

# TEMARIO: ARQUITECTURA EN LA NUBE - PRÁCTICA DE INSTALACIÓN DE WORDPRESS EN LA NUBE

## Objetivos

Al finalizar esta práctica, serás capaz de:

- Instalar un servidor web completo (LAMP)
  - Configurar WordPress manualmente
  - Usar un dominio gratuito con DuckDNS
  - Activar HTTPS con certificado SSL gratuito
- 

## Requisitos Previos

Antes de comenzar necesitas:

- Ordenador con Ubuntu 22.04 o superior (o WSL2 en Windows)
  - Conexión a Internet
  - Acceso al router (para abrir puertos)
  - Cuenta en Google, GitHub o Twitter
- 

## PARTE 1 – Instalación del Servidor LAMP

### 1.1 – Actualizar el sistema

```
sudo apt update
```

```
sudo apt upgrade -y
```

---

### 1.2 – Instalar Apache (Servidor Web)

```
sudo apt install apache2 -y
```

Verificar servicio:

```
sudo systemctl status apache2
```

Probar en el navegador: <http://localhost>

### 1.3 – Instalar MySQL (Base de Datos)

```
sudo apt install mysql-server -y
```

Asegurar la instalación:

```
sudo mysql_secure_installation
```

Respuestas recomendadas:

Pregunta	Respuesta
¿Validación de contraseñas?	N
¿Cambiar contraseña de root?	Y
Contraseña (ejemplo)	Admin123!
Restantes preguntas	Y

---

### 1.4 – Instalar PHP

```
sudo apt install php php-mysql libapache2-mod-php php-curl php-gd php-mbstring php-xml php-xmlrpc php-intl php-zip -y
```

Verificar PHP:

```
php -v
```

Reiniciar Apache:

```
sudo systemctl restart apache2
```

---

## PARTE 2 – Crear Base de Datos para WordPress

### 2.1 – Acceder a MySQL

```
sudo mysql
```

---

### 2.2 – Crear base de datos y usuario

```
CREATE DATABASE wordpress;
```

```
CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'WordPress123!';
```

```
GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'localhost';
```

```
FLUSH PRIVILEGES;
```

```
EXIT;
```

Guardar datos de acceso:

Elemento	Valor
Base de datos	wordpress
Usuario	wpuser
Contraseña	WordPress123!

---

## PARTE 3 – Instalar WordPress

### 3.1 – Descargar WordPress

```
cd /tmp
```

```
wget https://wordpress.org/latest.tar.gz
```

```
tar -xzf latest.tar.gz
```

---

### 3.2 – Copiar archivos a Apache

```
sudo rm -rf /var/www/html/*
```

```
sudo cp -r wordpress/* /var/www/html/
```

---

### 3.3 – Permisos

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

```
sudo chmod -R 755 /var/www/html/
```

---

### 3.4 – Configurar wp-config.php

```
sudo cp /var/www/html/wp-config-sample.php /var/www/html/wp-config.php
```

```
sudo nano /var/www/html/wp-config.php
```

Modificar valores:

```
define( 'DB_NAME', 'wordpress' );
```

```
define( 'DB_USER', 'wpuser' );
```

```
define( 'DB_PASSWORD', 'WordPress123!' );
```

```
define( 'DB_HOST', 'localhost' );
```

---

### 3.5 – Finalizar en el navegador

Acceder a: <http://localhost>

Completar la instalación (idioma, usuario, contraseña, título del sitio).

---

## PARTE 4 – Hacer WordPress Accesible desde Internet con ngrok

### 4.1 – ¿Qué es ngrok?

ngrok es un servicio que crea túneles seguros desde Internet hacia tu máquina local, sin necesidad de configurar el router ni abrir puertos. Es ideal para desarrollo y pruebas.

