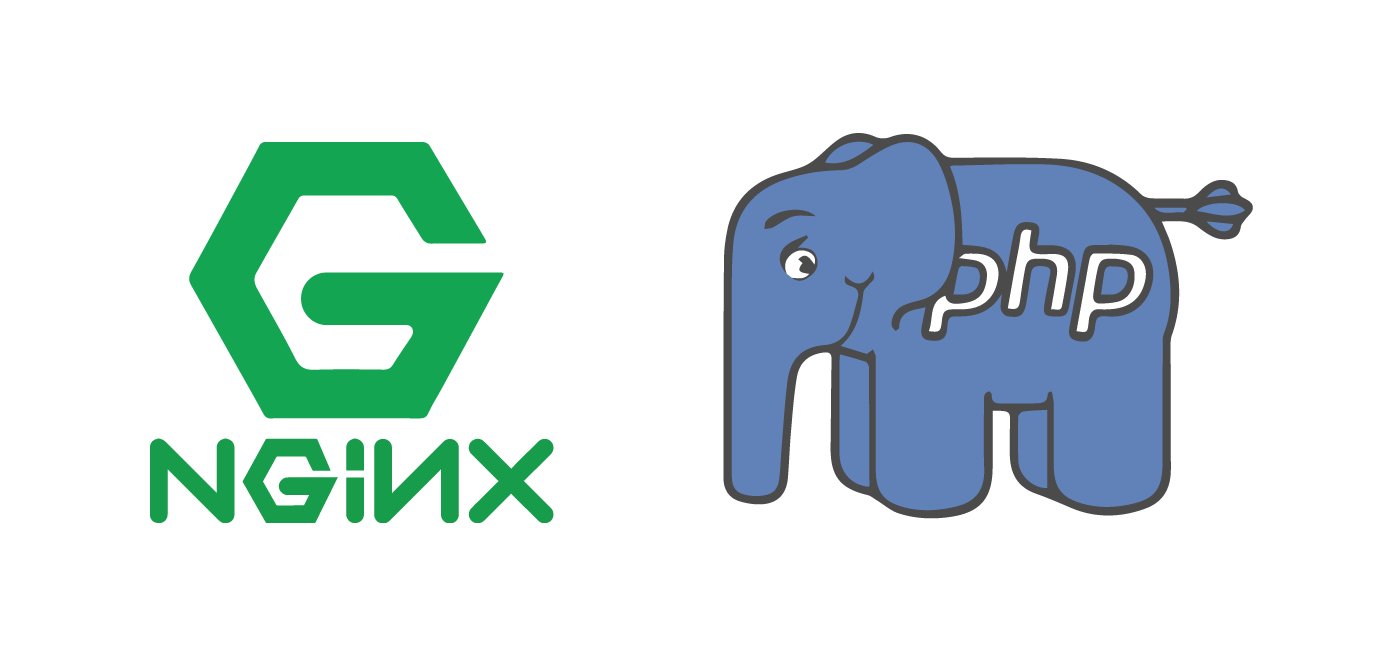
**Servidor Nginx proxyPass + PhP FPM**



| **Revisor** | **Estado** | **Fecha** | **Notas** |
| --- | --- | --- | --- |
| [dalosa@eeartesanos.com](mailto:dalosa@eeartesanos.com) | **Aprobada** | **15/02(24** | Configuración de Servidor Nginx proxyPass + PhP FPM |
| [Hasna Zalmat](mailto:hasnaehasnae063@gmail.com) | **Aprobada** | **15/02/24** | Actualización detalles. |
|  | **No iniciada** |  |  |

**Índice**

[**Descripción 3**](#_ubvvot6rluqf)

[Finalidad del servicio: 3](#_r8kj99qokuqn)

[Máquinas en las que se aloja: 4](#_xv7hue2owhdf)

[Paquetes usados y versión de los mismos: 4](#_4sp9bxllhtcp)

[**Configuración 5**](#_hhlhlh8o2fx5)

[Configuración previa: 5](#_a6ney3xb3a8j)

[Configuración del servicio: 7](#_km4rqbb4du1r)

[**Verificación del funcionamiento 11**](#_xy3pjf2ij9ex)

[Pruebas y comprobaciones 11](#_egw2txk054re)

[**Fuentes 12**](#_3ynhol5t4yh5)

# **Descripción**

## **Finalidad del servicio:**

Nginx, un popular servidor web y proxy inverso, no tiene integrado soporte nativo para PHP, un lenguaje de programación ampliamente utilizado para crear contenido dinámico en la web. En lugar de eso, se utiliza PHP-FPM (PHP FastCGI Process Manager) como un complemento para manejar el procesamiento de scripts PHP. Esta separación de responsabilidades permite una mayor eficiencia y flexibilidad en el manejo de solicitudes PHP.

