

Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

Laboratorio. Instalación completa de WordPress con Ansible

El objetivo principal de este proyecto es desplegar de forma automática un entorno web de prueba con un WordPress personalizado, utilizando **Vagrant como herramienta de Infraestructura como Código (IaC) y Ansible para su aprovisionamiento automatizado.**

Simplemente ejecutando en la terminal el comando **vagrant up** en el directorio donde se encuentra el **Vagrantfile**, se despliega todo el entorno sin necesidad de realizar configuraciones adicionales.

Antes de configurar y desplegar el entorno WordPress, es necesario llevar a cabo varias tareas de aprovisionamiento y configuración, entre ellas:

1. **Preparar y configurar la máquina virtual (MV)** con un servidor web **Apache** para redirigir y servir todo el contenido.
2. **Instalar los módulos específicos de PHP** requeridos por WordPress y Mysql.
3. **Configurar una base de datos MySQL** que será utilizada por WordPress para su funcionamiento.
4. **Levantar y configurar un servidor web Nginx** que actuará **como proxy inverso**, bloqueará las peticiones dirigidas a las rutas sensibles como las de administración de Wordpress y redirigirá el tráfico deseado al servidor de Apache.

Para verificar el correcto funcionamiento, basta con acceder a <http://192.168.155.10> desde el navegador.

1. Estructura del proyecto en Ansible

A continuación, se detalla la organización de los archivos y roles de Ansible, para facilitar la comprensión del funcionamiento general del proyecto:

1. Playbook principal: provision/playbook.yml

Este archivo actúa como el punto de inicio en Ansible. Desde aquí se **incluyen los roles** necesarios para configurar todos los componentes del entorno. En este caso, el código está dividido en cuatro roles: **apache, mysql, wordpress y nginx**, que se ejecutan en este orden.

Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

La instalación de **PHP** y sus módulos se ha decidido incluir directamente en este playbook, en lugar de crear un rol separado, ya que son solo unas pocas líneas de código.

El orden de aprovisionamiento es el siguiente:

1. Módulos de PHP
2. Apache
3. MySQL
4. WordPress
5. Nginx

2. Gestión de variables con Ansible

En lugar de usar **Hiera como con Puppet**, en Ansible se utilizan archivos **YAML dentro del directorio `group_vars/all.yml`**, lo que permite separar las variables del código principal.

Esto asegura un enfoque más seguro, evitando exponer credenciales sensibles al subir el proyecto a un repositorio. Aunque este proyecto es académico y no incluye variables encriptadas, **Ansible Vault** permite cifrar variables si fuera necesario.

- Las variables se declaran en: **`group_vars/all.yml`**
- Se utilizan plantillas **Jinja2 (.j2)** para inyectar valores dinámicos en los archivos de configuración.

3. Roles utilizados en Ansible

El proyecto está dividido en cuatro roles principales, lo que garantiza modularidad y organización en el código:

◇ Rol apache

Este rol **instala y configura el servidor web Apache** en la máquina virtual, dejándolo preparado para que **WordPress pueda administrar y servir contenido** a través de él.

Algunas de sus funciones principales son:

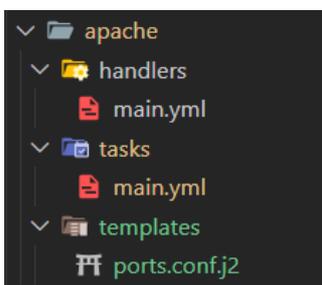
- Instalar Apache y asegurarse de que el servicio esté activo.
- **Eliminar la página por defecto** de Apache.
- **Configurar el puerto de escucha al 127.0.0.1:8080**

"Configurar el puerto de escucha en 127.0.0.1:8080 significa que Apache solo aceptará conexiones desde procesos locales en la misma máquina donde se ejecuta. La dirección

Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

127.0.0.1 es la dirección de loopback (localhost), lo que impide que se acceda a Apache desde otras máquinas en la red. Esto es útil cuando Apache está detrás de un proxy inverso, como Nginx, que gestiona las conexiones externas y reenvía las solicitudes a Apache en el puerto 8080."