---

### 4.2 – Registro en ngrok

1. Accede a: <https://ngrok.com>
  2. Haz clic en Sign up y crea una cuenta (puedes usar Google o GitHub).
  3. En el panel principal, ve a Your Authtoken.
  4. Copia tu token de autenticación (lo necesitarás más adelante).
- 

### 4.3 – Instalar ngrok en Ubuntu

#### Método 1: Descarga directa (recomendado)

```
cd ~
```

```
wget https://bin.equinox.io/c/bNyj1mQVY4c/ngrok-v3-stable-linux-amd64.tgz
```

Extraer el archivo:

```
tar -xvzf ngrok-v3-stable-linux-amd64.tgz
```

Mover ngrok a una ubicación del sistema:

```
sudo mv ngrok /usr/local/bin/
```

## Método 2: Usando Snap

```
sudo snap install ngrok
```

---

### 4.4 – Verificar instalación

```
ngrok version
```

Salida esperada: ngrok version 3.x.x

---

### 4.5 – Autenticar ngrok

Configura tu token de autenticación (reemplaza con tu token real):

```
ngrok config add-authtoken TU_TOKEN_AQUI
```

Ejemplo:

```
ngrok config add-authtoken  
2abc3def4ghi5jkl6mno7pqr8stu9vwx_YourActualTokenHere123
```

---

### 4.6 – Iniciar túnel HTTP

Ejecuta ngrok para exponer el puerto 80 (Apache):

```
ngrok http 80
```

---

### 4.7 – Anotar URLs

ngrok proporciona dos URLs:

- HTTP: <http://abc123def456.ngrok-free.app>
- HTTPS: <https://abc123def456.ngrok-free.app> (usar esta)

Importante: Guarda esta URL para configurar WordPress.

## 4.8 – Probar acceso

Abre un navegador y accede a:

<https://tu-url.ngrok-free.app>

En la primera visita, ngrok puede mostrar una página de advertencia. Haz clic en Visit Site para continuar.

---

## 4.9 – Verificación

- ngrok está ejecutándose
  - Dispone de una URL del tipo: <https://xxxxx.ngrok-free.app>
  - Acceso desde cualquier navegador
  - Interfaz web disponible en <http://localhost:4040>
- 

## Ventajas de ngrok

- No requiere configurar router
  - No necesita IP pública fija
  - HTTPS incluido automáticamente
  - Funciona detrás de firewalls corporativos
  - Interfaz web para debugging
- 

## Desventajas de ngrok

- La URL cambia cada vez que se reinicia (*plan gratuito*)
- Límite de conexiones por minuto (*plan gratuito*)
- Página de advertencia en primera visita
- Depende de un servicio externo

## Notas importantes

- Para producción real: no usar ngrok. Es solo para desarrollo y pruebas.
  - URL dinámica: con el plan gratuito, cada reinicio genera una URL nueva. Es necesario actualizar WordPress.
  - Alternativas más estables: DuckDNS con túnel SSH, Cloudflare Tunnel u otros servicios similares.
  - No usar contraseñas de ejemplo en producción
  - Realizar copias de seguridad
  - Mantener WordPress y plugins actualizados
- 

## Entregables en PDF - Capturas requeridas:

### 1. ngrok en ejecución

- Terminal mostrando ngrok activo con la URL pública generada
- Debe verse claramente la URL tipo `https://xxxxx.ngrok-free.app`

### 2. Servicios en terminal

- Captura de `sudo systemctl status apache2`
- Captura de `sudo systemctl status mysql`
- Ambos servicios deben mostrar estado "active (running)"

### 3. WordPress en navegador

- Página principal de WordPress cargando correctamente
- URL en la barra de direcciones mostrando la dirección ngrok
- Candado HTTPS visible (verde o gris según navegador)

### 4. Archivos de configuración

- Contenido de `wp-config.php` mostrando las líneas `WP_HOME` y `WP_SITEURL` con la URL de ngrok
- Comando: `cat /var/www/html/wp-config.php | grep -A2 WP_HOME`
- Panel de administración de WordPress accesible: `https://tu-url.ngrok-free.app/wp-admin`





# PRÁCTICAS OPCIONALES: WORDPRESS CON TECNOLOGÍAS ALTERNATIVAS

## Índice de Prácticas

1. **XAMPP (Windows/WSL)** – Dificultad: Fácil
  2. **Docker** – Dificultad: Media
  3. **Nginx + PHP-FPM** – Dificultad: Media
  4. **Base de Datos Alternativa (PostgreSQL/MariaDB)** – Dificultad: Avanzada
  5. **Cloud PaaS (Render/Railway)** – Dificultad: Avanzada
- 

## PRÁCTICA 1: XAMPP (Windows/WSL)

### Descripción

Instalación de WordPress utilizando XAMPP, un entorno gráfico que integra Apache, MySQL y PHP. Es la opción más sencilla para instalar WordPress en Windows.

