EBENEZER IT · GUÍAS PASO A PASO

GUÍA 05 · EPISODIO 10

WordPress con Docker Compose, detrás de Nginx y protegido

Una web WordPress en contenedores, con su base de datos MariaDB, publicada con HTTPS a través de Nginx y protegida contra los ataques que recibe cualquier WordPress desde el primer día. Con copias de seguridad automáticas.

~2–3 horas
la primera vez, sin prisa

Requisito
Guías 01, 02 y 03 terminadas

Ubuntu 24.04 LTS
WordPress · MariaDB · Nginx

www.ebenezer.org.es · Alejandro Quevedo · Versión pública 1.0 · noviembre 2026

Índice

Antes de empezar

0 Qué vas a construir y valores de esta guía

Parte A · Preparación

Paso 01 Comprobar lo que ya tienes

Paso 02 Subdominio y certificado

Parte B · Los contenedores

Paso 03 Carpeta y contraseñas

Paso 04 Elegir las versiones

Paso 05 El compose.yaml

Paso 06 Ajustes de PHP

Paso 07 Arrancar y comprobar

Parte C · Publicar e instalar

Paso 08 Publicar con Nginx y HTTPS

Paso 09 El asistente de instalación

Paso 10 Primeros ajustes y limpieza

Parte D · Proteger

Paso 11 Cerrar las puertas que no usas

Paso 12 fail2ban para los intentos de acceso

Paso 13 Actualizaciones

Paso 14 WP-CLI: WordPress desde la terminal

Parte E · Mantener

Paso 15 Que WordPress pueda enviar correos

Paso 16 Copias de seguridad automáticas

Paso 17 Comprobación final e instantánea

Anexos

A Problemas frecuentes

B Puertos al terminar esta guía

C Chuleta de WordPress con Docker

Cómo usar esta guía

Tienes aquí **todos los pasos y comandos**, en orden. La explicación de cada paso (el porqué de cada línea) está en el **vídeo del episodio**.

Para copiar los comandos sin errores, usa la versión web: **https://www.ebenezer.org.es/guia-05-wordpress.html**. Al copiar desde un PDF se pueden romper los saltos de línea y las comillas.

ANTES DE EMPEZAR

Qué vas a construir

Un WordPress en `https://blog.TU_DOMINIO`, montado con el mismo esquema de la Guía 03: contenedores que solo escuchan dentro del servidor y Nginx como única puerta de entrada. Esta vez son **dos** contenedores que trabajan juntos:

Pieza	Qué hace	Dónde vive
WordPress	La web: PHP y Apache dentro del contenedor oficial. Genera las páginas y el panel de administración	Contenedor, en `127.0.0.1:8082`
MariaDB	La base de datos: entradas, páginas, usuarios, ajustes	Contenedor, **sin ningún puerto publicado**. Solo WordPress habla con ella, por la red interna
Tus datos	Los archivos de WordPress (temas, plugins, imágenes subidas) y los de la base de datos	Carpetas en `/opt/apps/wordpress/datos`, fuera de los contenedores
Nginx	HTTPS, cabeceras de seguridad, y los filtros contra ataques (Parte D)	El del servidor (Guías 02 y 03)

El recorrido de una visita

1. El navegador pide `https://blog.TU_DOMINIO`. Nginx recibe la conexión en el 443 y presenta el certificado.
2. Nginx comprueba sus filtros (¿es un intento de acceso masivo? ¿pide algo prohibido?) y, si todo está bien, reenvía la petición a `127.0.0.1:8082`.
3. WordPress pide a MariaDB lo que necesita (la entrada, el menú, los ajustes) por la red interna de Docker.
4. WordPress monta la página y se la devuelve a Nginx, que la entrega cifrada.

Por qué hay que proteger WordPress desde el primer minuto

WordPress mueve una parte enorme de las webs del mundo, así que hay robots probando contraseñas en `wp-login.php` y buscando plugins vulnerables en cualquier WordPress nuevo, a todas horas. No es paranoia: en cuanto publiques el sitio lo verás en el registro de Nginx. Por eso la Parte D no es opcional.

Valores de esta guía

Valor	Qué es
`TU_DOMINIO`	Tu dominio, como en las guías anteriores
`TU_IP`	La IP del VPS
`TU_USUARIO`	Tu usuario del servidor

Valor	Qué es
TU_VERSION_WP	La versión exacta de la imagen de WordPress (paso 4)
TU_VERSION_MARIADB	La versión de MariaDB (paso 4)
TU_USUARIO_WP	Tu usuario administrador de WordPress. **Nunca** `admin` (paso 9)
TU_CORREO	El correo al que te llegarán los avisos de WordPress

Recordatorio

- Haz una instantánea en IONOS antes de empezar (`04-docker-proxy` o la última que tengas).
- Bloques **verdes** en el servidor (`ssh lab`), **azules** en PowerShell de tu PC, **morados** son contenido de archivos, **ocres** son clics en un panel web.
- `sudo nginx -t` siempre antes de recargar Nginx.

PARTE A

Preparación

Dos comprobaciones rápidas y el certificado del subdominio. Es el mismo procedimiento de la Guía 03, así que irá rápido.

Paso 1 Episodio 10

Comprobar lo que ya tienes

Por qué: WordPress se apoya en todo lo anterior: Docker, Nginx con sus snippets y el certificado por *webroot*.

