# WSL Apache MySQL PHPMyAdmin WordPress

## Objetivos

- Introducir a *WSL* como herramienta esencial para el desarrollo web moderno en entornos Windows.
- Instalar y configurar Ubuntu mediante WSL2 en Windows, creando una base sólida de sistema operativo Linux integrado.
- Desplegar un entorno *LAMP* local (Linux, Apache, MySQL, PHP) dentro del sistema Ubuntu instalado.
- Instalar y configurar WordPress en el entorno local para simular un sitio web real.
- Fomentar buenas prácticas de administración de bases de datos, permisos y seguridad básica durante el proceso.
- Demostrar la viabilidad del desarrollo web local sin depender de servicios externos o máquinas virtuales pesadas.

En la actualidad, el desarrollo web requiere entornos flexibles, potentes y lo más cercanos posible a los servidores de producción. Muchos sistemas backend, como WordPress, se ejecutan nativamente en entornos Linux, lo que puede representar un desafío para estudiantes o desarrolladores que utilizan Windows como sistema operativo principal.

Es aquí donde Windows Subsystem for Linux (*WSL*) surge como una solución innovadora: permite ejecutar una distribución completa de Linux directamente sobre Windows, sin necesidad de máquinas virtuales ni reinicios. Esta herramienta no solo facilita el aprendizaje de tecnologías basadas en Linux, sino que también habilita la creación de ambientes de desarrollo locales robustos, como el stack *LAMP* (Linux, Apache, MySQL, PHP), ideal para probar y desarrollar sitios con WordPress.

En esta presentación, demostraremos cómo configurar un entorno de desarrollo completo utilizando WSL2 con Ubuntu, instalando posteriormente Apache, MySQL, PHP y WordPress, todo ello desde nuestro propio equipo con Windows. El resultado será un sitio local funcional, listo para pruebas, personalización y aprendizaje práctico de desarrollo web.

## ¿Qué es WSL?

*WSL* (Windows Subsystem for Linux) es una herramienta desarrollada por Microsoft que permite ejecutar un entorno completo de Linux directamente dentro de Windows, sin necesidad de máquinas virtuales ni dual boot.

Es como tener una terminal de Ubuntu, Debian o otro sistema Linux integrada en tu PC Windows, con acceso completo al kernel Linux y a todas las herramientas de línea de comandos. Ideal para desarrolladores que quieren usar tecnologías basadas en Linux —como WordPress, PHP, MySQL, Python, Go o Docker, entre muchos otros — sin dejar de usar Windows como sistema principal.

En resumen:

*WSL* te da Linux en Windows… ¡sin reiniciar!

## Iniciemos

## Paso 0: Instalar WSL

Vamos a instalar WSL con Ubuntu desde Microsoft Store

- Abre la Microsoft Store.
- Busca "Ubuntu" (la versión LTS más reciente, por ejemplo, Ubuntu 22.04 LTS o 24.04 LTS).
- Haz clic en Instalar.

Comprobar que se instaló nuestro Linux Ubuntu:

- PowerShell

  *wsl.exe --list --verbose*

Distribucciones que se pueden instalar :

- PowerShell

  *wsl.exe --list --online*

Configurar el Usuario en Ubuntu

Una vez instalado, abre Ubuntu desde el menú Inicio, durante el primer inicio, se te pedirá:

- Crear un nombre de usuario (ej: usuario).
- Establecer una contraseña (no se ve al escribir, es normal).

Para instalar WSL desde PowerShell y mas información, visitar el siguiente enlace:

- https://learn.microsoft.com/es-es/windows/wsl/install

Identificar que version de Ubuntu se instalo:

- Bash

  *cat /etc/os-release*

Requisitos Previos

- Un servidor con Ubuntu 26.04.
- Un usuario con privilegios sudo.
- Un nombre de dominio apuntando a la IP del servidor (o se puede usar localhost para pruebas locales).