- Copiar la configuración personalizada desde **una plantilla Jinja2** (wp-apache-config.conf.j2).
- Habilitar el nuevo sitio y reiniciar Apache automáticamente.

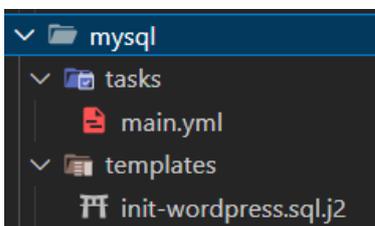


◇ Rol mysql

Este rol **instala y configura un servidor MySQL en la MV**, asegurando el correcto funcionamiento de la base de datos para WordPress.

Las principales tareas que realiza este rol son:

- Instalar el servidor **MySQL**.
- Crear la base de datos necesaria para WordPress.
- Crear el usuario y asignarle los permisos adecuados.
- Ejecutar un **script de inicialización** (init-wordpress.sql.j2) para preparar la base de datos.



◇ Rol wordpress

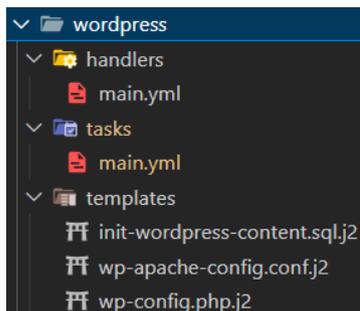
Este rol **instala y configura WordPress**, dejándolo completamente funcional.

Las principales acciones realizadas son:

- **Descargar y extraer WordPress** en /var/www/html/wordpress.

Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

- **Crear el archivo wp-config.php** basado en una plantilla (wp-config.php.j2).
- **Asegurar los permisos de WordPress** (chown -R www-data:www-data).
- **Instalar wp-cli** y usarlo para configurar WordPress automáticamente.
- **Inicializar la base de datos** con un contenido mínimo utilizando la plantilla init-wordpress-content.sql.j2.
- **Configurar Apache** para servir el contenido de WordPress.

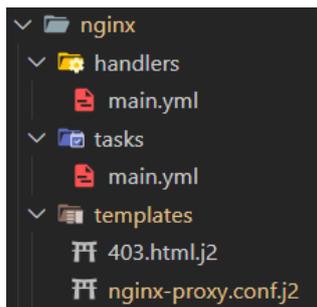


◊ Rol nginx

Este rol aprovisiona a la máquina de un servidor web **Nginx** que actúa como un **proxy inverso** y **forma una primera capa** de defensa del sistema. Bloquea las solicitudes a las urls de administración y redirige el tráfico permitido a apache servir su contenido.

Las principales acciones realizadas son:

- **Instalar Nginx** en la máquina virtual.
- **Configurar Nginx como proxy inverso**, redirigiendo las solicitudes a Apache en el puerto 8080.
Bloquear el acceso a rutas sensibles como /wp-admin y /wp-login.php para aumentar la seguridad.
- **Deshabilitar la página por defecto de Nginx** y habilitar una configuración específica para WordPress.
- **Reiniciar Nginx automáticamente** después de aplicar la configuración.



Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

1.1 Conclusión

Este proyecto implementa un despliegue automatizado de **WordPress con Ansible**, asegurando un entorno reproducible y modular.