EN EL SERVIDOR

```
docker compose version
ls /etc/nginx/snippets/ /etc/nginx/conf.d/
free -h
sudo ss -tlpn | grep -E ':(8082|3306) '
```

SALIDA ESPERADA

```
Docker Compose version v2.x.x
/etc/nginx/conf.d/:
websocket.conf

/etc/nginx/snippets/:
fastcgi-php.conf  proxy.conf  snakeoil.conf  tls.conf
```

Comprueba antes de seguir

- Docker Compose responde con una versión 2.
- Existen `tls.conf`, `proxy.conf` y `websocket.conf`.
- Tienes al menos 1 GB disponible.
- El `ss` no muestra nada.

Paso 2 Episodio 10

Subdominio y certificado

Por qué: el blog tendrá su propio subdominio, con su certificado.

2.1 · Registro DNS

En el panel DNS de IONOS, crea un registro **A**: nombre de host `blog`, apunta a `TU_IP`, TTL 1 hora. Si usas IPv6, crea también el **AAAA** igual que hiciste para el dominio principal. Comprueba a los pocos minutos:

EN TU PC · POWERSHELL

```
Resolve-DnsName blog.TU_DOMINIO -Server 1.1.1.1
```

2.2 · Sitio HTTP para Let's Encrypt

EN EL SERVIDOR

```
sudo tee /etc/nginx/sites-available/blog.TU_DOMINIO > /dev/null <<'EOF'
server {
    listen 80;
    listen [::]:80;
    server_name blog.TU_DOMINIO;

    location ^~ /.well-known/acme-challenge/ {
        root /var/www/letsencrypt;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}
EOF
sudo ln -s /etc/nginx/sites-available/blog.TU_DOMINIO /etc/nginx/sites-enabled/
sudo nginx -t && sudo systemctl reload nginx
```

2.3 · El certificado

EN EL SERVIDOR

```
sudo certbot certonly --webroot -w /var/www/letsencrypt -d blog.TU_DOMINIO --dry-run
sudo certbot certonly --webroot -w /var/www/letsencrypt -d blog.TU_DOMINIO --cert-name
blog.TU_DOMINIO
```

Primero el ensayo; si dice `The dry run was successful`, el de verdad.

Comprueba antes de seguir

- `Resolve-DnsName` responde `TU_IP`.
- `sudo certbot certificates` muestra el certificado `blog.TU_DOMINIO`.

PARTE B

Los contenedores

Describimos WordPress y su base de datos en un `compose.yaml`, con las contraseñas fuera del archivo y versiones fijas. Al terminar, WordPress responderá dentro del servidor, todavía sin publicar.

Paso 3 Episodio 10

Carpeta y contraseñas

Por qué: las contraseñas de la base de datos no se escriben en el `compose.yaml`: van en un archivo `.env` aparte, que solo tú puedes leer.

EN EL SERVIDOR

```
mkdir -p /opt/apps/wordpress/datos /opt/apps/wordpress/php
cd /opt/apps/wordpress
umask 077
cat > .env <<EOF
CLAVE_DB=$(openssl rand -hex 24)
CLAVE_ROOT_DB=$(openssl rand -hex 24)
EOF
umask 022
ls -la
```

SALIDA ESPERADA

```
drwx------ 4 TU_USUARIO TU_USUARIO 4096 ... .
-rw------- 1 TU_USUARIO TU_USUARIO   92 ... .env
drwxrwxr-x 2 TU_USUARIO TU_USUARIO 4096 ... datos
drwxrwxr-x 2 TU_USUARIO TU_USUARIO 4096 ... php
```

No necesitas apuntar estas contraseñas

A diferencia de otras, estas no las vas a escribir nunca a mano: Compose las lee del `.env` y se las pasa a los contenedores. Lo importante es que el `.env` entre en las copias de seguridad (paso 16). Si quieres verlas: `cat .env`, pero no lo hagas mientras grabas.

Comprueba antes de seguir

`.env` tiene permisos `-rw-------` y `cat .env` muestra dos líneas con 48 caracteres cada una.

Paso 4 Episodio 10

Elegir las versiones

Por qué: como aprendiste en la Guía 03, nada de `latest`: versiones fijas que solo cambias tú, cuando decides actualizar.

4.1 · WordPress

En `hub.docker.com/_/wordpress` → pestaña **Tags**, filtra por `apache`. Verás etiquetas con este formato:

SALIDA ESPERADA

```
7.0.1-php8.4-apache
7.0.1-php8.3-apache
7.0-apache
```

Los números de arriba son un **ejemplo** del formato. Apunta la etiqueta completa que elijas como TU_VERSION_WP.

4.2 · MariaDB

MariaDB publica cada año una versión **LTS** (soporte largo, varios años de parches de seguridad). Usa la LTS más reciente: compruébala en `endoflife.date/mariadb` (columna *LTS*). En `hub.docker.com/_/mariadb` → Tags, busca esa versión con solo dos números (por ejemplo `11.8`). Apúntala como TU_VERSION_MARIADB.

¿Por qué aquí dos números y en WordPress tres?
Con `mariadb:11.8` recibes los parches de seguridad de la 11.8 cada vez que actualizas (`11.8.1` , `11.8.2` …), pero **nunca** saltas solo a otra versión mayor que pudiera cambiar el formato de los datos. En WordPress fijamos los tres números porque el propio WordPress se actualiza por dentro (paso 13): la imagen solo aporta PHP y Apache.

Comprueba antes de seguir
Tienes apuntadas TU_VERSION_WP (formato `X.Y.Z-phpA.B-apache`) y TU_VERSION_MARIADB (formato `X.Y`).

Paso 5 Episodio 10

El compose.yaml

Por qué: el archivo que describe los dos contenedores, cómo se conectan y dónde guardan sus datos.