En caso de olvidar la clave del WSL, hacer lo siguiente:

- Abra la terminal de Windows cmd.exe o PowerShell
- Tipo wsl -u root ( opcional -d <distro-name> )
- Escribe passwd username y cambia la contraseña
- Digitar exit

## Paso 1: Instalar el stack LAMP

Actualiza los paquetes del sistema e instala Apache, MySQL y PHP con las extensiones necesarias:
bash

- Bash

  *sudo apt update && sudo apt upgrade -y*

## Paso 2: Instalar MySQL y crear una base de datos para WordPress

En caso de generar error, reinstalamos el MySQL Server

- Bash

  *sudo apt install mysql-server*

  En caso de error:
  - Bash
    *sudo apt install --reinstall mysql-server*

  o recreamos el archivo de configuración

  - Bash
    *sudo update-alternatives --config my.cnf*

Identificamos si MySQL esta activo:

- Bash

  *sudo systemctl status mysql*

Ingresa a la consola de MySQL como usuario root para crear una base de datos y un usuario dedicados: bash

- Bash

  *sudo mysql -u root -p*

Una vez dentro del shell de MySQL, ejecuta los siguientes comandos (asegúrate de cambiar 'tu_contraseña_segura' por una contraseña fuerte):

Sql

*CREATE DATABASE wordpress_db;*

#Creamos un usuario para la conexión local

*CREATE USER 'wordpress_user'@'localhost' IDENTIFIED BY 'Unicatolica2025+';*
*GRANT ALL PRIVILEGES ON wordpress_db.* TO 'wordpress_user'@'localhost';*

*FLUSH PRIVILEGES;*

#Creamos un usuario para la conexión remota

*CREATE USER 'wp_user' IDENTIFIED BY 'Unicatolica2025+';*
*GRANT ALL PRIVILEGES ON *.* to 'wp_user'@'%';*

*FLUSH PRIVILEGES;*

# Identicamos que los usuarios esten creados y que en host  wp_user tenga %

*SELECT user,host FROM mysql.user;*

*EXIT;*

Permitimos que MySQL escuche ip externas, para eso editamos el archivo mysqld.cnf para cambiar la directiva bind-address a 0.0.0.0 o a colocamos la IP del servidor.

- Bash

  *sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf*

  # Comentamos esta linea
  # bind-address  = 127.0.0.1
  # Adicionamos esta linea
  bind-address  = 0.0.0.0

Reiniciamos MySQL para que tengan efectos los cambios.

- Bash

  *sudo systemctl restart mysql*

**Probar conexión en MySQL Workbench.**

Identificamos la IP de nuestro Ubuntu con :

- Bash

  *hostname -I*

ó

- Bash

  *ip a*

Con esto ya podríamos probar la conexión en nuestro MySQL Workbench.

MySQL Online

*https://onecompiler.com/mysql*

**Instalamos los paquetes necesarios, como el PHP, Apache**

- Bash

  *sudo apt install apache2 php libapache2-mod-php php-mysql php-imagick php-curl php-zip php-xml php-bcmath php-intl php-json php-mbstring php-gd ghostscript -y*

Mientras instala:

- Que es sudo? (superuser do) es un comando de Linux que te permite ejecutar acciones con permisos de administrador o de otro usuario de forma temporal

- Que es cURL? cURL (Client URL) es una herramienta de línea de comandos que permite transferir datos hacia/desde un servidor utilizando varios protocolos.

**Archivos configuración Apache2 :**

- /etc/apache2/apache2.conf
- /etc/apache2/ports.conf
- /etc/apache2/sites-available/000-default.conf

**Archivos configuración PHP :**

- /etc/php/8.3/apache2/php.ini
- /etc/php/8.3/mods-available
- /etc/php/8.3/apache2/conf.d
- /etc/php/8.3/cli/conf.d

**/etc/php/8.3/apache2/conf.d**

- **Propósito**: Configurar PHP para el servidor web Apache.
- **Uso**: Se aplica a scripts PHP que se ejecutan cuando se accede a ellos a través de un navegador web.
- **Ejemplo**: Modificar directivas como upload_max_filesize para subida de archivos en aplicaciones web o memory_limit para el consumo de memoria del servidor.
- **Archivo de configuración principal**: El archivo php.ini para el módulo de Apache, que suele estar en una ruta como /etc/php/8.3/apache2/php.ini en sistemas Ubuntu.