Cuando configuras Nginx junto con PHP-FPM, estás creando un entorno donde Nginx se encarga de servir archivos estáticos y actúa como un intermediario entre los clientes y PHP-FPM. Cuando un cliente solicita una página PHP, Nginx envía la solicitud al proceso de PHP-FPM correspondiente, que se encarga de ejecutar el script PHP y devolver el resultado a Nginx, que luego lo entrega al cliente.

En resumen, al utilizar Nginx junto con PHP-FPM, puedes crear un entorno de servidor web eficiente y escalable que es capaz de manejar tanto contenido estático como dinámico de manera óptima.

## **Máquinas en las que se aloja:**



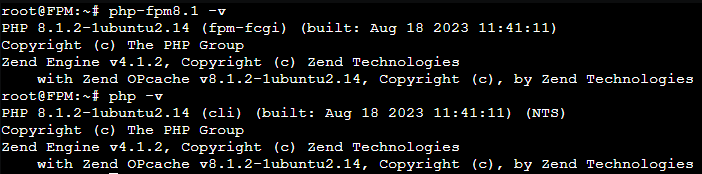
Para esta práctica necesitaremos 2 contenedores, aunque también se pueden utilizar VMs, pero para agilizar nosotros nos hemos decantado por los CTs. Tenemos “NginxPHP” que será nuestro proxy, y “FPM” (php-fpm) será el contenedor que por detrás esté corriendo php8.1-fpm.

Además, hemos configurado los “Ids” de los CTs de manera que sean la dirección de host de la IP correspondiente, o sea, “NginxPHP” será la IP “172.16.1.**140**”, y “FPM” será la IP “172.16.1.**141**”.

## **Paquetes usados y versión de los mismos:**



Tenemos la última versión de Nginx instalado.

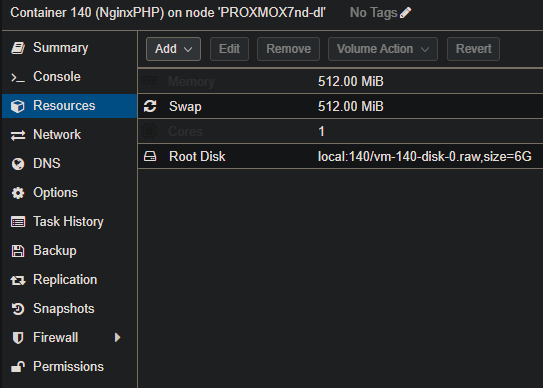


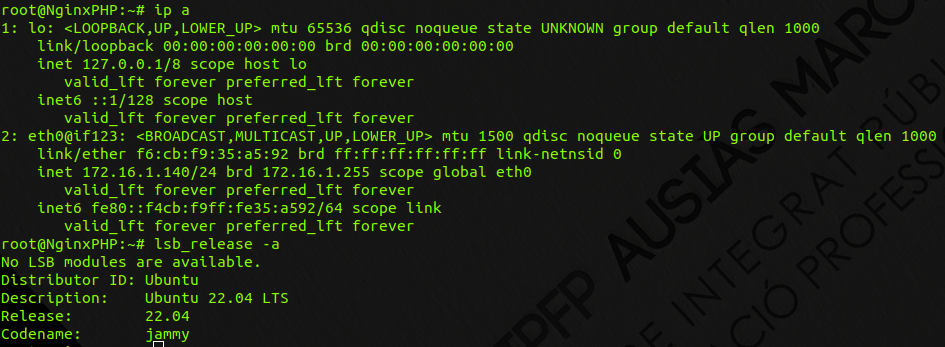
Tenemos la versión 8.1.2 de php y php-fpm8.1.