Créalo con `nano compose.yaml` y pega el contenido, cambiando las dos versiones:

CONTENIDO DEL ARCHIVO · /OPT/APPS/WORDPRESS/COMPOSE.YAML

```yaml
services:
  db:
    image: mariadb:TU_VERSION_MARIADB
    container_name: wordpress-db
    restart: unless-stopped
    environment:
      MARIADB_DATABASE: wordpress
      MARIADB_USER: wordpress
      MARIADB_PASSWORD: ${CLAVE_DB}
      MARIADB_ROOT_PASSWORD: ${CLAVE_ROOT_DB}
    volumes:
      - ./datos/db:/var/lib/mysql
    healthcheck:
      test: ["CMD", "healthcheck.sh", "--connect", "--innodb_initialized"]
      interval: 10s
      timeout: 5s
      retries: 5
      start_period: 30s

  wordpress:
    image: wordpress:TU_VERSION_WP
    container_name: wordpress
    restart: unless-stopped
    depends_on:
      db:
        condition: service_healthy
    ports:
      - "127.0.0.1:8082:80"
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: ${CLAVE_DB}
      WORDPRESS_CONFIG_EXTRA: |
        define('DISALLOW_FILE_EDIT', true);
        define('WP_AUTO_UPDATE_CORE', 'minor');
    volumes:
      - ./datos/html:/var/www/html
      - ./php/subidas.ini:/usr/local/etc/php/conf.d/subidas.ini:ro

  wpcli:
    image: wordpress:cli
    profiles: ["herramientas"]
    user: "33:33"
    depends_on:
      - db
    environment:
      HOME: /tmp
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: ${CLAVE_DB}
    volumes:
      - ./datos/html:/var/www/html
```

El servicio db (MariaDB)

El servicio wordpress

El servicio wpcli

Comprueba antes de seguir

Valida el archivo antes de arrancar nada:

```
EN EL SERVIDOR
cd /opt/apps/wordpress
docker compose config --quiet && echo "compose correcto"
```

Debe responder `compose correcto`. Si sale `yaml: line N`, revisa la sangría de esa línea (solo espacios, de dos en dos).

Paso 6 Episodio 10

Ajustes de PHP

Por qué: por defecto, PHP solo deja subir archivos de 2 MB.

```
EN EL SERVIDOR
tee /opt/apps/wordpress/php/subidas.ini > /dev/null <<'EOF'
upload_max_filesize = 64M
post_max_size = 64M
memory_limit = 256M
max_execution_time = 120
EOF
```

Tres sitios con límite de tamaño

Para subir un archivo grande, los tres tienen que dejarlo pasar: **Nginx** (`client_max_body_size`, paso 8), **PHP** (este archivo) y, a veces, el propio WordPress o el plugin. Si uno se queda corto, la subida falla aunque los otros dos estén bien. Ponemos los tres en 64 MB.

Comprueba antes de seguir

`cat /opt/apps/wordpress/php/subidas.ini` muestra las cuatro líneas.

Paso 7 Episodio 10

Arrancar y comprobar

Por qué: primero descargamos las imágenes, después arrancamos y vemos cómo MariaDB pasa a «sana» antes de que arranque WordPress.

EN EL SERVIDOR

```
cd /opt/apps/wordpress
docker compose pull
docker compose up -d
docker compose ps
```

Espera unos 30 segundos y repite `docker compose ps` hasta ver esto:

SALIDA ESPERADA

```
NAME           IMAGE                          STATUS                    PORTS
wordpress      wordpress:TU_VERSION_WP        Up 20 seconds             127.0.0.1:8082→80/tcp
wordpress-db   mariadb:TU_VERSION_MARIADB     Up 45 seconds (healthy)   3306/tcp
```

Comprueba que WordPress responde por dentro y mira qué ha creado en tus carpetas:

EN EL SERVIDOR

```
curl -I http://127.0.0.1:8082
ls datos/html | head
sudo ls -ld datos/db datos/html
```

SALIDA ESPERADA

```
HTTP/1.1 302 Found
Location: http://127.0.0.1:8082/wp-admin/install.php
...
index.php  license.txt  readme.html  wp-activate.php  wp-admin ...
drwxr-xr-x  5 999      999        4096 ... datos/db
drwxr-xr-x  5 www-data www-data   4096 ... datos/html
```

No abras todavía el asistente desde fuera

El asistente de instalación lo completa **el primero que llegue**: quien lo rellene elige el usuario administrador. Por eso publicaremos la web (paso 8) y la instalaremos (paso 9) seguidos, sin hacer pausas entre uno y otro.

Comprueba antes de seguir

- Los dos contenedores en `Up` y MariaDB en `(healthy)`.
- `curl` da `302` hacia `install.php`.

PARTE C

Publicar e instalar

Publicamos WordPress con HTTPS y lo instalamos en seguida, antes de que nadie más llegue al asistente. Después, los ajustes que conviene hacer el primer día.

Paso 8 Episodio 10

Publicar con Nginx y HTTPS

Por qué: sustituimos el sitio HTTP mínimo del paso 2 por el sitio completo: el mismo bloque HTTP más un bloque HTTPS que reenvía todo a WordPress.