**/etc/php/8.3/cli/conf.d**

- **Propósito**: Configurar PHP para la ejecución a través de la línea de comandos (CLI).
- **Uso**: Se aplica a scripts PHP que se ejecutan directamente en la terminal del sistema operativo.
- **Ejemplo**: Modificar el error_reporting para ver más detalles de errores en el desarrollo en la consola o cambiar memory_limit para scripts de procesamiento por lotes que no son accedidos por el web server.
- **Archivo de configuración principal**: El archivo php.ini para la versión CLI de PHP, que podría encontrarse en rutas como /etc/php/8.3/cli/php.ini en sistemas Ubuntu.
- **Ejemplo cli**: Crear un archivo llamado hello.php, gruardarlo en /var/www/html con este contenido y ejecutarlo con php hello.php:

  - PHP
    ```
    <?php
    echo "¡Hola mundo!\n";
    ?>
    ```

Desde nuestro navegador en Windows, podremos ver la pagina de prueba de Apache2.

Identificamos la IP de nuestro Ubuntu con :

- Bash

  *hostname -I*

ó

- Bash

  *ip a*

Y colocamos la IP en nuestro navegador.

## Descargar e instalar WordPress

Descarga la última versión de WordPress desde el sitio oficial y configúrala en el directorio web de Apache:

- Bash

  *cd /var/www/html/*
  *sudo curl -O https://wordpress.org/latest.tar.gz*
  *sudo tar -xzvf latest.tar.gz*

Asigna los permisos correctos al directorio de WordPress para que el servidor web (www-data) pueda acceder a él:

- Bash

  sudo chown -R www-data:www-data /var/www/html/wordpress/

## Configurar WordPress

Identificamos la IP de nuestro Ubuntu con :

- Bash

  *hostname -I*

ó

- Bash

  *ip a*

Ahora, desde tu navegador web, visita la dirección IP de tu servidor o tu nombre de dominio (ej. http://172.19.73.227/). Esto te llevará a la interfaz gráfica de instalación de WordPress.

Haz clic en "Let's go!" (¡Vamos!).

Introduce los detalles de la base de datos que hemos creado:

- **Nombre de la base de datos**: wordpress_db
- **Nombre de usuario**: wordpress_user
- **Contraseña**: Unicatolica2025+
- **Servidor de la base de datos**: localhost

- **Prefijo de tabla**: wp_ (dejamos el valor predeterminado)

Haz clic en "Submit" (Enviar) y luego en "Run the installation" (Ejecutar la instalación).

Completa la información del sitio (título, nombre de usuario administrador, contraseña y correo electrónico) y haz clic en "Install WordPress" (Instalar WordPress).

**Sitio :** http://172.19.73.227/wordpress/
**Admin:** http://172.19.73.227/wordpress/wp-admin

## FTP

Instalamos el paquete vsftpd usando el siguiente comando:

- Bash

  *sudo apt install vsftpd*

Editamos el archivo vsftpd.conf, pero primero creamos una copia del archivo.

- Bash

  *sudo cp /etc/vsftpd.conf /etc/vsftpd.conf_orig*
  *sudo nano /etc/vsftpd.conf*

Revisamos que este asi:

```
listen=NO
listen_ipv6=YES
anonymous_enable=NO
local_enable=YES
write_enable=YES
```

Creamos el usuario ftp_user

- Bash

  *sudo adduser ftp_user*

Cambiamos los permisos

- Bash

  *sudo usermod -a -G www-data ftp_user*
  *sudo usermod -d /var/www/html/wordpress ftp_user*

Opcional

- Bash

  *sudo chmod -R 755 /var/www/html/wordpress*

Reiniciamos el servicio VSFTPD

- Bash
  *sudo systemctl restart vsftpd*

## Filezilla

Protocolo : FTP
Servidor : ** IP Ubuntu **
Cifrado : Usar FTP explicito sobre TLS si esta disponible
Modo de acceso : Normal
Usuario : ftp_user
Contraseña : Unicatolica2025+

## Visual Studio

Configuramos en nuestro FileZilla o en nuestro Visual Studio.