# **Configuración**

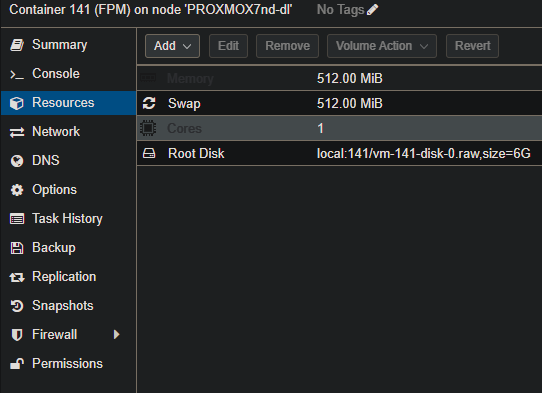
## **Configuración previa:**

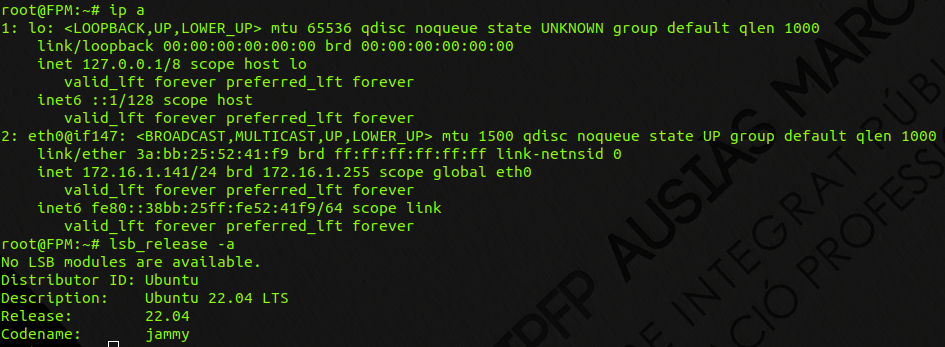
**NginxPHP:**



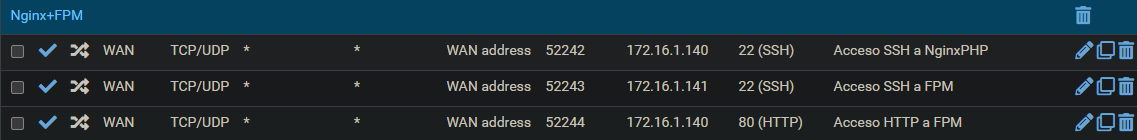


**FPM:**





**Pfsense:**



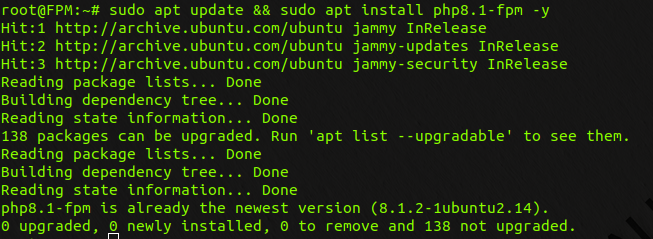
Para poder empezar a trabajar con los contenedores, yo he creado unas reglas en el Pfsense para poder operar desde ssh…



Y esta línea del archivo sshd\_config en la ruta “/etc/ssh/” hay que editarlo para que quede de la manera que se muestra en la imagen, porque de lo contrario no podremos conectarnos por ssh a nuestros CTs puesto que accedemos como root y necesitamos este permiso especial.

## **Configuración del servicio:**

**FPM:**

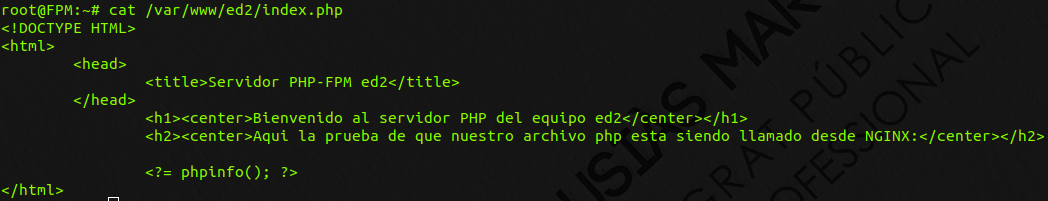


Actualizamos repositorios en nuestro CT FPM con “sudo apt update” e instalamos el paquete php8.1-fpm.