EN EL SERVIDOR

```
sudo tee /etc/nginx/sites-available/blog.TU_DOMINIO > /dev/null <<'EOF'
# HTTP: Let's Encrypt y redirección a HTTPS
server {
    listen 80;
    listen [::]:80;
    server_name blog.TU_DOMINIO;

    location ^~ /.well-known/acme-challenge/ {
        root /var/www/letsencrypt;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

# HTTPS: reenvío a WordPress (contenedor en 127.0.0.1:8082)
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name blog.TU_DOMINIO;

    ssl_certificate     /etc/letsencrypt/live/blog.TU_DOMINIO/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/blog.TU_DOMINIO/privkey.pem;
    include snippets/tls.conf;

    add_header Strict-Transport-Security "max-age=86400" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;
    add_header X-Frame-Options "SAMEORIGIN" always;

    client_max_body_size 64m;

    location / {
        proxy_pass http://127.0.0.1:8082;
        include snippets/proxy.conf;
    }

    access_log /var/log/nginx/blog.access.log;
    error_log  /var/log/nginx/blog.error.log;
}
EOF
sudo nginx -t && sudo systemctl reload nginx
```

Pruébalo desde tu PC:

EN TU PC · POWERSHELL

```
curl.exe -I http://blog.TU_DOMINIO
curl.exe -I https://blog.TU_DOMINIO
```

Comprueba antes de seguir

- HTTP da `301` hacia `https://blog.TU_DOMINIO/`.
- HTTPS da `302` hacia `https://blog.TU_DOMINIO/wp-admin/install.php`. Fíjate en que la dirección empieza por `https`: eso confirma que `X-Forwarded-Proto` funciona.
- Ahora, sin pausa, el paso 9.

`Paso 9` `Episodio 10`

El asistente de instalación

Por qué: WordPress se instala desde el navegador en dos pantallas.

1. Abre `https://blog.TU_DOMINIO`. Elige **Español** y pulsa **Continuar**.
2. **Título del sitio:** el nombre de tu blog (se cambia cuando quieras).
3. **Nombre de usuario:** TU_USUARIO_WP. **Nunca** `admin`, `administrador`, ni tu nombre o el del dominio: es lo primero que prueban los robots. Algo que no se adivine, como un apodo con números.
4. **Contraseña:** deja la que propone WordPress (larga y aleatoria) y guárdala en tu gestor **antes** de seguir.
5. **Tu correo electrónico:** TU_CORREO. Ahí llegan los avisos de seguridad y la recuperación de contraseña (cuando el correo funcione, paso 15).
6. **Visibilidad en los motores de búsqueda:** en el laboratorio, **marca** «Disuadir a los motores de búsqueda». En una web real, déjala sin marcar.
7. Pulsa **Instalar WordPress** y entra con tu usuario y contraseña.

¿Y no faltan los datos de la base de datos?

En una instalación normal, WordPress te pediría el nombre de la base de datos, el usuario y la contraseña. Aquí no, porque la imagen ya los ha escrito en `wp-config.php` con las variables `WORDPRESS_DB_...` del compose. Por eso el asistente va directo a crear el sitio.

Comprueba antes de seguir

- Entras en el panel en `https://blog.TU_DOMINIO/wp-admin/` con el candado del navegador.
- Tu usuario no es `admin` y la contraseña está en tu gestor.

`Paso 10` `Episodio 10`

Primeros ajustes y limpieza

Por qué: WordPress viene con contenido de ejemplo y algún plugin que no necesitas.

10.1 · Ajustes

Dónde	Qué poner	Por qué
Ajustes → Generales	Zona horaria: tu ciudad (por ejemplo *Madrid* o *Canarias*). Comprueba que las dos direcciones empiezan por `https://`	Que las fechas de las entradas y las tareas programadas vayan con tu hora

Dónde	Qué poner	Por qué
Ajustes → Enlaces permanentes	**Nombre de la entrada** → Guardar	Direcciones legibles (`/mi-primera-entrada/`) en vez de `/?p=123` : mejor para leerlas, compartirlas y para Google
Ajustes → Comentarios	Si no vas a usar comentarios, desmarca «Permitir que se publiquen comentarios en nuevas entradas»	Los comentarios son la principal fuente de spam
Usuarios → Perfil	Rellena Nombre y, en «Mostrar este nombre públicamente como», elige tu nombre, **no** el usuario	Así tu nombre de usuario no aparece en las entradas: es la mitad de lo que necesita quien quiere entrar

10.2 · Limpieza

- **Entradas:** borra «¡Hola, mundo!». **Páginas:** borra la página de ejemplo. Vacía la papelera.
- **Plugins:** desactiva y borra *Hello Dolly* y, si no lo vas a usar, *Akismet*.
- **Apariencia → Temas:** deja el tema que uses y **uno** de los temas oficiales por defecto (sirve de repuesto si el tuyo falla). Borra el resto.
- **Herramientas → Salud del sitio:** léelo. Es normal que avise de que no puede enviar correos (lo arreglamos en el paso 15).

Plugins: los justos y de confianza

Cada plugin es código de un tercero con acceso total a tu web. Antes de instalar uno, mira en wordpress.org cuándo se actualizó por última vez, cuántas instalaciones activas tiene y si es compatible con tu versión. Un plugin abandonado es la forma más habitual de que hackeen un WordPress.

Comprueba antes de seguir

Las direcciones de las entradas son legibles, no quedan contenido de ejemplo ni plugins sin usar, y **Salud del sitio** no muestra errores críticos salvo el del correo.

PARTE D

Proteger

Las cuatro medidas que más ataques paran en un WordPress: cerrar lo que no usas, frenar y bloquear a quien prueba contraseñas, mantenerlo al día y tener una forma de arreglarlo si no puedes entrar al panel.

Paso 11 Episodio 10

Cerrar las puertas que no usas

Por qué: tres rutas de WordPress que casi nadie necesita y que los robots atacan a diario, y un límite de velocidad para el formulario de acceso.

