Desafio # 6

Fecha de entrega: 04/12/2024

Objetivo:

Este desafío tiene como objetivo implementar un playbooks de Ansible y Roles para cada servicio.
Podremos desplegar un nuevo EC2 en AWS Academy de ser necesario para utilizar una familia de Ubuntu, **importante** usar la misma PEM para facilitar el acceso.

Utilizar el README de Sandbox como ayuda para acceder a la instancia de EC2 y configurar el *inventory.ini*

Escenario:

Nuestro equipo de trabajo recientemente inició un nuevo sprint y nos asignaron una tarea muy importante para modularizar nuestro proyecto de *configuration management*, este sprint tiene como fecha de cierre el 06/12/2024 y debemos tener una versión funcional el día 04/12/204 para la reunión de demos.

Este proyecto utiliza como herramienta de *configuration as a code* Ansible y permite instalar y desplegar nuestro sitio web de Wordpress en un host EC2 que es utilizaremos a demanda en un laboratorio de AWS Academy.

Como estamos trabajando en un entorno de Demo, necesitamos instalar PHP junto con sus componentes y MySQL dentro de una instancia EC2, no es necesario para este proyecto una base de datos RDS.

Requisitos:

1. Utilizar un Bastion de AWS Academy
2. Identificar y crear posibles variables que se puedan reutilizar.

3. Crear un archivo playbook y roles específicos para PHP y MySQL
4. Testar y validar que todo funciona

Proyecto ejemplo: https://medium.com/how-to-install-wordpress-on-ubuntu-a-step-by-step-guide

Entregables:

Los entregables establecidos para este proyecto con:

1. Código fuente del playbook de Ansible publicado en un repositorio de Github.
2. Guía detallada de cómo utilizar el Role publicado en el archivo README.md del repositorio.
3. Evidencia de las pruebas con resultado exitoso.

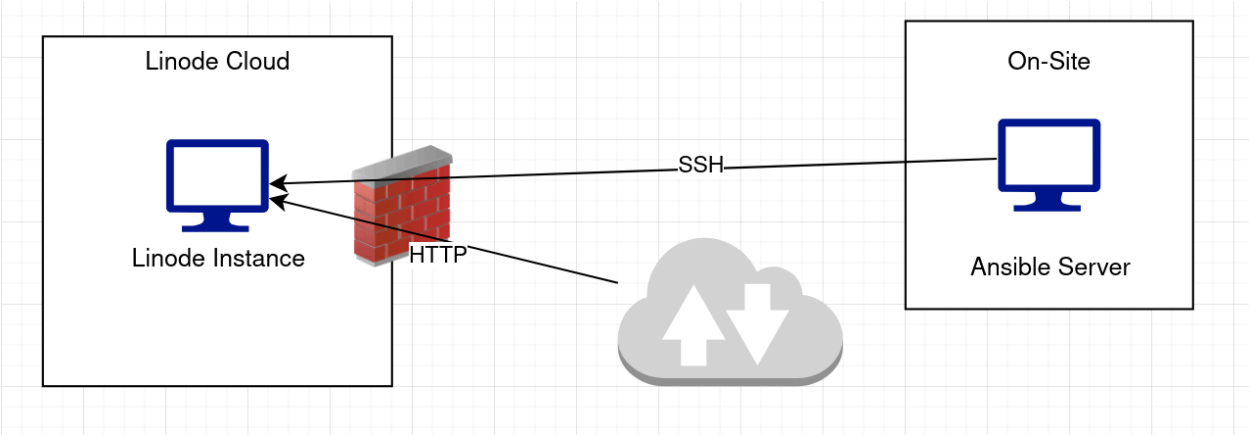
Evaluacion:

- Entrega en fecha.
- Redactar documentación legible y que sea comprendida por terceros.
- Añade material de soporte adicional.
    - Ejemplo: Diagrama de alto nivel.
- Cumple con las consignas solicitadas.
- Utilizar un archivo *playbook.yaml* y una carpeta role para separar cada servicio
- El entregable es funcional.
    - Ejemplo: repositorio con los archivos necesarios
    - Estructura de archivos (comando *tree* de Linux)

Solución:

Setup:

Usaré el siguiente esquema para la implementación:



1. Desde el servidor con Ansible voy a generar la clave SSH y copiarla a la instancia remota con:

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/ansible_key -N ""
```

2. y la copio al sistema remoto