### Objetivos de aprendizaje

- Uso de un entorno gráfico como alternativa a la línea de comandos
- Configuración de dominios locales
- Generación de certificados SSL autofirmados

### Pasos principales

#### 1. Descargar e instalar XAMPP

- Acceder a: <https://www.apachefriends.org>
- Descargar versión para Windows
- Instalar en C:\xampp
- Abrir **XAMPP Control Panel**

#### 2. Iniciar servicios

En el panel de control, pulsar **Start** en:

- Apache
- MySQL

### 3. Descargar WordPress

- Descargar desde <https://wordpress.org/latest.zip>
- Extraer en: C:\xampp\htdocs\miweb

### 4. Crear base de datos

- Abrir navegador: http://localhost/phpmyadmin
- Clic en **Nueva** → Nombre: wordpress → Crear

### 5. Configurar WordPress

Editar wp-config.php:

```
define('DB_NAME', 'wordpress');
```

```
define('DB_USER', 'root');
```

```
define('DB_PASSWORD', ''); // Vacío en XAMPP
```

```
define('DB_HOST', 'localhost');
```

### 6. Configurar dominio local

Editar archivo hosts (como administrador):

Ruta: C:\Windows\System32\drivers\etc\hosts

```
127.0.0.1 miweb.local
```

Crear Virtual Host

Editar C:\xampp\apache\conf\extra\httpd-vhosts.conf:

```
<VirtualHost *:80>
```

```
    ServerName miweb.local
```

```
    DocumentRoot "C:/xampp/htdocs/miweb"
```

```
    <Directory "C:/xampp/htdocs/miweb">
```

```
        AllowOverride All
```

```
        Require all granted
```

```
    </Directory>
```

```
</VirtualHost>
```

Reiniciar Apache.

## 7. (Opcional) SSL Autofirmado

Ejecutar: `C:\xampp\apache\makecert.bat`

Configurar Virtual Host para puerto 443.

### Capturas requeridas

1. Panel XAMPP con servicios activos
2. phpMyAdmin con base de datos creada
3. Navegador accediendo a `http://miweb.local`
4. Archivo `httpd-vhosts.conf`
5. WordPress instalado
6. Certificado SSL (si se realiza)

### Ventajas

- Uso sencillo
- Todo en un mismo instalador
- Ideal para desarrollo local

### Desventajas

- No apto para producción
- Dependiente de Windows

---

## PRÁCTICA 2: Docker

### Descripción

Creación de un entorno WordPress mediante contenedores Docker. Servicios separados en contenedores independientes (web, base de datos, etc.).

### Objetivos de aprendizaje

- Contenedorización
- Uso de Docker Compose
- Volúmenes persistentes
- Arquitectura por servicios

## Pasos principales

### 1. Instalar Docker Desktop

Comprobar instalación:

```
docker --version
```

```
docker-compose --version
```

### 2. Crear proyecto

```
mkdir wordpress-docker
```

```
cd wordpress-docker
```

### 3. Crear archivo docker-compose.yml

```
version: '3.8'
```

```
services:
```

```
  wordpress:
```

```
    image: wordpress:latest
```

```
    container_name: mi-wordpress
```

```
    restart: always
```

```
    ports:
```

```
      - "8080:80"
```

```
    environment:
```

```
      WORDPRESS_DB_HOST: db
```

```
      WORDPRESS_DB_USER: wpuser
```

```
      WORDPRESS_DB_PASSWORD: wppassword
```

```
      WORDPRESS_DB_NAME: wordpress
```

```
    volumes:
```

```
      - ./wordpress-data:/var/www/html
```

```
    depends_on:
```

```
      - db
```

db:

image: mysql:8.0

container\_name: mi-mysql

restart: always

environment:

MYSQL\_DATABASE: wordpress

MYSQL\_USER: wpuser

MYSQL\_PASSWORD: wppassword

MYSQL\_ROOT\_PASSWORD: rootpassword

volumes:

- ./db-data:/var/lib/mysql

phpmyadmin:

image: phpmyadmin:latest

container\_name: mi-phpmyadmin

restart: always

ports:

- "8081:80"

environment:

