

# NGINX

---

2DAW DESPLIEGUE APLICACIONES WEB

# INDICE

---

# ¿Qué es NGINX?

---

Servidor web Open Source de alto rendimiento.

Permite numerosas conexiones simultáneas, lo que proporciona más velocidad y escalabilidad.

Entrega el contenido estático del sitio web rápidamente, es fácil de configurar y tiene un bajo consumo de recursos.

# Cómo funciona: diferencias con Apache

---

Apache funciona de forma individual, es decir, el usuario solicita una página a través del protocolo HTTP o HTTPS, que procesa y devuelve el resultado.

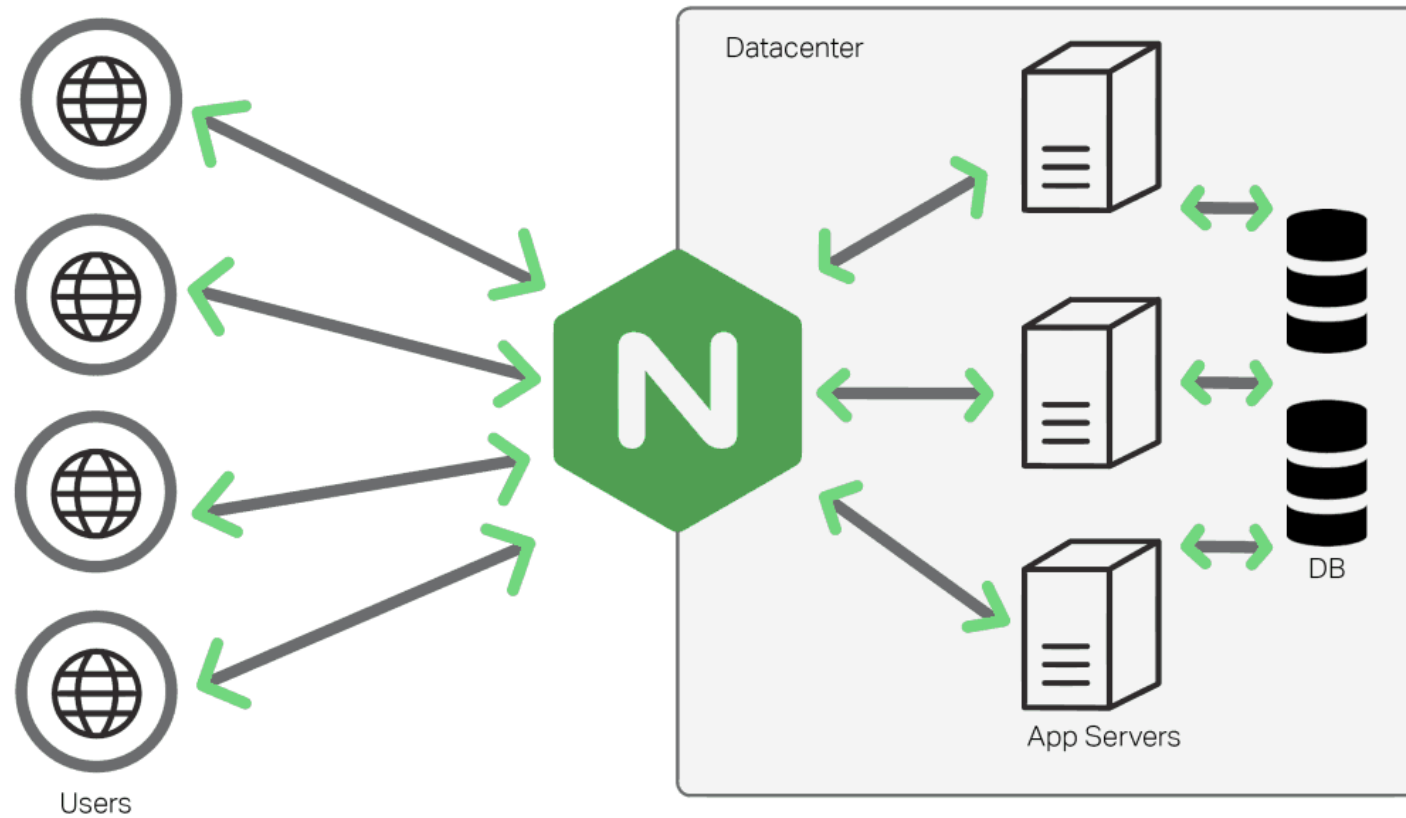
Este proceso es llamado de un thread individual, que se realiza para cada solicitud solicitada desde el servidor.