```
ssh-copy-id -i ~/.ssh/ansible_key.pub root@172.233.27.216
```

3. Configuro el archivo de inventario para Ansible:

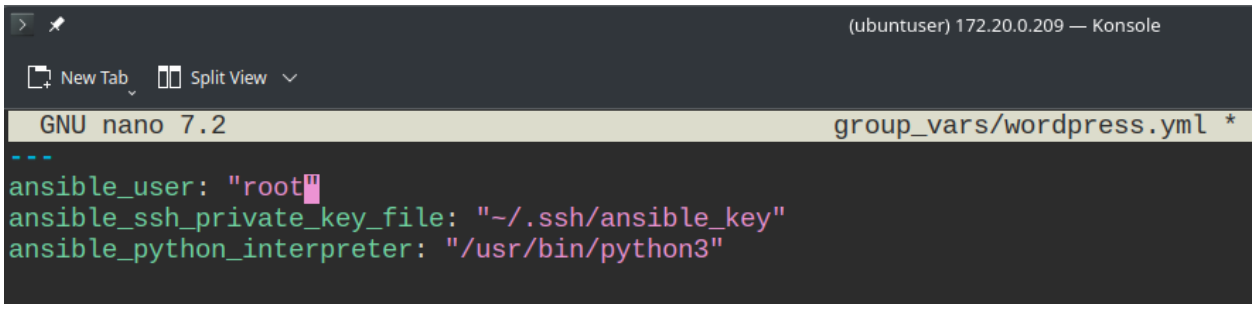
```
[wordpress]
172.233.27.216 ansible_user=root
ansible_ssh_private_key_file=~/.ssh/ansible_key
ansible_python_interpreter=/usr/bin/python3
```

4. Test de ping

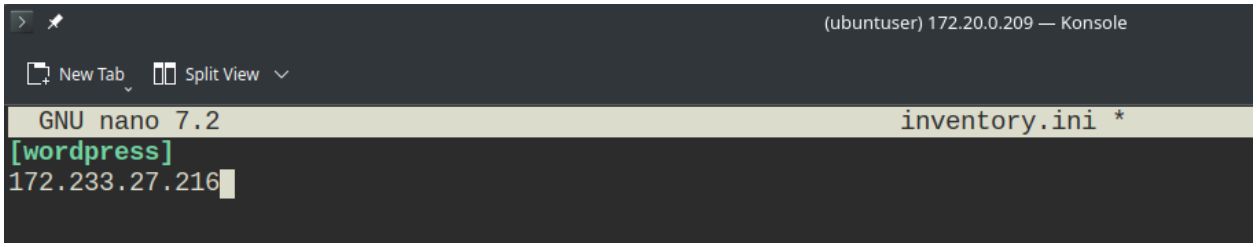
```
ubuntuser@ubuntulab:~/Desf6$ ansible -i inventory.ini wordpress -m ping
172.233.27.216 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ubuntuser@ubuntulab:~/Desf6$
```

Variables:

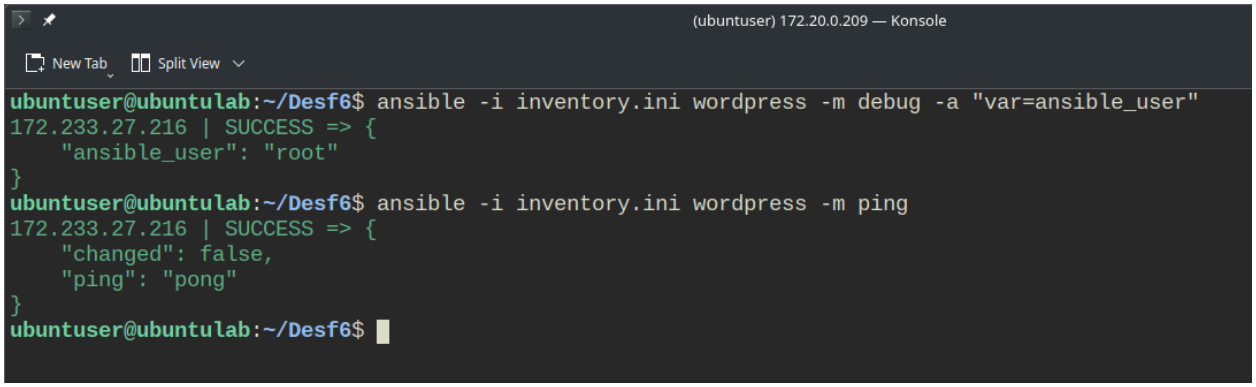
Voy a mover las variables de usuario para el grupo de wordpress a un archivo de variables, dentro de un directorio group_vars, archivo wordpress.yml (mismo nombre que el grupo en el inventario)



Y en Inventory solo queda la IP de la instancia remota:



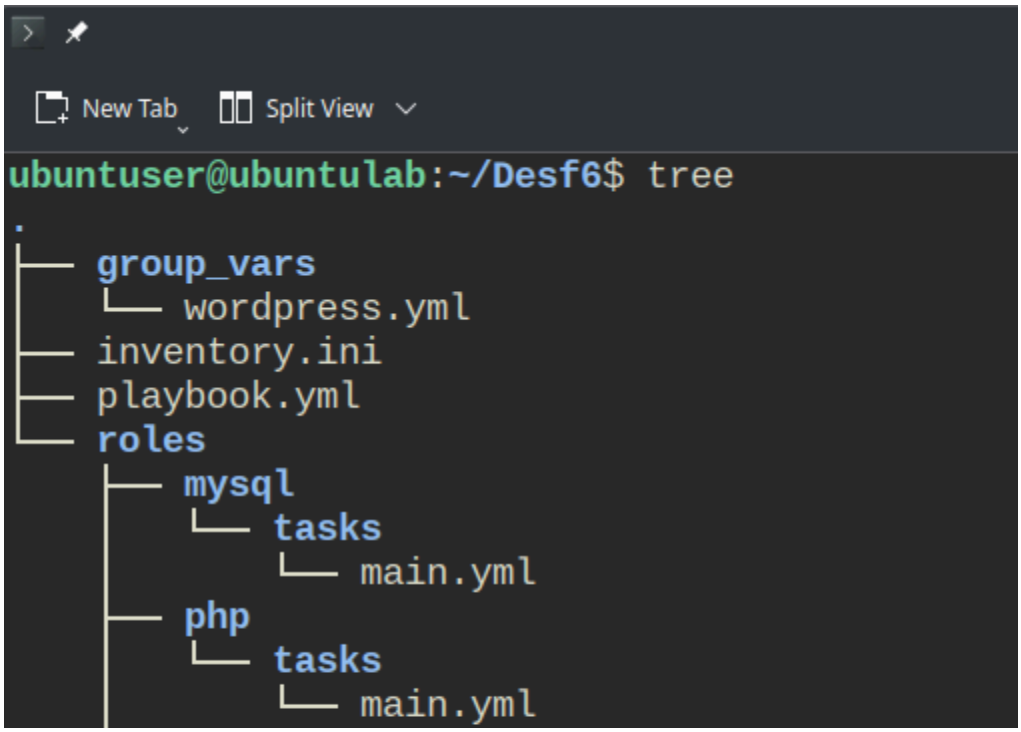
Pruebo nuevamente ping y debug



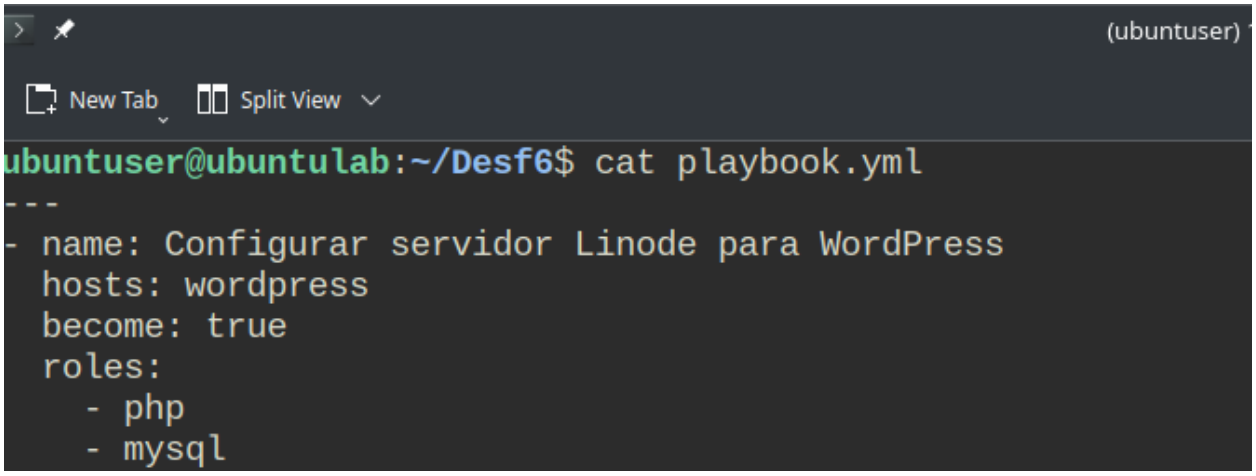
Playbooks:

Crearé la siguiente estructura de directorios para el playbook:

roles/php/ para la instalación de PHP y dependencias
roles/mysql/ para la instalación de MySQL y dependencias
playbook.yml, el playbook principal



playbook.yaml



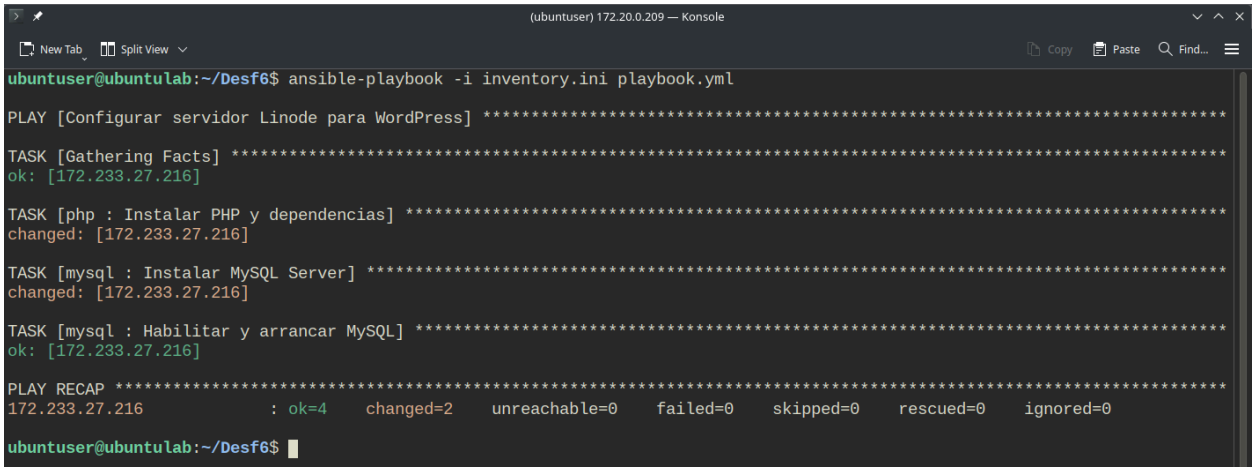
roles/php/tasks/main.yml