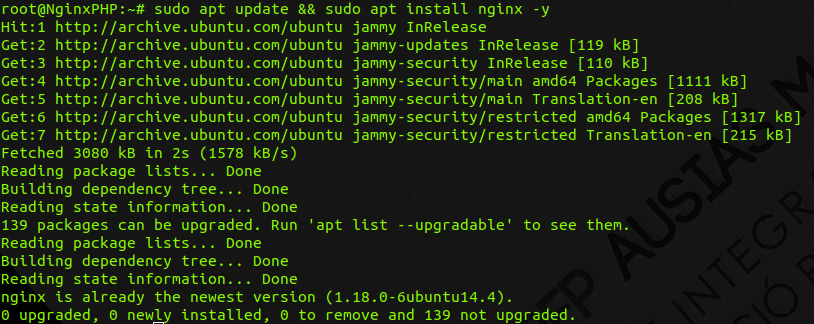
A continuación, editamos el fichero en la ruta “/etc/php/8.1/fpm/pool.d/www.conf” de manera que el parámetro “listen = 127.0.0.1” esté como “listen = “0.0.0.0:9000””.

Esto permite conexión remota. El servidor Nginx servirá el contenido estático y se comunicará con el PHP-FPM para manejar contenido dinámico. Por eso es necesario que PHP-FPM escuche en una interfaz de red accesible desde el servidor Nginx.

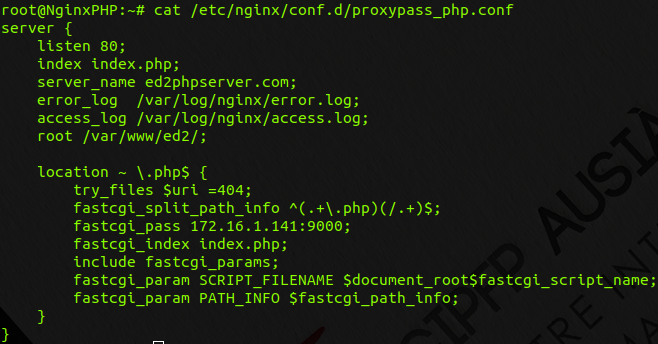


Breve página montada en el servidor FPM en “/var/www/ed2” mostrando un mensaje en texto plano y la configuración php del server mezclando php y html para demostrar que sirve una página web con php.

**NginxPHP:**



Actualizamos los repositorios del sistema con “sudo apt update” e instalamos Nginx.



Aquí el archivo de configuración de nuestro Nginx, donde le indicamos que escuche por el puerto 80 (HTTP), que busque el index llamado index.php, el server\_name será ed2phpserver.com, luego las rutas donde se alojarán nuestros registros log y error, y la ruta donde estará nuestro index.php “/var/www/ed2”.

**Análisis de los parámetros empleados en la configuración**

**location ~ \.php$:** Aplicará a las solicitudes que coincidan con el patrón de archivos PHP (cualquier archivo que termine en .php). La tilde seguida de una barra invertida (~ \) indica que se utilizará una expresión regular para hacer coincidir la ubicación.

**try\_files $uri =404:** Esta directiva indica a Nginx que intente servir primero el archivo solicitado directamente. Si el archivo no se encuentra, Nginx devolverá un error 404. Esto proporciona una capa adicional de seguridad contra posibles ataques a través de archivos PHP maliciosos.

**fastcgi\_split\_path\_info ^(.+\.php)(/.+)$:** Divide la información de la ruta de la solicitud PHP en dos partes: el nombre del archivo PHP y cualquier información de la ruta adicional. Esto es necesario para manejar correctamente las solicitudes PHP que incluyen parámetros adicionales en la URL.

**fastcgi\_pass 172.16.1.141:9000:** Esta directiva especifica la dirección IP y el puerto donde se encuentra el servidor PHP-FPM.

**fastcgi\_index index.php:** Esto establece el archivo de índice predeterminado que se utilizará si no se especifica un archivo en la solicitud.

**include fastcgi\_params:** Esta línea incluye un archivo de configuración que contiene parámetros comunes necesarios para el funcionamiento de FastCGI.

**fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name:** Esta línea define el nombre del script PHP que se va a ejecutar. **$document\_root** se expande al directorio raíz del servidor, y **$fastcgi\_script\_name** se expande al nombre del archivo PHP solicitado.

**fastcgi\_param PATH\_INFO $fastcgi\_path\_info:** Esta línea define el valor de la variable PATH\_INFO, que contiene información adicional de la ruta de la solicitud. Esto es útil para aplicaciones web que utilizan rutas limpias o URL amigables.



Ahora hemos creado un archivo vacío “index.php” en la misma ruta que está en el servidor FPM, esto es muy importante, porque aunque este vacío tiene que estar en el Nginx, porque si no, este daría fallo ya que no encontraría la ruta.