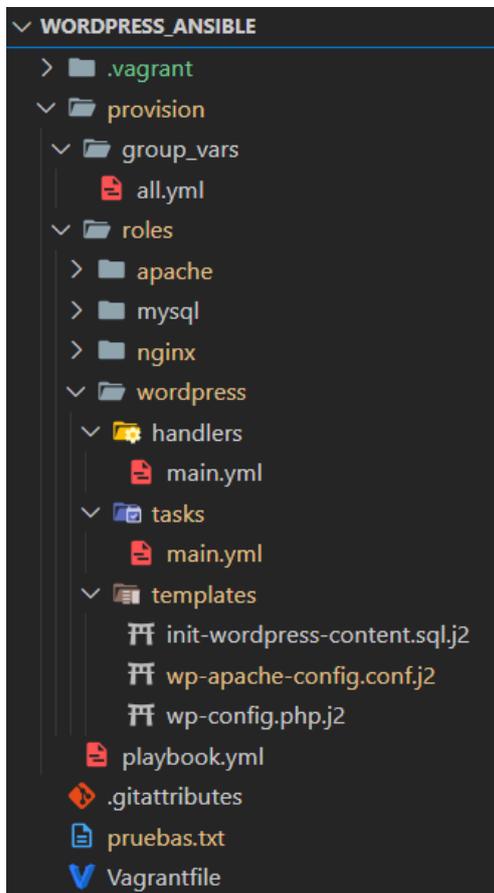
Gracias a Ansible, **se eliminan configuraciones manuales y se garantiza la coherencia en cada despliegue.**

Al ejecutar `vagrant up`, se configura automáticamente:

- **PHP**, con las librerías necesarias.
- **Apache**, para servir WordPress.
- **MySQL**, con la base de datos y el usuario configurados.
- **WordPress**, con su configuración y contenido inicial.
- **Nginx**, para bloquear el tráfico sensible

Finalmente, el acceso a la web se puede realizar a través de:

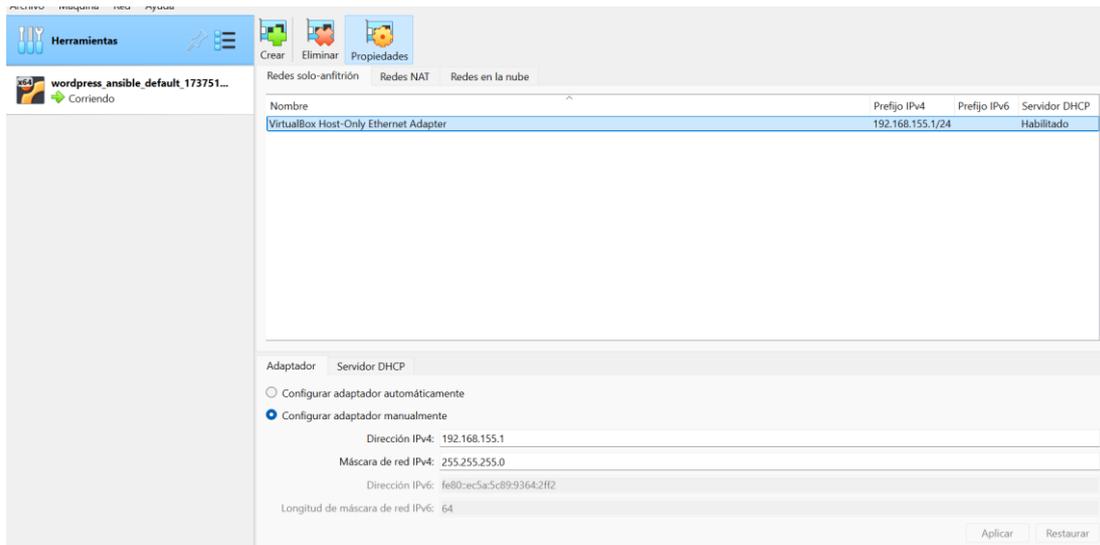
<http://192.168.155.10>



Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

2. Puntos destacables en la realización del proyecto

- Tener adaptador de red host-only con el rango de ips que se le da a la mv



- En este caso es el rango de ips 192.168.155.0/24

```
Vagrantfile
1 Vagrant.configure("2") do |config|
2
3   config.vm.box = "bento/ubuntu-22.04" # Usa la misma caja base de
4
5   # Configuración de red
6   config.vm.network "forwarded_port", guest: 80, host: 8080
7   config.vm.network "private_network", ip: "192.168.155.10"
8
```