```
ubuntuser@ubuntulab:~/Desf6$ cat roles/php/tasks/main.yml
---
- name: Instalar PHP y dependencias
  apt:
    name:
      - php
      - php-mysql
      - php-xml
      - php-mbstring
      - php-curl
      - php-zip
      - php-gd
    state: present
    update_cache: yes
ubuntuser@ubuntulab:~/Desf6$
```

roles/mysql/tasks/main.yml

```
ubuntuser@ubuntulab:~/Desf6$ cat roles/mysql/tasks/main.yml
---
- name: Instalar MySQL Server
  apt:
    name: mysql-server
    state: present
    update_cache: yes

- name: Habilitar y arrancar MySQL
  service:
    name: mysql
    state: started
    enabled: yes
```

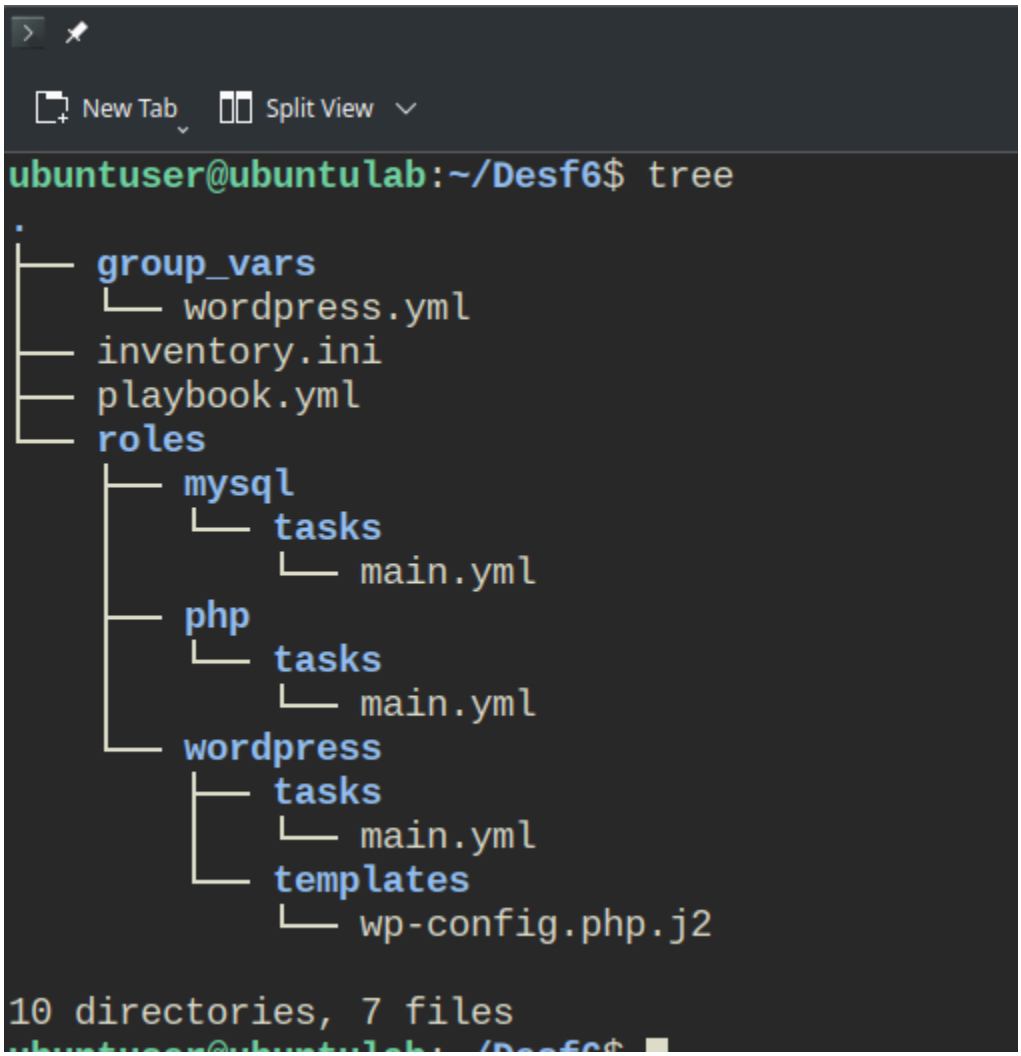
Ejecución del Playbook:



Configuración y Wordpress

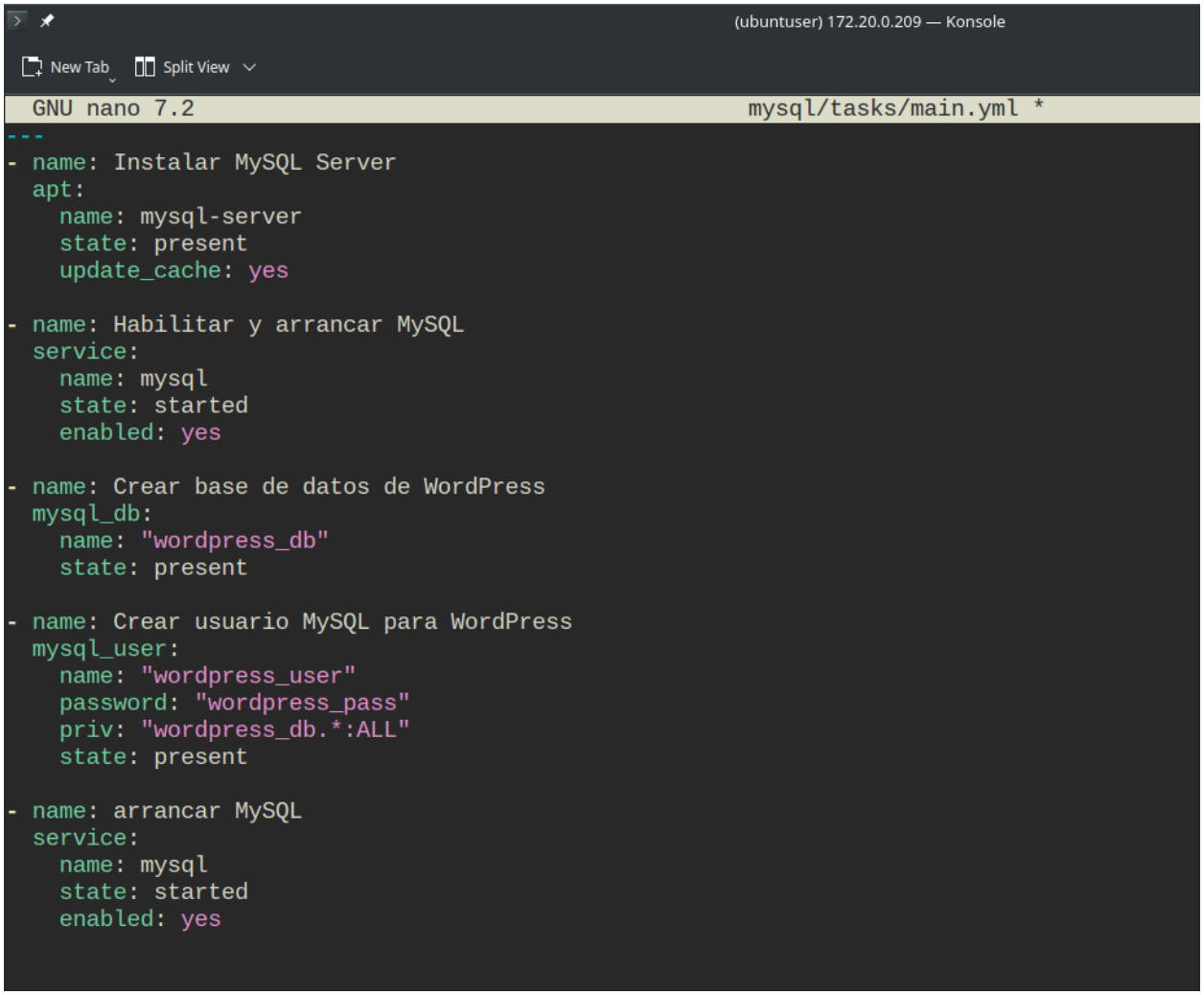
Nota: Luego de terminar la configuracion y, leyendo aqui https://www.digitalocean.com/community/tutorials/how-to-use-ansible-to-install-and-set-up-wordpress-with-lamp-on-ubuntu-18-04 me di cuenta de que hay muchas variables que se podrían "subir" al paso de definicion de variables. Esto es bueno porque hace el playbook más agnóstico y reutilizable pero no llegué a implementarlo.

Creo un nuevo rol para la instalación de wordpress, la estructura de directorios queda así:



Actualización de los roles:

mysql/tasks/main.yml



roles/wordpress/tasks/main.yml

(ubuntuser) 172.20.0.209 — Konsole

New Tab Split View

GNU nano 7.2 wordpress/tasks/main.yml *

```
---
- name: Descargar WordPress
  get_url:
    url: "https://wordpress.org/latest.tar.gz"
    dest: "/tmp/wordpress.tar.gz"

- name: Extraer WordPress
  unarchive:
    src: "/tmp/wordpress.tar.gz"
    dest: "/var/www/html/"
    remote_src: yes