11.1 · La zona de límite de velocidad

EN EL SERVIDOR

```
sudo tee /etc/nginx/conf.d/wordpress-limites.conf > /dev/null <<'EOF'
# Límite de intentos en el formulario de acceso de WordPress
limit_req_zone $binary_remote_addr zone=wplogin:10m rate=10r/m;
limit_req_status 429;
EOF
```

11.2 · El sitio con las protecciones

Sustituye el sitio del paso 8 por esta versión. Es el mismo archivo con el bloque «Protecciones para WordPress» añadido:

EN EL SERVIDOR

```
sudo tee /etc/nginx/sites-available/blog.TU_DOMINIO > /dev/null <<'EOF'
# HTTP: Let's Encrypt y redirección a HTTPS
server {
    listen 80;
    listen [::]:80;
    server_name blog.TU_DOMINIO;

    location ^~ /.well-known/acme-challenge/ {
        root /var/www/letsencrypt;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

# HTTPS: reenvío a WordPress (contenedor en 127.0.0.1:8082)
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name blog.TU_DOMINIO;

    ssl_certificate     /etc/letsencrypt/live/blog.TU_DOMINIO/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/blog.TU_DOMINIO/privkey.pem;
    include snippets/tls.conf;

    add_header Strict-Transport-Security "max-age=86400" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;
    add_header X-Frame-Options "SAMEORIGIN" always;

    client_max_body_size 64m;

    # Protecciones para WordPress
    location = /xmlrpc.php {
        deny all;
    }
    location ~* ^/wp-content/uploads/.*\.php$ {
        deny all;
    }
    location ~ /\.(?!well-known) {
        deny all;
    }
    location = /wp-login.php {
        limit_req zone=wplogin burst=5 nodelay;
        proxy_pass http://127.0.0.1:8082;
        include snippets/proxy.conf;
    }

    location / {
        proxy_pass http://127.0.0.1:8082;
        include snippets/proxy.conf;
    }

    access_log /var/log/nginx/blog.access.log;
```

```
    error_log  /var/log/nginx/blog.error.log;
}
EOF
sudo nginx -t && sudo systemctl reload nginx
```

> **El orden de los location en Nginx**
>
> Nginx no elige el primer `location` que coincide, sino el más concreto: primero los exactos (`=`), después las expresiones regulares (`~` , `~*`) en el orden en que aparecen y, por último, el prefijo más largo (`/`). Por eso `/wp-login.php` y las prohibiciones se aplican aunque `location /` también coincida.

Comprueba cada protección desde tu PC:

EN TU PC · POWERSHELL

```
curl.exe -s -o NUL -w "%{http_code}\n" https://blog.TU_DOMINIO/xmlrpc.php
curl.exe -s -o NUL -w "%{http_code}\n" https://blog.TU_DOMINIO/wp-content/uploads/prueba.php
curl.exe -s -o NUL -w "%{http_code}\n" https://blog.TU_DOMINIO/.htaccess
1..20 | ForEach-Object { curl.exe -s -o NUL -w "%{http_code} " https://blog.TU_DOMINIO/wp-
login.php }
```

SALIDA ESPERADA

```
403
403
403
200 200 200 200 200 200 429 429 429 429 429 429 429 429 429 429 429 429 429 429
```

Las primeras peticiones pasan (la tasa más la ráfaga) y el resto recibe `429`. Pasado un minuto, vuelve a dejarte entrar: el límite frena, no bloquea. El bloqueo de verdad lo pone fail2ban, en el siguiente paso.

> **Comprueba antes de seguir**
>
> - `sudo nginx -t` sin errores.
> - Los tres primeros `curl` dan `403` y la ráfaga acaba en `429`.
> - La web y el panel siguen funcionando con normalidad.

Paso 12 Episodio 10

fail2ban para los intentos de acceso

Por qué: el límite de velocidad frena a los robots rápidos, pero los hay lentos: prueban una contraseña cada pocos segundos, durante días.

12.1 · Cómo se ve un intento fallido

Cuando alguien envía el formulario con una contraseña incorrecta, WordPress responde `200` (vuelve a mostrar el formulario con el error). Cuando acierta, responde `302` (le redirige al panel). En el registro de Nginx se ve así:

SALIDA ESPERADA

```
203.0.113.7 - - [02/Nov/2026:18:04:11 +0000] "POST /wp-login.php HTTP/2.0" 200 4521 "-"
"Mozilla/5.0 ..."
198.51.100.4 - - [02/Nov/2026:18:05:02 +0000] "POST /wp-login.php HTTP/2.0" 302 0 "-"
"Mozilla/5.0 ..."
```

La primera línea es un fallo y la segunda, un acceso correcto. La regla es: **`POST` a `/wp-login.php` con respuesta `200` = intento fallido**.

12.2 · El filtro y la jail

EN EL SERVIDOR

```
sudo tee /etc/fail2ban/filter.d/wordpress-login.conf > /dev/null <<'EOF'
[Definition]
failregex = ^<HOST> \S+ \S+ \[[^\]]+\] "POST /wp-login\.php HTTP/[0-9.]+" 200
ignoreregex =
EOF

sudo tee /etc/fail2ban/jail.d/wordpress.conf > /dev/null <<'EOF'
[wordpress-login]
enabled  = true
port     = http,https
filter   = wordpress-login
logpath  = /var/log/nginx/blog.access.log
backend  = auto
maxretry = 5
findtime = 10m
EOF
```

12.3 · Probar el filtro antes de activarlo

Entra en `https://blog.TU_DOMINIO/wp-login.php` y prueba **dos veces** con una contraseña incorrecta (desde tu casa: tu IP está en `ignoreip`, así que no te bloqueará). Después:

EN EL SERVIDOR

```
sudo fail2ban-regex /var/log/nginx/blog.access.log /etc/fail2ban/filter.d/wordpress-login.conf
| grep -E "^Lines|matched"
sudo systemctl restart fail2ban
sudo fail2ban-client status wordpress-login
```

SALIDA ESPERADA

```
Lines: 214 lines, 0 ignored, 2 matched, 212 missed
...
Status for the jail: wordpress-login
├─ Filter
|  ├─ Currently failed: 1
|  `- File list:  /var/log/nginx/blog.access.log
`- Actions
   ├─ Currently banned: 0
```

Si te bloqueas a ti mismo

Si haces la prueba desde el móvil con datos (otra IP), al sexto fallo esa IP quedará bloqueada: la web no cargará en el móvil. Desbloquéala con `sudo fail2ban-client set wordpress-login unbanip LA_IP` (la ves en `status`). No uses `unban --all`: liberaría también a los atacantes de verdad.