NGINX funciona con base en eventos. En lugar de hacer una solicitud directa al servidor, ejecuta un proceso maestro, llamado worker, y varios procesos de trabajo, llamados conexiones worker. Todo este proceso funciona de forma continua y asincrónica.

De esa manera, cuando hay una solicitud de procesamiento, se realizan las conexiones worker, que realizan la solicitud al proceso maestro, que a su vez procesa y devuelve el resultado.

Esta funcionalidad permite el manejo de numerosas conexiones simultáneas, ya que cada conexión worker es capaz de procesar 1024 solicitudes.

# Cómo funciona: diferencias con Apache



# Cómo funciona: diferencias con Apache

---

## Compatibilidad del sistema operativo

Los dos servidores funcionan en ambiente basados en UNIX, como LINUX. Con respecto a la plataforma Windows, NGINX tiene un rendimiento inferior en este ambiente.

## Configuraciones

La configuración de Apache se realiza de forma descentralizada, es decir, utiliza el archivo ".htaccess" extendido en los directorios de la aplicación y la carga de sus módulos se realiza en tiempo de ejecución.

En NGINX, la configuración se centraliza en el archivo "nginx.conf" y sus módulos se cargan dinámicamente.

## Rendimiento

La gran ventaja de NGINX es su capacidad de operar con miles de conexiones simultáneas, al doble de la velocidad requerida en Apache y, aun así, consumir menos memoria para contenido estático.

NGINX es un servidor web que ofrece numerosas funciones, como el equilibrio de carga, el proxy inverso y mucho más.

Su uso proporciona muchos beneficios a un sitio web, como mayor velocidad, escalabilidad y alta disponibilidad. Además, es fácil de configurar y compatible con las principales aplicaciones utilizadas en el mercado.

# Características

---

## Balanceo de carga

El balanceo de carga es un recurso extremadamente importante para aquellos que necesitan un sitio web con alta disponibilidad, ya que permite la distribución de solicitudes de servicio entre servidores:

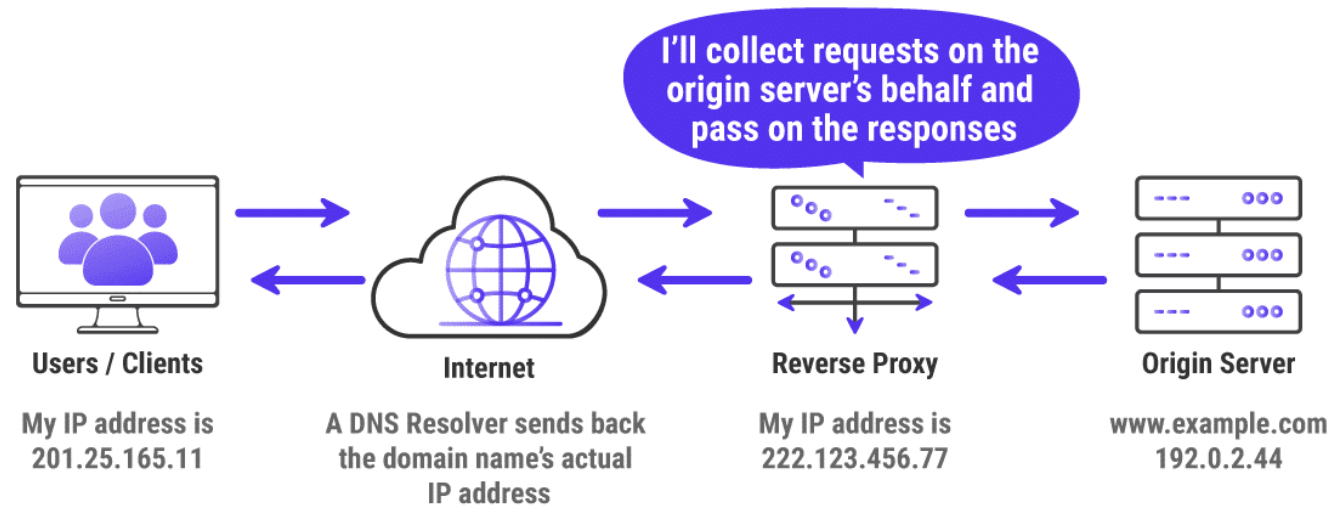
- por igual entre los servidores configurados,
- distribuirse al servidor que tiene pocas conexiones en este momento, o
- determinar la dirección IP de cada cliente para cada servidor específico.