- name: Configurar el archivo wp-config.php
  template:
    src: "wp-config.php.j2"
    dest: "/var/www/html/wordpress/wp-config.php"
  notify:
    - Reiniciar Apache
```

roles/wordpress/templates/wp-config.php.j2

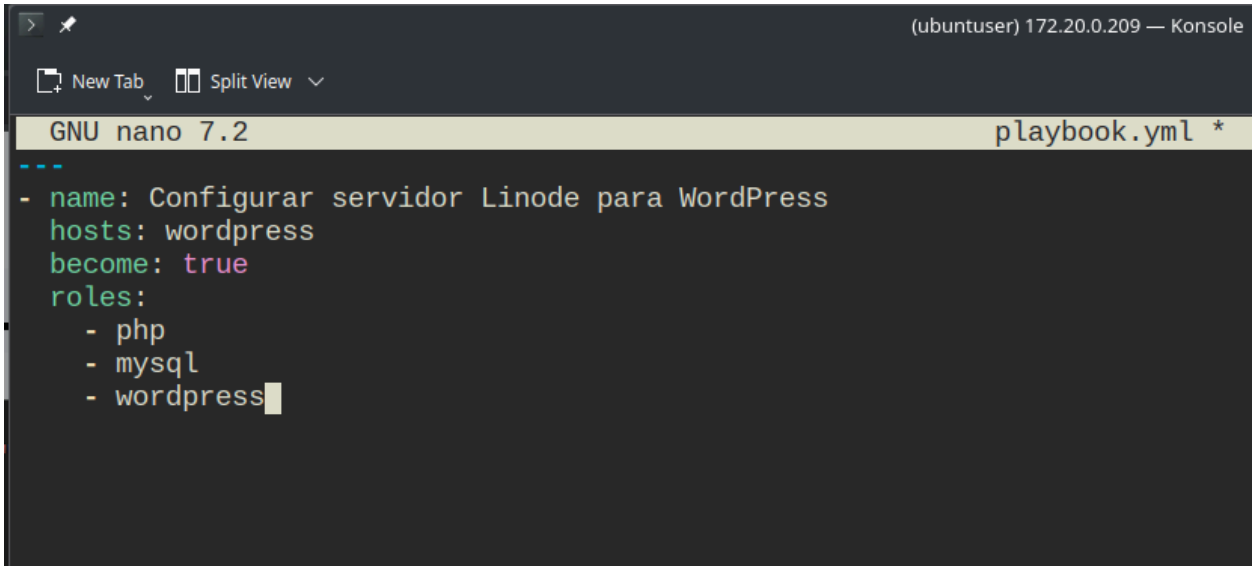
(ubuntuser) 172.20.0.209 — Konsole

New Tab Split View

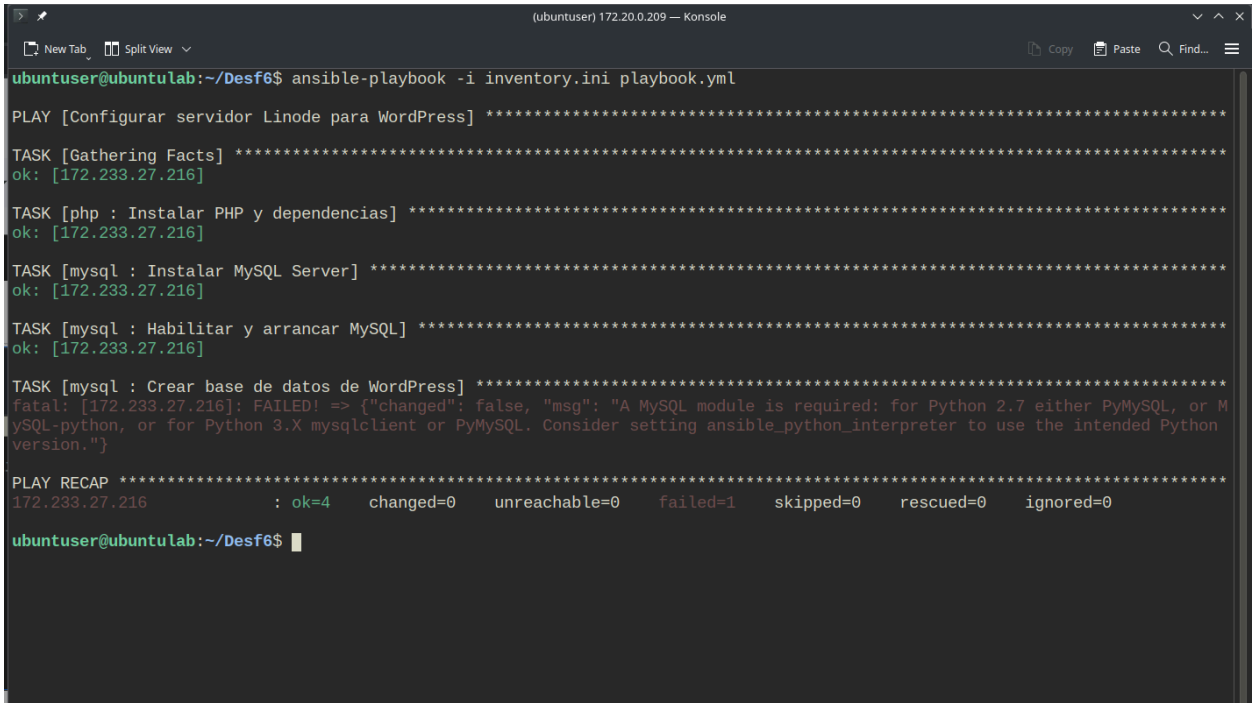
GNU nano 7.2 wordpress/templates/wp-config.php.j2

```
<?php
define( 'DB_NAME', 'wordpress_db' );
define( 'DB_USER', 'wordpress_user' );
define( 'DB_PASSWORD', 'wordpress_pass' );
define( 'DB_HOST', 'localhost' );
define( 'DB_CHARSET', 'utf8' );
define( 'DB_COLLATE', '' );

$table_prefix = 'wp_';

define( 'WP_DEBUG', false );

if ( ! defined( 'ABSPATH' ) ) {
        define( 'ABSPATH', __DIR__ . '/' );
}

require_once ABSPATH . 'wp-settings.php';
```

Y actualizo el playbook principal para incluir el rol de wordpress:

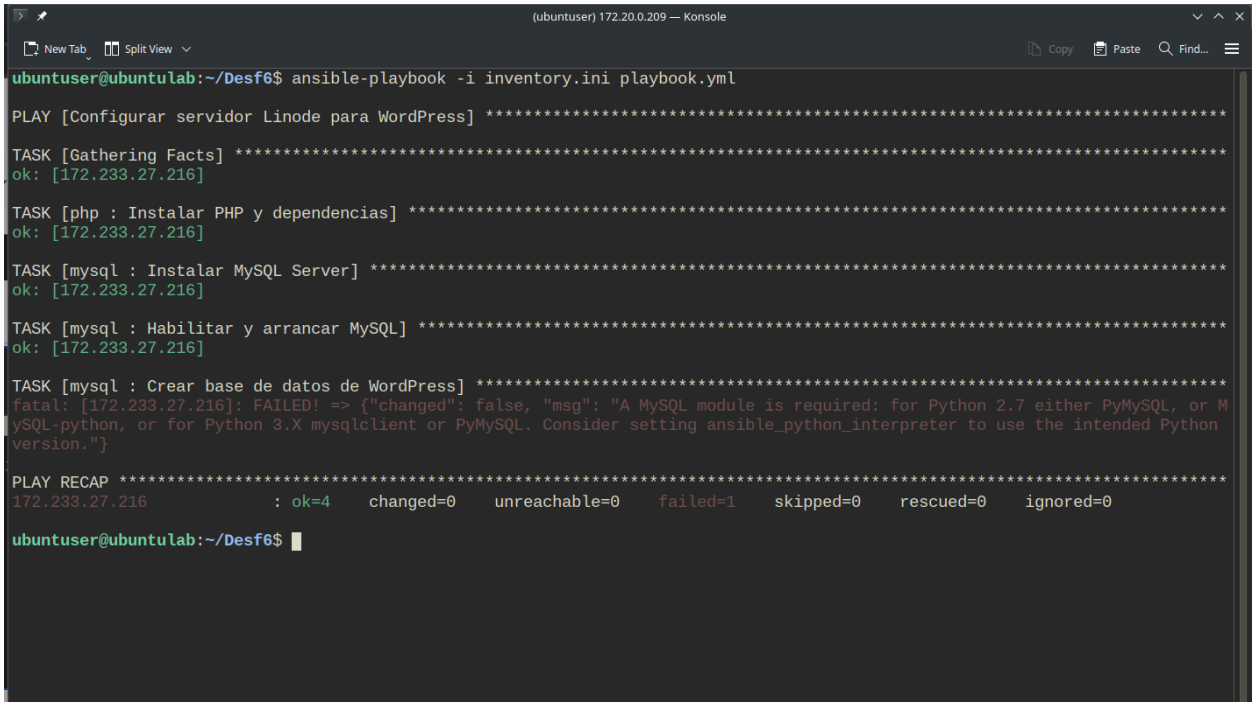


Ejecución del Playbook: (spoiler: no anduvo)



Troubleshooting

Ejecución del Playbook:




```
TASK [mysql : Crear base de datos de WordPress]
**********************************************************************
fatal: [172.233.27.216]: FAILED! => {"changed": false, "msg": "A MySQL module
is required: for Python 2.7 either PyMySQL, or MySQL-python, or for Python 3.X
mysqlclient or PyMySQL. Consider setting ansible_python_interpreter to use the
intended Python
version."}
```

Dado que la instancia remota viene con python3 instalo pip y PyMySQL

sudo apt-get install python3-pip
pip3 install PyMySQL

Lo agregué al playbook, junto con apt-update y apt-upgrade en el task de Wordpress

(ubuntuser) 172.20.0.209 — Konsole

New Tab Split View

GNU nano 7.2 main.yml *

```yaml
---
- name: apt update y upgrade
  apt:
    update_cache: yes
    upgrade: dist
    autoclean: yes
    autoremove: yes

- name: Instalar pip3
  apt:
    name: python3-pip
    state: present

- name: Instalar PyMySQL con pip3
  pip:
    name: PyMySQL
    state: present
    extra_args: --break-system-packages # Investigar por que necesita este argumento

- name: Descargar WordPress
  get_url:
    url: "https://wordpress.org/latest.tar.gz"
    dest: "/tmp/wordpress.tar.gz"

- name: Extraer WordPress
  unarchive:
    src: "/tmp/wordpress.tar.gz"
    dest: "/var/www/html/"
    remote_src: yes

- name: Configurar el archivo wp-config.php
  template:
    src: "wp-config.php.j2"
    dest: "/var/www/html/wordpress/wp-config.php"
  notify:
    - Reiniciar Apache
```

Ejecución del Playbook:

(ubuntuser) 172.20.0.209 — Konsole

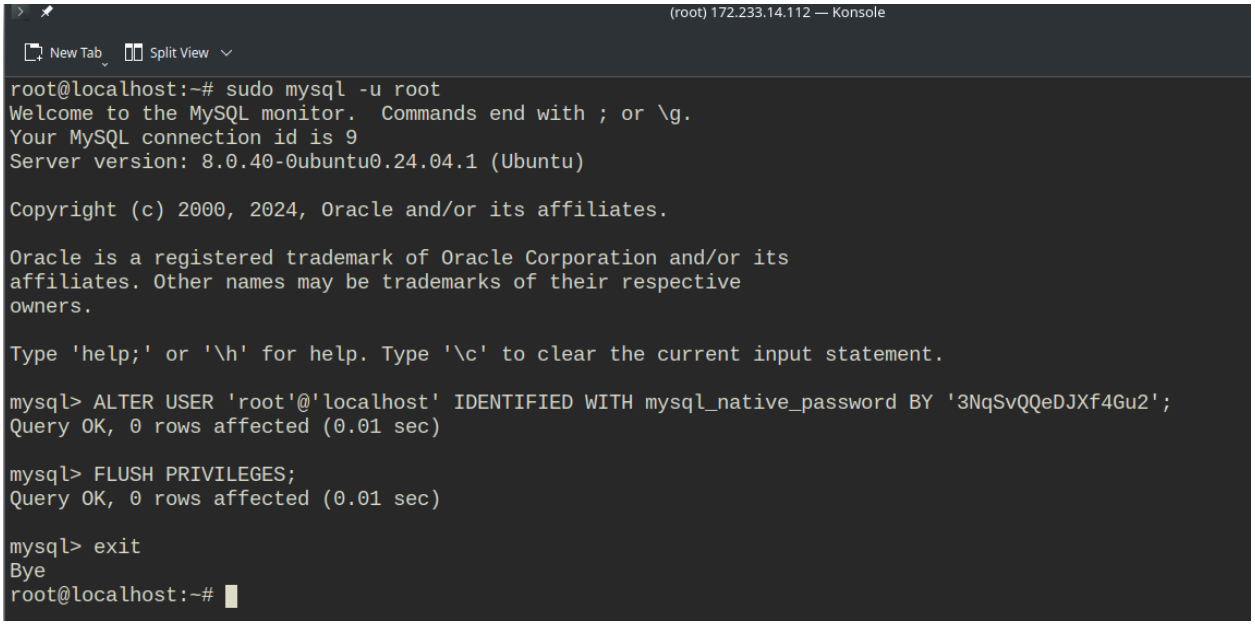
New Tab Split View Copy Paste Find...

```
ubuntuser@ubuntulab:~/Desf6$ ansible-playbook -i inventory.ini playbook.yml

PLAY [Configurar servidor Linode para WordPress] ********************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [172.233.27.216]

TASK [php : Instalar PHP y dependencias] ****************************************************************************
ok: [172.233.27.216]

TASK [mysql : Instalar MySQL Server] ********************************************************************************
ok: [172.233.27.216]

TASK [mysql : Habilitar y arrancar MySQL] ***************************************************************************
ok: [172.233.27.216]

TASK [mysql : Crear base de datos de WordPress] *********************************************************************
fatal: [172.233.27.216]: FAILED! => {"changed": false, "msg": "unable to find /root/.my.cnf. Exception message: (1698, \"Acces
s denied for user 'root'@'localhost'\")"}