Hay que tener en cuenta que el archivo de Nginx puede estar vacío, lo cual nos interesa ya que este no es procesado y de esta manera no se consume tanta RAM. Los archivos del servidor PHP-FPM son los que deben estar con la web con php.



Y por último en la configuración de Nginx, añadir al “hosts” el nombre del servidor apuntando a la IP adecuada (En este caso a si mismo, aunque este último paso no haria falta).

**FPM y NginxPHP:**

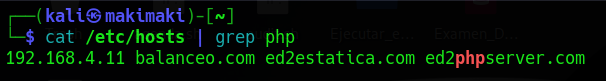




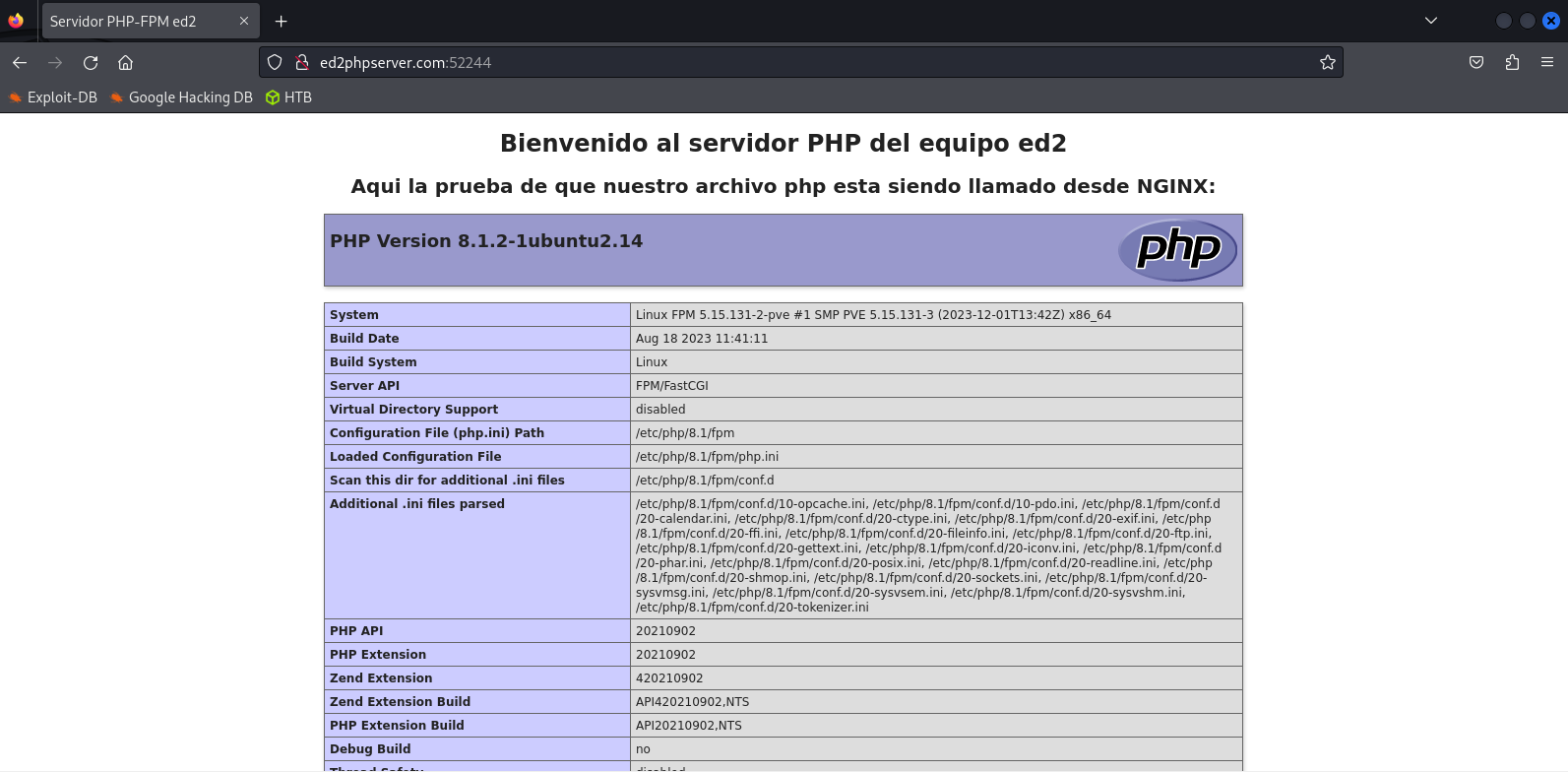
Reiniciamos ambos procesos para aplicar los cambios.

