmin
  mysql_db:
    name: "{{ phpmyadmin_db_name }}"
    state: present
    login_unix_socket: /var/run/mysqld/mysqld.sock

- name: Importar la base de datos de phpMyAdmin
  mysql_db:
    name: "{{ phpmyadmin_db_name }}"
    state: import
    target: /var/www/html/phpmyadmin/sql/create_tables.sql
    login_unix_socket: /var/run/mysqld/mysqld.sock

- name: Crear el usuario de phpMyAdmin
  mysql_user:
    name: "{{ phpmyadmin_user }}"
    password: "{{ phpmyadmin_password }}"
    priv: "{{ phpmyadmin_db_name }}.*:ALL"
    #host: '%'
    state: present
    login_unix_socket: /var/run/mysqld/mysqld.sock

```

[Ver en GitHub](#)

## 12.9. Ejemplo 9

Este ejemplo muestra cómo realizar el despliegue de una aplicación LAMP sencilla.

### Archivo *Playbook*: `main.yml`

```

---
- import_playbook: playbooks/install_lamp.yml
- import_playbook: playbooks/install_tools.yml
- import_playbook: playbooks/deploy_web.yml

```

### Archivo de variables: `vars/variables.yml`

```

- db:
  name: lamp_db
  user: lamp_user
  password: lamp_password

- phpmyadmin:

```

```
user: pma_user
password: pma_password
db_name: db_name
```

## Archivo *Playbook*: `playbooks/deploy_web.yml`

```
---
- name: Playbook para hacer el deploy de una aplicación web
  hosts: aws
  become: yes

  vars_files:
    - vars/variables.yml

  tasks:

    - name: Instalamos el módulo de pymysql
      apt:
        name: python3-pymysql
        state: present

    - name: Crear una base de datos
      mysql_db:
        name: "{{ db.name }}"
        state: present
        login_unix_socket: /var/run/mysqld/mysqld.sock

    - name: Crear el usuario de la base de datos
      mysql_user:
        name: "{{ db.user }}"
        password: "{{ db.password }}"
        priv: "{{ db.name }}.*:ALL"
        #host: "%"
        state: present
        login_unix_socket: /var/run/mysqld/mysqld.sock

    - name: Descargar el código fuente de GitHub
      git:
        repo: https://github.com/josejuansanchez/iaw-practica-lamp.git
        dest: /tmp/iaw-practica-lamp
        force: yes
        update: yes

    - name: Copiar el código fuente
      copy:
        src: /tmp/iaw-practica-lamp/src/
        dest: /var/www/html
        remote_src: yes
        mode: 0755

    - name: Editar el nombre de la base de datos en el archivo de configuración
      replace:
```

```
path: /var/www/html/config.php
regexp: "database_name_here"
replace: "{{ db.name }}"
```

- **name:** Editar el usuario de la base de datos en el archivo de configuración

**replace:**

```
path: /var/www/html/config.php
regexp: "username_here"
replace: "{{ db.user }}"
```

- **name:** Editar el password del usuario de la base de datos en el archivo de configuración

**replace:**

```
path: /var/www/html/config.php
regexp: "password_here"
replace: "{{ db.password }}"
```

- **name:** Importar la base de datos

**mysql\_db:**

```
name: "{{ db.name }}"
state: import
target: /tmp/iaw-practica-lamp/db/database.sql
login_unix_socket: /var/run/mysqld/mysqld.sock
```

- **name:** Eliminar el directorio iaw-practica-lamp

**file:**

```
path: /tmp/iaw-practica-lamp
state: absent
```

- **name:** Cambiar el propietario del directorio /var/www/html

**file:**

```
path: /var/www/html
owner: www-data
group: www-data
recurse: yes
```

[Ver en GitHub](#)

## 12.10. Ejemplo 12

Ejemplo de **Infraestructura como Código (IaC)** en AWS.

Este ejemplo muestra un playbook para crear un grupo de seguridad y una instancia EC2 en AWS.

En la [documentación oficial de Ansible](#) puede encontrar toda la colección de módulos que existen para gestionar los recursos de AWS con Ansible.

**Archivo *Playbook*:** `create_sg_ec2.yml`

```
---
```

- **name:** Playbook para crear un grupo de seguridad y una instancia EC2 en AWS  
**hosts:** localhost

```

connection: local
gather_facts: false

vars:
  ec2_security_group: sg_ejemplo_12
  ec2_security_group_description: Grupo de seguridad del ejemplo 12
  ec2_instance_name: instancia_ejemplo_12
  ec2_image: ami-06878d265978313ca
  ec2_instance_type: t2.small
  ec2_key_name: vockey

tasks:
- name: Crear un grupo de seguridad
  ec2_group:
    name: "{{ ec2_security_group }}"
    description: "{{ ec2_security_group_description }}"
    rules:
      - proto: tcp
        from_port: 22
        to_port: 22
        cidr_ip: 0.0.0.0/0
      - proto: tcp
        from_port: 80
        to_port: 80
        cidr_ip: 0.0.0.0/0
      - proto: tcp
        from_port: 443
        to_port: 443
        cidr_ip: 0.0.0.0/0
    register: security_group