PLAY RECAP **********************************************************************************************************
172.233.27.216             : ok=4    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0

ubuntuser@ubuntulab:~/Desf6$
```

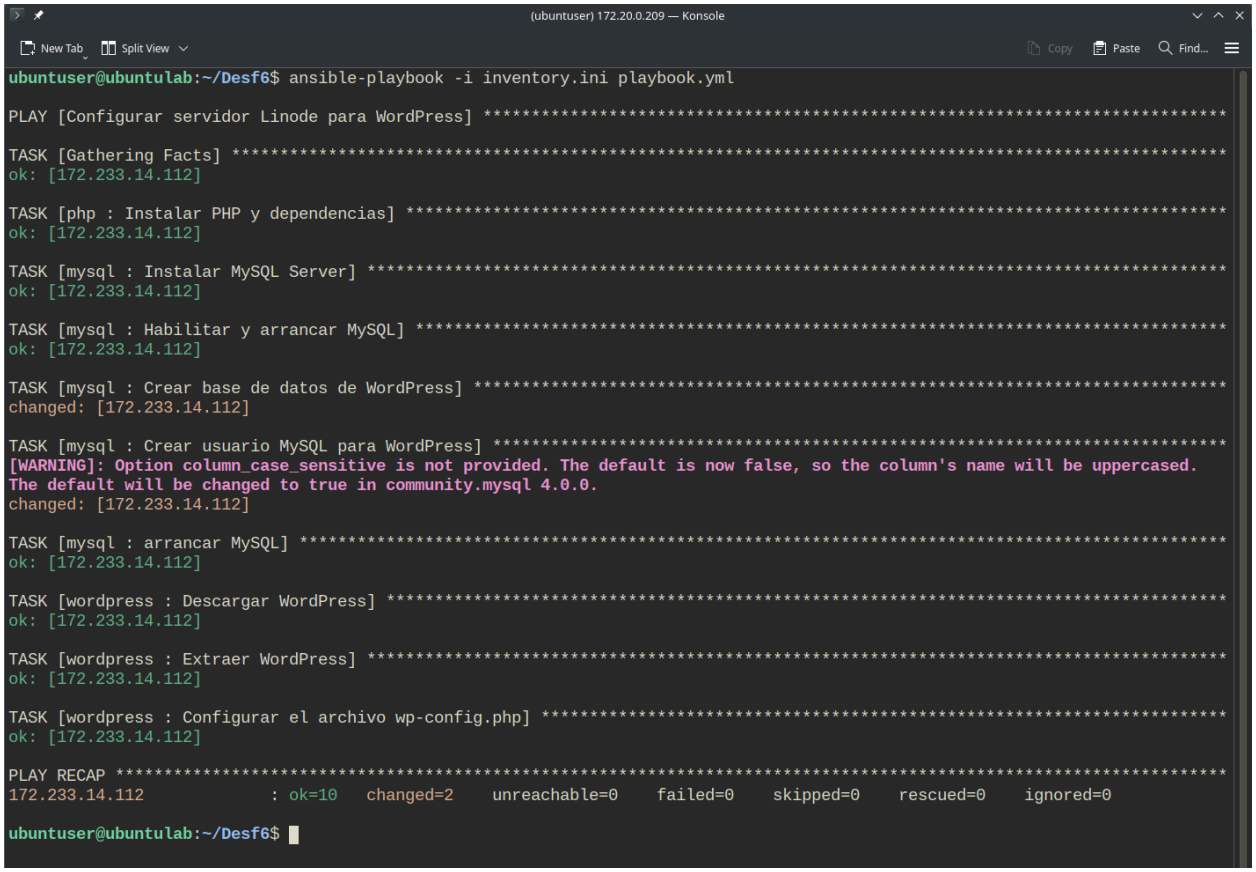
```
TASK [mysql : Crear base de datos de WordPress]
*********************************************************************
fatal: [172.233.27.216]: FAILED! => {"changed": false, "msg": "unable to find
/root/.my.cnf. Exception message: (1698, \"Access denied for user
'root'@'localhost'\")"}
```