PMA\_HOST: db

MYSQL\_ROOT\_PASSWORD: rootpassword

depends\_on:

- db

#### 4. Iniciar entorno

docker-compose up -d

#### 5. Acceso a servicios

- WordPress: <http://localhost:8080>
- phpMyAdmin: <http://localhost:8081>

## 6. Comandos útiles

`docker ps`

`docker-compose logs -f wordpress`

`docker-compose down`

`docker-compose down -v`

### Capturas requeridas

1. Resultado de `docker ps`
2. Archivo `docker-compose.yml`
3. WordPress funcionando
4. phpMyAdmin operativo
5. Logs de contenedores
6. Carpetas de volúmenes

### Ventajas

- Alta portabilidad
- Replicable y escalable
- Aislamiento completo

### Desventajas

- Mayor complejidad
- Mayor consumo de recursos

---

## PRÁCTICA 3: NGINX + PHP-FPM

### ¿Qué vas a hacer?

Instalar y configurar WordPress utilizando **Nginx** como servidor web y **PHP-FPM** como gestor de procesos PHP, reemplazando Apache.

### ¿Qué aprenderás?

- Diferencias entre Apache y Nginx
- Configuración de server blocks en Nginx (equivalente a Virtual Hosts)
- Optimización de rendimiento con PHP-FPM

- Manejo de logs y reglas de reescritura

## Pasos principales

### 1. Instalar Nginx y PHP-FPM

```
sudo apt update
```

```
sudo apt install nginx php8.1-fpm php8.1-mysql php8.1-curl php8.1-gd php8.1-mbstring php8.1-xml php8.1-zip -y
```

### 2. Verificar PHP-FPM

```
sudo systemctl status php8.1-fpm
```

### 3. Instalar MySQL

```
sudo apt install mysql-server -y
```

```
sudo mysql_secure_installation
```

### 4. Crear base de datos

```
sudo mysql
```

```
CREATE DATABASE wordpress;
```

```
CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'WordPress123!';
```

```
GRANT ALL PRIVILEGES ON wordpress.* TO 'wpuser'@'localhost';
```

```
FLUSH PRIVILEGES;
```

```
EXIT;
```

### 5. Instalar WordPress

```
cd /tmp
```

```
wget https://wordpress.org/latest.tar.gz
```

```
tar -xzf latest.tar.gz
```

```
sudo mkdir -p /var/www/miweb
```

```
sudo cp -r wordpress/* /var/www/miweb/
```

```
sudo chown -R www-data:www-data /var/www/miweb/
```

### 6. Configurar WordPress

```
sudo cp /var/www/miweb/wp-config-sample.php /var/www/miweb/wp-config.php
```

```
sudo nano /var/www/miweb/wp-config.php
```

Modificar credenciales de base de datos.

## 7. Crear Server Block en Nginx

```
sudo nano /etc/nginx/sites-available/miweb
```

```
server {
```

```
    listen 80;
```

```
    server_name tunombre.duckdns.org;
```

```
    root /var/www/miweb;
```

```
    index index.php index.html;
```

```
    access_log /var/log/nginx/miweb-access.log;
```

```
    error_log /var/log/nginx/miweb-error.log;
```

```
    location / {
```

```
        try_files $uri $uri/ /index.php?$args;
```

```
    }
```

```
    location ~ \.php$ {
```

```
        include snippets/fastcgi-php.conf;
```

```
        fastcgi_pass unix:/var/run/php/php8.1-fpm.sock;
```

```
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
```

```
        include fastcgi_params;
```

```
    }
```

```
    location ~ /\.ht {
```

```
        deny all;
```

```
    }
```

```
}
```



## 8. Activar sitio y reiniciar Nginx

```
sudo ln -s /etc/nginx/sites-available/miweb /etc/nginx/sites-enabled/
```

```
sudo rm /etc/nginx/sites-enabled/default
```

```
sudo nginx -t
```

```
sudo systemctl restart nginx
```

## 9. Configurar SSL (opcional)

```
sudo apt install certbot python3-certbot-nginx -y
```

```
sudo certbot --nginx -d tunombre.duckdns.org
```

### Capturas obligatorias

1. Resultado de `sudo nginx -t`
2. Server block en `/etc/nginx/sites-available/miweb`
3. `sudo systemctl status nginx`
4. `sudo systemctl status php8.1-fpm`
5. WordPress cargando por HTTP/HTTPS

---

# PRÁCTICA 4: BASE DE DATOS ALTERNATIVA (PostgreSQL o MariaDB)

## ¿Qué vas a hacer?

Reemplazar MySQL por **PostgreSQL** o **MariaDB**, adaptando WordPress para funcionar con motores de base de datos no nativos.