# Características

## Proxy inverso

Un proxy, en la práctica, actúa como un servidor intermediario entre las computadoras en una red y el servidor web.

El proxy inverso es un servidor web que recibe solicitudes de conexión y administra lo que se requerirá en el servidor principal o verifica si la solicitud ya está disponible en caché.





# Características

---

## Streaming

NGINX ofrece un módulo nativo para streaming. Por lo tanto, permite una serie de configuraciones sobre cómo el servidor manejará el contenido MP4 y FLV, como el tamaño del buffer utilizado, el tiempo de timeout, etc.

# ¿Por qué Apache y NGINX juntos?

---

Apache es consume demasiada memoria del servidor.

Nginx necesita la ayuda de php-FPM o módulos similares para servir contenido dinámico.

Podemos combinar los dos servidores para conseguir el máximo rendimiento:

- Nginx como servidor estático en el front-end
- Apache como encargado del procesamiento dinámico en backend.

# Instalación - instalación de PHP-FPM

---

Instalaremos el paquete PHP-FPM. También instalaremos el módulo PHP FastCGI de Apache, libapache2-mod-fastcgi, para admitir aplicaciones web FastCGI.

```
sudo apt update
```

```
sudo apt install php-fpm
```

El paquete php-fpm está diseñado para manejar tráfico pesado y provee características avanzadas como la generación adaptativa de procesos, lo cual puede ser beneficioso para el rendimiento.

# Instalación - instalación de PHP-FPM

---

El módulo FastCGI Apache no está disponible en el repositorio de Ubuntu, por lo que debe descargarlo desde [kernel.org](https://kernel.org) e instalarlo usando el comando dpkg.

```
wget https://mirrors.edge.kernel.org/ubuntu/pool/multiverse/liba/libapache-mod-fastcgi/libapache2-mod-fastcgi_2.4.7~0910052141-1.2_amd64.deb
```

```
sudo dpkg -i libapache2-mod-fastcgi_2.4.7~0910052141-1.2_amd64.deb
```

# Instalación – conf. de Apache y PHP-FPM

---

Vamos a cambiar el número de puerto de Apache a 8080.

Renombramos al archivo de configuración ports.conf de Apache:

```
sudo mv /etc/apache2/ports.conf /etc/apache2/ports.conf.default
```

Creamos un nuevo archivo ports.conf con el nuevo puerto establecido en 8080

```
echo "Listen 8080" | sudo tee /etc/apache2/ports.conf
```

# Instalación – conf. de Apache y PHP-FPM

---

Crearemos un archivo de host virtual para Apache. La <VirtualHost> directiva en este archivo se establecerá para servir sitios únicamente en el puerto 8080.