- Indicar que la mv se va a autoprovisionar con ansible

```
# Provisionamiento de la mv con ansible
config.vm.provision "ansible_local" do |ansible|
  ansible.playbook = "provision/playbook.yml" # Define el playbook de Ansible
  ansible.extra_vars = { host_key_checking: false } # Evita problemas de clave SSH
end
end
```

- Despliegue mv con vagrant

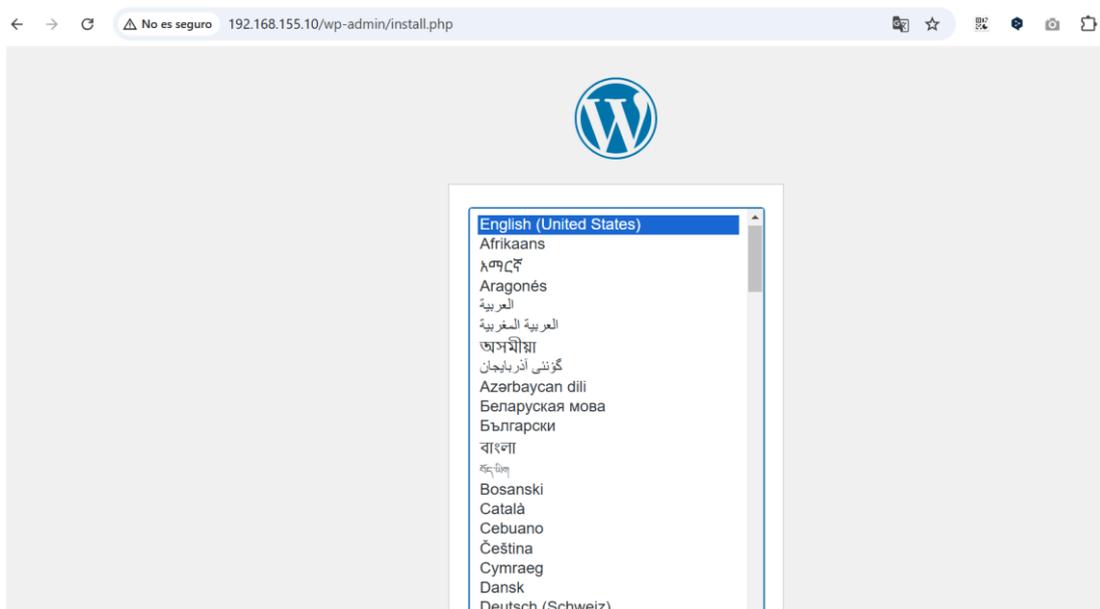
Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

```

o wordpress_ansible> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'bento/ubuntu-22.04'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'bento/ubuntu-22.04' version '202407.23.0' is up to date...
==> default: Setting the name of the VM: wordpress_ansible_default_1737517839267_45317
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
default: Adapter 2: hostonly
==> default: Forwarding ports...
default: 80 (guest) => 8080 (host) (adapter 1)
default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
default: SSH address: 127.0.0.1:2222
default: SSH username: vagrant
default: SSH auth method: private key

```

- Checkpoint del proceso: Wordpress funcionando pero sin la pagina web personalizada



2.1 Bloque extra

Como extensión opcional, añade a la configuración la instalación de un NGINX que actúe como proxy, bloqueando rutas no deseadas, por ejemplo, las de administración del servidor WordPress.

Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	

¿Cómo funciona la arquitectura Apache + Nginx?

Apache y Nginx se ejecutan al mismo tiempo, pero con funciones distintas:

Nginx recibe todas las solicitudes del usuario.

- Filtra y bloquea rutas peligrosas.
- Sirve contenido estático (CSS, JS, imágenes).
- Redirige las solicitudes dinámicas a Apache.

Apache sigue ejecutando WordPress en el backend.

- Procesa las páginas PHP y las consultas a la base de datos.
- Devuelve la respuesta a Nginx, que la envía al usuario.