## ¿Qué aprenderás?

- Diferencias entre motores SQL
- Compatibilidad y uso del plugin PG4WP
- Migración de bases de datos
- Resolución de errores en entornos no soportados oficialmente

## Pasos principales (PostgreSQL)

### 1. Instalar PostgreSQL

```
sudo apt install postgresql postgresql-contrib -y
```

## 2. Crear base de datos y usuario

```
sudo -u postgres psql
```

```
CREATE DATABASE wordpress;
```

```
CREATE USER wpuser WITH PASSWORD 'WordPress123!';
```

```
GRANT ALL PRIVILEGES ON DATABASE wordpress TO wpuser;
```

```
\q
```

## 3. Instalar WordPress

```
cd /tmp
```

```
wget https://wordpress.org/latest.tar.gz
```

```
tar -xzf latest.tar.gz
```

```
sudo cp -r wordpress/* /var/www/html/
```

## 4. Instalar plugin PG4WP

```
cd /tmp
```

```
wget https://github.com/PostgreSQL-For-Wordpress/postgresql-for-wordpress/archive/refs/heads/master.zip
```

```
unzip master.zip
```

```
sudo cp -r postgresql-for-wordpress-master/pg4wp /var/www/html/wp-content/
```

```
sudo cp /var/www/html/wp-content/pg4wp/db.php /var/www/html/wp-content/
```

## 5. Configurar wp-config.php

```
define('DB_NAME', 'wordpress');
```

```
define('DB_USER', 'wpuser');
```

```
define('DB_PASSWORD', 'WordPress123!');
```

```
define('DB_HOST', 'localhost');
```

```
define('DB_TYPE', 'pgsql');
```

```
define('DB_PORT', '5432');
```

## 6. Ajustar permisos y probar instalación

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

```
sudo systemctl restart apache2
```

## Capturas obligatorias

1. `sudo -u postgres psql -c "\l"`
2. Plugin PG4WP en `wp-content/pg4wp`
3. `wp-config.php` modificado
4. WordPress instalado con PostgreSQL

---

## PRÁCTICA 5: CLOUD PaaS (Render / Railway)

### ¿Qué vas a hacer?

Desplegar WordPress en una plataforma PaaS, utilizando Git, contenedores y bases de datos gestionadas en la nube.

### ¿Qué aprenderás?

- Despliegue continuo (CI/CD)
- Variables de entorno
- Escalado automático
- Uso de servicios gestionados de base de datos

### Estructura del proyecto

wordpress-render/

└─ Dockerfile

└─ .dockerignore

└─ nginx.conf

└─ README.md

### 1. Crear Dockerfile

FROM wordpress:latest

RUN apt-get update && apt-get install -y nginx && rm -rf /var/lib/apt/lists/\*

COPY . /var/www/html/

RUN chown -R www-data:www-data /var/www/html

EXPOSE 80

CMD ["apache2-foreground"]

## 2. Subir a GitHub

```
git init
```

```
git add .
```

```
git commit -m "Initial WordPress setup"
```

```
git branch -M main
```

```
git remote add origin https://github.com/tuusuario/wordpress-render.git
```

```
git push -u origin main
```

## 3. Configuración en Render.com

- Crear **Web Service** desde GitHub
- Seleccionar **Environment: Docker**
- Configurar base de datos PostgreSQL interna
- Añadir variables de entorno:

```
WORDPRESS_DB_HOST=
```

```
WORDPRESS_DB_USER=
```

```
WORDPRESS_DB_PASSWORD=
```

```
WORDPRESS_DB_NAME=
```

```
WORDPRESS_DB_PORT=5432
```

## 4. Instalar PG4WP si se usa PostgreSQL

### Capturas obligatorias

1. Repositorio GitHub
2. Panel Render o Railway
3. Logs de despliegue
4. Sitio en producción (<https://miapp.onrender.com>)
5. Variables de entorno configuradas