Para poder visualizar nuestros archivos en Visual Studio, instalamos SFTP,

```
{
   "name": "172.19.73.227",
   "host": "172.19.73.227",
   "protocol": "ftp",
   "port": 21,
   "username": "ftp_user",
   "password": "Unicatolica2025+",
   "remotePath": "/var/www/html/wordpress",
   "uploadOnSave": true,
   "useTempFile": false,
   "openSsh": false
}
```

https://www.youtube.com/watch?v=TU19tEzjkOs

# Instalación de PostgreSQL

Para instalar PostgreSQL en WSL (es decir, Ubuntu):

Abra el terminal WSL (es decir, Ubuntu).

Actualice los paquetes de Ubuntu:

- Bash

  *sudo apt update*

Una vez actualizados los paquetes, instale PostgreSQL (y el paquete contrib de PostgreSQL, que tiene algunas utilidades útiles) con:

- Bash

  *sudo apt install postgresql postgresql-contrib*

Confirme la instalación y obtenga el número de versión:

- Bash

  *psql --version*

Hay 3 comandos que debe saber una vez instalado PostgreSQL:

Comprobación del estado de la base de datos

- Bash

  *sudo service postgresql status*

Iniciar la base de datos

- Bash

  *sudo service postgresql start*

Detener la base de datos

- Bash

  *sudo service postgresql stop*

El usuario administrador predeterminado, postgres, necesita una contraseña asignada para conectarse a una base de datos. Para establecer una contraseña:

Escriba el comando:

- Bash

  *sudo passwd postgres*

Recibirá un mensaje para escribir la nueva contraseña. Cierre y vuelva a abrir el terminal.

Para ejecutar PostgreSQL con psql shell:

Inicie el servicio postgres:

- Bash

  *sudo service postgresql start*

Conéctese al servicio postgres y abra el shell de psql:

- Bash

  sudo -u postgres psql

Una vez que haya ingresado correctamente al shell de psql, verá que la línea de comandos cambia para que se vea así:

*psqlshell*

*postgres=#*

**Nota:**

Como alternativa, puede abrir el shell de psql cambiando al usuario de Postgres con: su - postgres y después escribiendo el comando: psql.

Para salir de postgres=# escriba:

*\q*

o use la tecla de método abreviado: Ctrl+D

Para ver qué cuentas de usuario se han creado en la instalación de PostgreSQL, use desde el terminal WSL: psql --command="\du" ... o simplemente \du si tiene abierto el shell de psql. Este comando mostrará columnas: Nombre de usuario de la cuenta, Lista de atributos de roles y Miembro de grupos de roles. Para volver a la línea de comandos, escriba: q.

Ing. Carlos Eduardo Molina C cemolina@redtauros.com

https://learn.microsoft.com/es-es/windows/wsl/tutorials/wsl-database

**PostgreSQL Online**

*https://onecompiler.com/postgresql*

# SQL Server 2022

Dentro de tu distribución Ubuntu en WSL, ejecuta:

1. Importar claves GPG de Microsoft

- Bash

  *curl https://packages.microsoft.com/keys/microsoft.asc | sudo tee /etc/apt/trusted.gpg.d/microsoft.asc*

2. Registrar el repositorio de SQL Server

- Bash

  *sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/config/ubuntu/22.04/mssql-server-2022.list)"*

3. Actualizar e instalar

- Bash

  *sudo apt-get update*

  *sudo apt-get install -y mssql-server*

4. Configurar SQL Server

- Bash

  *sudo /opt/mssql/bin/mssql-conf setup*

Instalar dependencia libladp

bash

*wget http://mirrors.kernel.org/ubuntu/pool/main/o/openldap/libldap-2.5-0_2.5.11+dfsg-1~exp1ubuntu3_amd64.deb (search here https://pkgs.org/search/?q=libldap)*

*sudo dpkg -i libldap-2.5-0_2.5.11+dfsg-1~exp1ubuntu3_amd64.deb*

Durante la configuración selecciona:

Edición: Developer (gratuita para desarrollo/educación) o Express
Contraseña SA: Debe tener mínimo 8 caracteres, mayúsculas, minúsculas, números y símbolos

5. Verificar que el servicio esté corriendo

bash
systemctl status mssql-server --no-pager

Paso 5: Instalar Herramientas de Línea de Comandos (sqlcmd)