# **Verificación del funcionamiento**

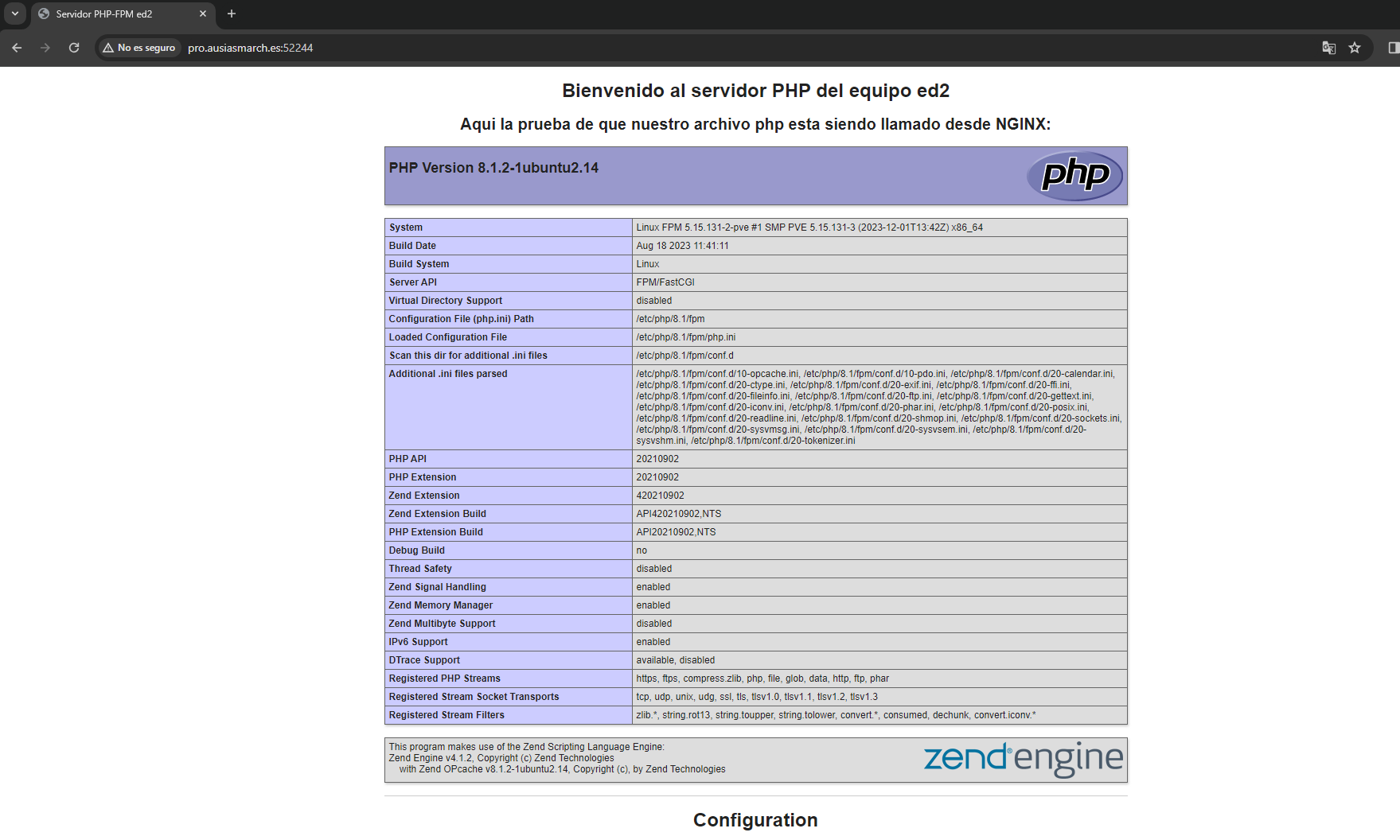
## **Pruebas y comprobaciones**



En nuestra máquina cliente Kali Linux, editamos el fichero de hosts apuntando a la IP que esta alojado nuestro servidor Nginx y agregando como se llama nuestro servidor para que nuestro sistema lo “conozca”.



Y aquí la prueba de que todo funcione correctamente!! =).



Como alternativa se puede acceder desde un navegador externo a la red local y se seguiría teniendo acceso.

# **Fuentes**

Stackoverflow —>

<https://stackoverflow.com/questions/47946739/how-to-configure-nginx-with-php-fpm>