MySQL está rechazando la conexión de root desde localhost, esto sucede en las versiones más actuales de MySQL.

Workaround: Permitir el login con el user root a mysql. Nota: aunque necesito acceso a la instancia remota para hacer este cambio, en un ambiente productivo no usaría root para conectarme sino un usuario dedicado, lo que anularía este problema.



Ejecución del Playbook:

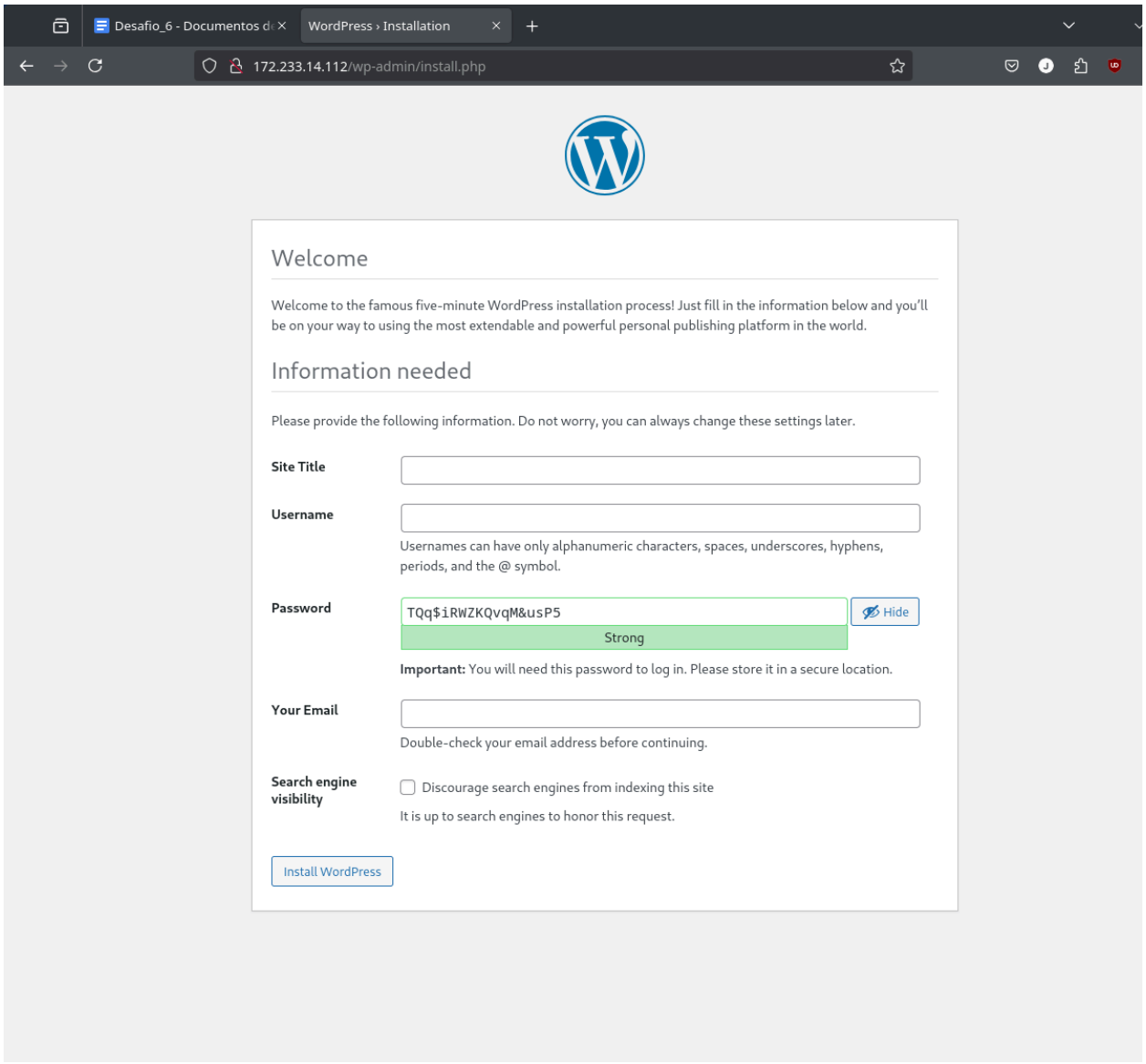


Prueba de conexión:

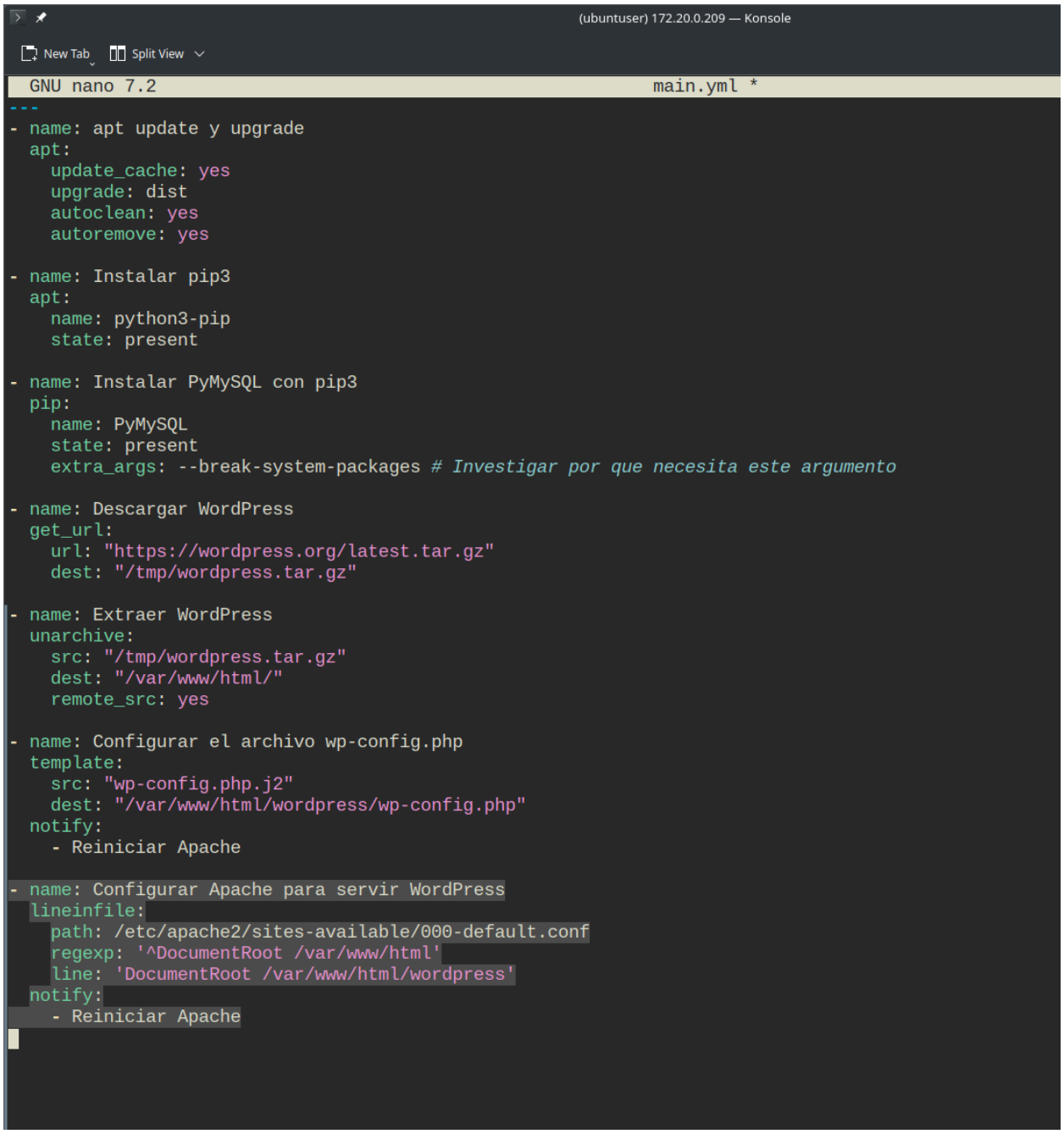
Al conectarme recibí el default page de Apache. Revisando el directorio Sites Available, note que el Documentroot seguía apuntando a /var/www/html.