Desactivamos el host virtual predeterminado

```
sudo a2dissite 000-default
```

creamos un nuevo archivo de host virtual usando el sitio predeterminado existente:

```
sudo cp /etc/apache2/sites-available/000-default.conf  
/etc/apache2/sites-available/001-default.conf
```

(es una sola línea)

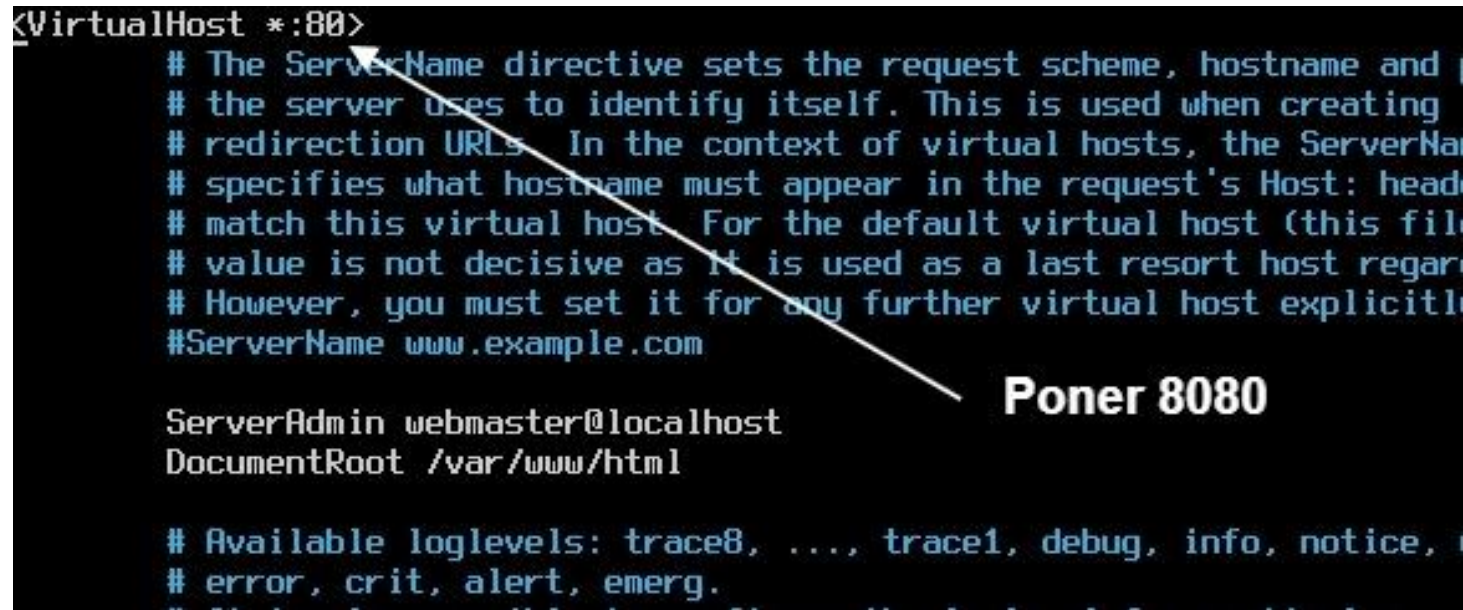
# Instalación – conf. de Apache y PHP-FPM

---

Ábrimos el nuevo fichero de configuración

```
sudo nano /etc/apache2/sites-available/001-default.conf
```

Y cambiamos el puerto de escucha a 8080



```
<VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and
# 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)
# value is not decisive as it is used as a last resort host regard
# 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,
# error, crit, alert, emerg.
```

**Poner 8080**

# Instalación – conf. de Apache y PHP-FPM

---

Activamos el nuevo archivo de configuración:

```
sudo a2ensite 001-default
```

vuelva a cargar Apache:

```
sudo systemctl reload apache2
```

Verificamos que, ahora, Apache escuche en 8080:

```
sudo netstat -tlnp
```

```
root@servidorlinux20:/home/alumno# systemctl reload apache2
root@servidorlinux20:/home/alumno# netstat -tlnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      1849/mysqld
tcp        0      0 127.0.0.53:53          0.0.0.0:*               LISTEN      608/systemd-resolv
tcp        0      0 127.0.0.1:33060        0.0.0.0:*               LISTEN      1849/mysqld
tcp6       0      0 :::8080                 :::*                    LISTEN      31720/apache2
root@servidorlinux20:/home/alumno#
```