Comprueba antes de seguir

- `fail2ban-regex` muestra tus intentos como `matched`.
- `fail2ban-client status` lista la jail `wordpress-login` junto a `sshd`.

Paso 13 Episodio 10

Actualizaciones

Por qué: en WordPress con Docker hay **dos cosas distintas** que actualizar, y conviene no confundirlas: el propio WordPress (con sus temas y plugins) y la imagen del contenedor (PHP y Apache).

Qué	Dónde vive	Cómo se actualiza
WordPress (núcleo)	En tu carpeta `datos/html`	Solo, en las versiones menores (`WP_AUTO_UPDATE_CORE`). Las mayores, desde **Escritorio → Actualizaciones**, después de una copia (paso 16)
Plugins y temas	En `datos/html/wp-content`	En **Plugins**, pulsa «Activar actualizaciones automáticas» en cada uno de confianza. Los demás, a mano, cuando WordPress avise
PHP y Apache	En la imagen del contenedor	Cambiando `TU_VERSION_WP` en el compose por una más nueva, como en la Guía 03

Cambiar la imagen no actualiza WordPress

La imagen copia WordPress en `datos/html` **solo la primera vez**, cuando la carpeta está vacía. Después, WordPress se actualiza a sí mismo dentro de esa carpeta. Por eso, si cambias la etiqueta de la imagen de la 7.0.1 a la 7.0.2, lo que cambias es PHP y Apache; el WordPress de tu carpeta sigue igual hasta que él mismo se actualice. Es normal que no coincidan los números.

13.1 · Actualizar la imagen (PHP y Apache)

EN EL SERVIDOR

```
cd /opt/apps/wordpress
sudo /usr/local/bin/copia-wordpress.sh
nano compose.yaml
docker compose pull
docker compose up -d
docker compose ps
```

Comprueba antes de seguir

Sabes distinguir qué se actualiza solo (WordPress menor), qué activas tú (plugins de confianza) y qué cambias en el compose (la imagen).

Paso 14 Episodio 10

WP-CLI: WordPress desde la terminal

Por qué: si un día un plugin rompe la web y no puedes entrar al panel, o olvidas la contraseña, WP-CLI te deja arreglarlo desde el servidor.

EN EL SERVIDOR

```
cd /opt/apps/wordpress
docker compose run --rm wpcli wp core version
docker compose run --rm wpcli wp plugin list
docker compose run --rm wpcli wp user list --fields=ID,user_login,roles
```

Los rescates más útiles

Problema	Comando
La web da error después de instalar o actualizar un plugin	`docker compose run --rm wpcli wp plugin deactivate NOMBRE`
No sabes cuál es: desactivarlos todos	`docker compose run --rm wpcli wp plugin deactivate --all`
Olvidaste la contraseña (te la pide sin que quede en el historial)	`docker compose run --rm wpcli wp user update TU_USUARIO_WP --prompt=user_pass`
Actualizar todos los plugins	`docker compose run --rm wpcli wp plugin update --all`

Comprueba antes de seguir

`wp core version` muestra tu versión de WordPress y `wp user list`, solo tu usuario. Si sale `Permission denied`, revisa la línea `user: "33:33"` del compose.

PARTE E

Mantener

Lo que hace que un WordPress funcione meses sin sustos: que pueda avisarte por correo, copias automáticas y una comprobación final antes de dar la guía por terminada.

Paso 15 Episodio 10

Que WordPress pueda enviar correos

Por qué: WordPress envía correos para recuperar la contraseña, avisar de actualizaciones o de comentarios.

1. Necesitas un buzón para enviar (por ejemplo `web@TU_DOMINIO`). Si ya tienes tu servidor de correo (Temporada 4), créalo ahí con una contraseña propia, como el buzón de avisos de la Guía 04. Si no, usa el buzón de tu proveedor de correo.
2. En WordPress, **Plugins** → **Añadir nuevo**, busca **WP Mail SMTP** (de WPForms, con millones de instalaciones) e instálalo y actívalo.
3. En su asistente elige **Other SMTP** y rellena: servidor `mail.TU_DOMINIO` (o el de tu proveedor), cifrado **TLS**, puerto **587**, autenticación activada, usuario `web@TU_DOMINIO` y su contraseña.
4. En **From Email**, el mismo buzón, y marca **Force From Email** para que todos los correos salgan desde él.
5. En la pestaña **Email Test**, envía un correo de prueba a tu dirección.

La contraseña del buzón queda en la base de datos

El plugin guarda la contraseña del buzón en la base de datos de WordPress. Por eso usamos un buzón **solo para la web**: si un día alguien entrara en WordPress, solo podría enviar correos desde ese buzón, no leer los tuyos. El plugin también permite ponerla en `wp-config.php`; con Docker, lo más sencillo es dejarla en el panel.

Comprueba antes de seguir

- Te llega el correo de prueba (mira también en spam).
- **Herramientas** → **Salud del sitio** ya no avisa del correo.