- name: Crear una instancia EC2
  ec2_instance:
    name: "{{ ec2_instance_name }}"
    key_name: "{{ ec2_key_name }}"
    security_group: "{{ ec2_security_group }}"
    instance_type: "{{ ec2_instance_type }}"
    image_id: "{{ ec2_image }}"
    count: 1
    state: running
    register: ec2

```

[Ver en GitHub](#)

## 12.11. Ejemplo 13

Ejemplo de **Infraestructura como Código (IaC)** en AWS.

Este ejemplo muestra dos playbooks para crear y eliminar un grupo de seguridad, una instancia EC2 y una IP elástica en AWS.

## Archivo *Playbook*: create\_sg\_ec2\_ep.yml

```
---
- name: Playbook para crear un grupo de seguridad, una instancia EC2 y una IP elástica en AWS
  hosts: localhost
  connection: local
  gather_facts: false

  vars:
    ec2_security_group: sg_ejemplo_13
    ec2_security_group_description: Grupo de seguridad del ejemplo 13
    ec2_instance_name: instancia_ejemplo_13
    ec2_image: ami-06878d265978313ca
    ec2_instance_type: t2.small
    ec2_key_name: vockey

  tasks:
    - name: Crear un grupo de seguridad
      ec2_group:
        name: "{{ ec2_security_group }}"
        description: "{{ ec2_security_group_description }}"
        rules:
          - proto: tcp
            from_port: 22
            to_port: 22
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            from_port: 80
            to_port: 80
            cidr_ip: 0.0.0.0/0
          - proto: tcp
            from_port: 443
            to_port: 443
            cidr_ip: 0.0.0.0/0
        register: security_group

    - name: Crear una instancia EC2
      ec2_instance:
        name: "{{ ec2_instance_name }}"
        key_name: "{{ ec2_key_name }}"
        security_group: "{{ ec2_security_group }}"
        instance_type: "{{ ec2_instance_type }}"
        image_id: "{{ ec2_image }}"
        count: 1
        state: running
        register: ec2

    # - name: Mostramos el contenido de la variable ec2
    #   debug:
    #     msg: "ec2: {{ ec2 }}"

    - name: Crear una nueva IP elástica y asociar a la instancia
      ec2_eip:
```

```
    device_id: "{{ ec2.instances[0].instance_id }}"
  register: eip
```

- **name:** Mostrar la IP elástica

```
  debug:
```

```
    msg: "La IP elástica es: {{ eip.public_ip }}"
```

## Archivo *Playbook*: delete\_sg\_ec2\_ep.yml

```
---
```

- **name:** Playbook para eliminar un grupo de seguridad y una instancia EC2 en AWS

```
hosts: localhost
```

```
connection: local
```

```
gather_facts: false
```

```
vars:
```

```
  ec2_security_group: sg_ejemplo_13
```

```
  ec2_instance_name: instancia_ejemplo_13
```

```
tasks:
```

- **name:** Obtener el id de instancia a partir del nombre

```
  ec2_instance_info:
```

```
    filters:
```

```
      "tag:Name": "{{ ec2_instance_name }}"
```

```
      "instance-state-name": "running"
```

```
    register: ec2
```

- **name:** Desasociar de IP elástica de la instancia EC2

```
  ec2_eip:
```

```
    device_id: "{{ ec2.instances[0].instance_id }}"
```

```
    state: absent
```

- **name:** Eliminar la instancia EC2

```
  ec2_instance:
```

```
    filters:
```

```
      "tag:Name": "{{ ec2_instance_name }}"
```

```
    state: absent
```

- **name:** Eliminar el grupo de seguridad

```
  ec2_group:
```

```
    name: "{{ ec2_security_group }}"
```

```
    state: absent
```

[Ver en GitHub](#)

# Autor

Este material ha sido desarrollado por [José Juan Sánchez Hernández](#).

# Licencia

El contenido de esta web está bajo una [licencia de Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional](#).

# Referencias

- [Videotutorial de Ansible](#). Makigas.es.
- [Primeros pasos con Ansible](#). Carlos Aparicio.
- [How to install Apache on Ansible](#).
- [How To Manage Remote Servers with Ansible eBook](#). Erika Heidi. DigitalOcean.
- [Ansible for DevOps](#). Jeff Geerling. O'Reilly Media.
- [Ansible Playbook Essentials](#). Gourav Shah. Packt Publishing.
- [Ansible Best Practices. How to write, how to execute, and how to use in real life](#). Red Hat.