```
root@localhost:/etc/apache2/sites-available# cat 000-default.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Luego de ajustarlo a /var/www/html/wordpress se pudo acceder a la pagina de configuracion:



Para corregir esto agregue un task en el rol de wordpress forzar la actualización del archivo:


(ubuntuser) 172.20.0.209 — Konsole
New Tab
Split View
GNU nano 7.2 main.yml *

- name: apt update y upgrade
apt:
update_cache: yes
upgrade: dist
autoclean: yes
autoremove: yes
- name: Instalar pip3
apt:
name: python3-pip
state: present
- name: Instalar PyMySQL con pip3
pip:
name: PyMySQL
state: present
extra_args: --break-system-packages # Investigar por que necesita este argumento
- name: Descargar WordPress
get_url:
url: "https://wordpress.org/latest.tar.gz"
dest: "/tmp/wordpress.tar.gz"
- name: Extraer WordPress
unarchive:
src: "/tmp/wordpress.tar.gz"
dest: "/var/www/html/"
remote_src: yes
- name: Configurar el archivo wp-config.php
template:
src: "wp-config.php.j2"
dest: "/var/www/html/wordpress/wp-config.php"
notify:
- Reiniciar Apache
- name: Configurar Apache para servir WordPress
lineinfile:
path: /etc/apache2/sites-available/000-default.conf
regexp: '^DocumentRoot /var/www/html'
line: 'DocumentRoot /var/www/html/wordpress'
notify:
- Reiniciar Apache