Paso 16 Episodio 10

Copias de seguridad automáticas

Por qué: lo mismo que hicimos con Chatwoot: un script que copia la base de datos y los archivos cada noche, guarda 14 días y borra lo más antiguo, con un temporizador de systemd que lo lanza solo.

16.1 · El script

EN EL SERVIDOR

```
sudo tee /usr/local/bin/copia-wordpress.sh > /dev/null <<'EOF'
#!/bin/bash
# Copia diaria de WordPress · Ebenezer IT
set -euo pipefail
umask 077

DESTINO=/var/backups/wordpress
FECHA=$(date +%F)
DIAS=14

mkdir -p "$DESTINO"
cd /opt/apps/wordpress

# 1. Base de datos: entradas, páginas, usuarios, ajustes
docker compose exec -T db sh -c 'MYSQL_PWD="$MARIADB_ROOT_PASSWORD" mariadb-dump -u root --
single-transaction --databases wordpress' \
    | gzip > "$DESTINO/bd-$FECHA.sql.gz"
gzip -t "$DESTINO/bd-$FECHA.sql.gz"

# 2. Archivos: temas, plugins, subidas y configuración
tar -czf "$DESTINO/archivos-$FECHA.tar.gz" -C /opt/apps/wordpress \
    .env compose.yaml php datos/html/wp-content datos/html/wp-config.php

# 3. Borrar las copias de más de $DIAS días
find "$DESTINO" -type f -mtime +$DIAS -delete

echo "Copia terminada: $(ls "$DESTINO" | wc -l) archivos, $(du -sh "$DESTINO" | cut -f1) en
total"
EOF
sudo chmod 700 /usr/local/bin/copia-wordpress.sh
```

16.2 · Pruébalo y mira dentro

EN EL SERVIDOR

```
sudo /usr/local/bin/copia-wordpress.sh
sudo ls -lh /var/backups/wordpress/
sudo zcat /var/backups/wordpress/bd-$(date +%F).sql.gz | grep -c "CREATE TABLE"
```

Comprueba antes de seguir

Sale «Copia terminada: 2 archivos», los dos archivos tienen permisos `-rw-------` de `root`, y el último comando cuenta **12** tablas o más (las de una instalación limpia de WordPress).

16.3 · Que se haga sola cada noche

EN EL SERVIDOR

```
sudo tee /etc/systemd/system/copia-wordpress.service > /dev/null <<'EOF'
[Unit]
Description=Copia de seguridad de WordPress
After=docker.service
Requires=docker.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/copia-wordpress.sh
EOF

sudo tee /etc/systemd/system/copia-wordpress.timer > /dev/null <<'EOF'
[Unit]
Description=Copia diaria de WordPress a las 03:45

[Timer]
OnCalendar=*-*-* 03:45:00
Persistent=true

[Install]
WantedBy=timers.target
EOF

sudo systemctl daemon-reload
sudo systemctl enable --now copia-wordpress.timer
systemctl list-timers copia-wordpress.timer
```

Cómo se restaura

Para devolver la base de datos al estado de una copia (por ejemplo, la de ayer):

EN EL SERVIDOR

```
cd /opt/apps/wordpress
sudo zcat /var/backups/wordpress/bd-FECHA.sql.gz | docker compose exec -T db sh -c
'MYSQL_PWD="$MARIADB_ROOT_PASSWORD" mariadb -u root'
```

Y los archivos, descomprimiendo `archivos-FECHA.tar.gz` en `/opt/apps/wordpress` con `sudo tar -xzf ... -C /opt/apps/wordpress`. Lo practicaremos de verdad en el episodio de restaurar copias (Temporada 8).

Estas copias están en el mismo servidor

Te protegen de errores, de un plugin que rompe la web o de un hackeo que borra contenido. No te protegen de perder el servidor entero. Sacarlas fuera del VPS es la Temporada 8. Mientras tanto, las instantáneas de IONOS cubren ese caso.

Comprueba antes de seguir

`list-timers` muestra la próxima copia mañana a las 03:45.

Paso 17 Episodio 10

Comprobación final e instantánea

Por qué: repaso completo, reinicio para ver que todo vuelve solo, y la instantánea que será el punto de partida de la siguiente guía.

EN EL SERVIDOR

```
cd /opt/apps/wordpress
docker compose ps
sudo ss -tlpn | grep -E ':(8082|3306) '
sudo nginx -t
sudo fail2ban-client status wordpress-login | grep "File list"
systemctl list-timers copia-wordpress.timer --no-pager
sudo certbot renew --dry-run
```

Comprobación	Resultado correcto
Contenedores	`wordpress` en `Up` y `wordpress-db` en `Up (healthy)`
Puertos	`127.0.0.1:8082` y **nada** en el 3306 del servidor
Nginx	`syntax is ok`
fail2ban	La jail lee `/var/log/nginx/blog.access.log`
Copias	Próxima ejecución a las 03:45
Renovación	Todos los certificados, incluido `blog.TU_DOMINIO`, en `succeeded`
Desde tu PC	`https://blog.TU_DOMINIO` carga con candado; `/xmlrpc.php` da 403; `http://TU_IP:8082` no responde
Panel	Salud del sitio sin errores críticos; correo de prueba recibido

Reinicia y comprueba que todo arranca solo:

EN EL SERVIDOR

```
sudo reboot
# ...espera un minuto, ssh lab, y:
docker ps --format '{{.Names}}  {{.Status}}'
```

Después, en IONOS, crea la instantánea `05-wordpress`.

Comprueba antes de seguir

Tienes un WordPress con HTTPS, protegido contra los ataques más comunes, que se actualiza en lo importante y se copia cada noche. La siguiente temporada monta tu propia VPN y, con ella, podrás dejar el panel de WordPress accesible solo desde tus dispositivos.