Para poder interactuar desde terminal:

- Bash

*# Importar claves*
*curl https://packages.microsoft.com/keys/microsoft.asc | sudo tee /etc/apt/trusted.gpg.d/microsoft.asc*

*# Registrar repositorio de herramientas*
*curl https://packages.microsoft.com/config/ubuntu/22.04/prod.list | sudo tee /etc/apt/sources.list.d/mssql-release.list*

# Instalar herramientas
*sudo apt-get update*
*sudo ACCEPT_EULA=Y apt-get install -y mssql-tools18 unixodbc-dev*

# Añadir al PATH
*echo 'export PATH="$PATH:/opt/mssql-tools18/bin"' >> ~/.bashrc*
*source ~/.bashrc*

Paso 6: Conexión desde Windows
Para conectarte desde SQL Server Management Studio (SSMS) o Azure Data Studio en Windows:

https://learn.microsoft.com/es-es/ssms/install/install

Obtener la IP de WSL:

- Bash

*hostname -I*

Conexión:

| | |
|---|---|
| *Servidor* | : La IP que arrojó el comando anterior, o localhost |
| *Puerto* | : 1433 (por defecto) |
| *Autenticación* | : SQL Server Authentication |
| *Usuario* | : sa |
| *Contraseña* | : La que configuraste en el paso 4 |

Nota: Si usas localhost desde Windows, necesitas configurar port forwarding en PowerShell:

- powershell

  *netsh interface portproxy add v4tov4 listenport=1433 listenaddress=0.0.0.0 connectport=1433 connectaddress=(wsl hostname -I).trim()*

Paso 7: Comandos Útiles

- bash

  *# Iniciar/Detener/Reiniciar SQL Server*
  *sudo systemctl start mssql-server*
  *sudo systemctl stop mssql-server*
  *sudo systemctl restart mssql-server*

  *# Ver estado*
  *sudo systemctl status mssql-server*

  *# Conectar localmente con sqlcmd*
  *sqlcmd -S localhost -U sa -P 'TuContraseñaSegura123!'*

  *# Crear base de datos desde sqlcmd*
  *CREATE DATABASE UniversidadDB;*
  *GO*
  *USE UniversidadDB;*
  *GO*
  *ALTER LOGIN sa WITH PASSWORD = '<strong_password>';*

  *DROP USER IF EXISTS [Universidad];*

  # Crear usuario con permisos
  -- 1. Crear el login (Inicio de sesión) en el servidor
  USE master
  *GO*
  *CREATE LOGIN Universidad WITH PASSWORD = N'Clave123!!',*

```
    DEFAULT_DATABASE = [master],
    CHECK_EXPIRATION = OFF,
    CHECK_POLICY = ON
GO

-- 2. Otorgar permisos de Administrador Total (Sysadmin)
ALTER SERVER ROLE [sysadmin] ADD MEMBER [NuevoUsuario]
GO

-- 3. (Opcional) Crear usuario en una base de datos específica y hacerlo dueño
USE [NombreDeTuBaseDeDatos]
GO
CREATE USER [NuevoUsuario] FOR LOGIN [NuevoUsuario]
GO
ALTER ROLE [db_owner] ADD MEMBER [NuevoUsuario]
GO
```

**https://learn.microsoft.com/es-es/sql/tools/sqlcmd/sqlcmd-use-utility?view=sql-server-ver17**

**SQL Server Online**

*https://onecompiler.com/sqlserver*

## Resumen

| Motor | Instalación en WSL | Puerto Default |
|---|---|---|
| MySQL | sudo apt install mysql-server | 3306 |
| PostgreSQL | sudo apt install postgresql | 5432 |
| SQL Server | Proceso | 1433 |

A considerar:

- Rendimiento : SQL Server en WSL es para desarrollo/educación, no para producción
- Persistencia : Los datos se mantienen en el filesystem de WSL (/var/opt/mssql/)
- Backups : Puedes hacer backup de bases de datos en /mnt/c/ para acceso desde Windows
- Docker de WSL : Si prefieres, también puedes usar SQL Server en contenedores Docker dentro