La arquitectura sería así:

Usuario → Nginx (proxy inverso) → Apache (WordPress) → Nginx → Usuario

- **Nginx escucha en el puerto 80 en todas las interfaces de la máquina (0.0.0.0), aceptando conexiones desde cualquier IP. Luego, reenvía las solicitudes a la dirección de loopback (127.0.0.1:8080), donde solo Apache está escuchando. Esto significa que Apache solo es accesible desde la misma máquina, mientras que Nginx actúa como un proxy inverso gestionando las conexiones externas.**

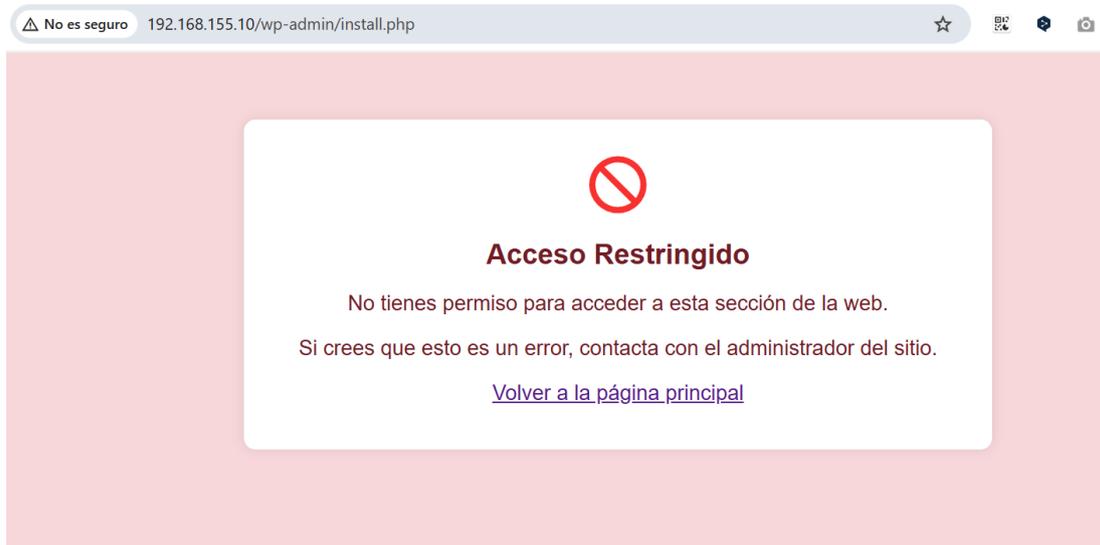
```
vagrant@vagrant:~$ sudo ss -tulnp | grep -E ':80|8080'
tcp LISTEN 0      511      127.0.0.1:8080      0.0.0.0:*      users:(("apache2",pid=4278,fd=3),("apache2",pid=4234,fd=3),("apache2",pid=4233,fd=3),("apache2",pid=4232,fd=3),("apache2",pid=4231,fd=3),("apache2",pid=4230,fd=3),("apache2",pid=4229,fd=3))
tcp LISTEN 0      511      0.0.0.0:80        0.0.0.0:*      users:(("nginx",pid=4269,fd=6),("nginx",pid=4268,fd=6),("nginx",pid=4267,fd=6))
vagrant@vagrant:~$
```

- **Acceso al sitio web personalizado de Wordpress**



- **Nginx bloquea petición a url de administración de wordpress.**

Asignatura	Datos del alumno	Fecha
Herramientas de Automatización de Despliegues	Apellidos: Inglés Martínez	23/01/2025
	Nombre: Alejandro	



```
vagrant@vagrant:~$ curl -I http://192.168.155.10
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 23 Jan 2025 06:20:28 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Link: <https://wordpress.org/latest.tar.gz/index.php?rest_route=/>; rel="https://api.w.org/"
```

ENLACE A VÍDEO DE YT (en oculto):

<https://youtu.be/lksomzUvzA0>