ANEXOS

Para cuando algo no sale

Orden de búsqueda: `docker compose ps` → `docker compose logs wordpress --tail 50` → `docker compose logs db --tail 50` → `curl -I http://127.0.0.1:8082` → el `blog.error.log` de Nginx.

Anexo A · Problemas frecuentes

Lo que ves	Qué pasa	Qué hacer
«Error al establecer una conexión con la base de datos»	WordPress no puede entrar en MariaDB: aún está arrancando, o la contraseña no coincide	`docker compose ps`: ¿`db` está `(healthy)`? Si sí, mira `docker compose logs db`. Si cambiaste `CLAVE_DB` en el `.env` después del primer arranque, MariaDB sigue con la antigua (ver fila siguiente)
Cambié una contraseña del `.env` y no hace nada	MariaDB solo lee `MARIADB_PASSWORD` y `MARIADB_ROOT_PASSWORD` la **primera** vez, con la carpeta vacía	En una instalación de pruebas sin contenido: `docker compose down`, `sudo rm -rf datos/db datos/html` y repite desde el paso 7. Con contenido real, la contraseña se cambia dentro de MariaDB; pide ayuda
`ERR_TOO_MANY_REDIRECTS`	WordPress cree que está en HTTP	Comprueba que el sitio incluye `snippets/proxy.conf` (`X-Forwarded-Proto`) y que en Ajustes → Generales las dos direcciones empiezan por `https://`
Las imágenes o los estilos no cargan; la consola del navegador dice *mixed content*	Alguna dirección guardada empieza por `http://`	`docker compose run --rm wpcli wp search-replace 'http://blog.TU_DOMINIO' 'https://blog.TU_DOMINIO' --dry-run` y, si lo que muestra es correcto, repítelo sin `--dry-run`
413 Request Entity Too Large al subir	Nginx corta la subida	`client_max_body_size` del sitio (paso 8)
«El archivo subido excede la directiva upload_max_filesize»	PHP corta la subida	`php/subidas.ini` (paso 6) y `docker compose up -d --force-recreate wordpress`
429 al entrar en `wp-login.php`	Has superado el límite de velocidad (paso 11)	Espera un minuto. Si te pasa en uso normal, sube `rate` a `20r/m`
La web no carga desde una IP concreta (por ejemplo, tu móvil)	fail2ban ha bloqueado esa IP	`sudo fail2ban-client status wordpress-login` y `sudo fail2ban-client set wordpress-login unbanip LA_IP`
`fail2ban-regex` da `0 matched`	El formato del registro no encaja con el filtro	Mira una línea real con `sudo grep wp-login /var/log/nginx/blog.access.log | tail -3` y compárala con la del paso 12.1
Pantalla blanca o «Ha habido un error crítico»	Casi siempre, un plugin o tema	`docker compose logs wordpress --tail 50` y desactívalo con WP-CLI (paso 14)

Lo que ves	Qué pasa	Qué hacer
WP-CLI: `Permission denied` o `Error: This does not seem to be a WordPress installation`	Usuario distinto al de los archivos, o falta el volumen	Revisa `user: "33:33"` y `volumes` del servicio `wpcli`
Los correos de WordPress no llegan	No hay SMTP configurado o los datos son incorrectos	Paso 15; la pestaña **Email Log** o **Email Test** de WP Mail SMTP muestra el error exacto

Anexo B · Puertos al terminar esta guía

Puerto	Servicio	Accesible desde	Qué lo protege
443 → `blog.TU_DOMINIO`	WordPress (a través de Nginx)	Internet	HTTPS, cabeceras, límite en wp-login, fail2ban, rutas prohibidas
127.0.0.1:8082	WordPress (contenedor)	Solo el propio servidor (Nginx)	Publicado en 127.0.0.1
3306	MariaDB	Solo la red interna del proyecto	Sin publicar + contraseñas

No se abre ningún puerto nuevo en el firewall de IONOS ni en UFW.

Lista de puertos internos

Puerto interno	Aplicación	Subdominio
127.0.0.1:8081	IT-Tools	`herramientas.TU_DOMINIO`
127.0.0.1:8082	WordPress	`blog.TU_DOMINIO`
127.0.0.1:8083	(siguiente aplicación)	

Anexo C · Chuleta de WordPress con Docker

Para...	Comando (en `/opt/apps/wordpress`)
Estado	`docker compose ps`
Registros de WordPress / de la base de datos	`docker compose logs wordpress --tail 50` · `docker compose logs db --tail 50`
Reiniciar WordPress	`docker compose restart wordpress`
Aplicar cambios del compose o de `subidas.ini`	`docker compose up -d --force-recreate wordpress`
Versión, plugins y usuarios	`docker compose run --rm wpcli wp core version` · `wp plugin list` · `wp user list`
Desactivar un plugin que rompe la web	`docker compose run --rm wpcli wp plugin deactivate NOMBRE`
Cambiar tu contraseña	`docker compose run --rm wpcli wp user update TU_USUARIO_WP --prompt=user_pass`
Copia ahora	`sudo /usr/local/bin/copia-wordpress.sh`
Ver las copias	`sudo ls -lh /var/backups/wordpress/`
Intentos bloqueados	`sudo fail2ban-client status wordpress-login`
Quién está probando contraseñas	`sudo grep 'POST /wp-login.php' /var/log/nginx/blog.access.log | awk '{print $1}' | sort | uniq -c | sort -rn | head`

Si te atascas

Pega: la salida de `docker compose ps`, las últimas 50 líneas del registro del servicio que falla y, si el problema es al entrar por el navegador, el error que ves y las últimas líneas de `blog.error.log`. No pegues el contenido del `.env`.