# Instalación – conf. de Apache para usar mod\_fastcgi

---

Desactivamos primero mod\_php

```
sudo a2dismod php7.4
```

Habilitamos mod\_action

```
sudo a2enmod actions
```

Cambiamos el nombre del archivo de configuración FastCGI

```
sudo mv /etc/apache2/mods-enabled/fastcgi.conf /etc/apache2/mods-enabled/fastcgi.conf.default
```

Creamos un nuevo fichero de configuración

```
sudo nano /etc/apache2/mods-enabled/fastcgi.conf
```

# Instalación – conf. de Apache para usar mod\_fastcgi

---

```
<IfModule mod_fastcgi.c>
    AddHandler fastcgi-script .fcgi
    FastCgiIpcDir /var/lib/apache2/fastcgi
    AddType application/x-httpd-fastphp .php
    Action application/x-httpd-fastphp /php-fcgi
    Alias /php-fcgi /usr/lib/cgi-bin/php-fcgi

    FastCgiExternalServer /usr/lib/cgi-bin/php-fcgi -socket /run/php/php7.4-fpm.sock
    -pass-header Authorization

    <Directory /usr/lib/cgi-bin>
        Require all granted
    </Directory>
</IfModule>
```

# Instalación – conf. de Apache para usar mod\_fastcgi

---

Guardamos los cambios y probamos la configuración

```
sudo apachectl -t
```

Si todo va ok volvemos a cargar Apache

```
sudo systemctl reload apache2
```

Si vemos la advertencia *Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message.*, podemos ignorarlo por ahora. Configuraremos los nombres de servidor más adelante.

# Instalación – probar funcionalidad PHP

Creamos el archivo `/var/www/html/info.php` que contiene una llamada a la función `phpinfo`

```
echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php
```

Y probamos en el navegador del equipo cliente con

`http://ip del servidor:8080/info.php`

PHP Version 7.4.3-4ubuntu2.19



System	Linux servidorlinux20 5.4.0-163-generic #180-Ubuntu SMP Tue Sep 5 13:21:23 UTC 2023 x86_64
Build Date	Jun 27 2023 15:49:59
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/fpm
Loaded Configuration File	/etc/php/7.4/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/fpm/conf.d
Additional .ini files parsed	/etc/php/7.4/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.4/fpm/conf.d/10-opcache.ini, /etc/php/7.4/fpm/conf.d/10-pdo.ini, /etc/php/7.4/fpm/conf.d/15-xml.ini, /etc/php/7.4/fpm/conf.d/20-bz2.ini, /etc/php/7.4/fpm/conf.d/20-calendar.ini, /etc/php/7.4/fpm/conf.d/20-ctype.ini, /etc/php/7.4/fpm/conf.d/20-curl.ini, /etc/php/7.4/fpm/conf.d/20-dom.ini,

# Instalación – probar funcionalidad PHP

En la parte superior de la página, verifica que Server API diga FPM/FastCGI. A los dos tercios de la página, aproximadamente, la sección PHP Variables indicará que `**SERVER_SOFTWARE**` es Apache en Ubuntu. Esto confirma que `mod_fastcgi` está activo y que Apache está usando PHP-FPM para procesar archivos PHP.

PHP Variables

Variable	Value
<code>\$_SERVER['USER']</code>	www-data
<code>\$_SERVER['HOME']</code>	/var/www
<code>\$_SERVER['ORIG_SCRIPT_NAME']</code>	/php-fcgi
<code>\$_SERVER['ORIG_PATH_TRANSLATED']</code>	/var/www/html/info.php
<code>\$_SERVER['ORIG_PATH_INFO']</code>	/info.php
<code>\$_SERVER['ORIG_SCRIPT_FILENAME']</code>	/usr/lib/cgi-bin/php-fcgi
<code>\$_SERVER['SCRIPT_NAME']</code>	/info.php
<code>\$_SERVER['REQUEST_URI']</code>	/info.php
<code>\$_SERVER['QUERY_STRING']</code>	<i>no value</i>
<code>\$_SERVER['REQUEST_METHOD']</code>	GET
<code>\$_SERVER['SERVER_PROTOCOL']</code>	HTTP/1.1
<code>\$_SERVER['GATEWAY_INTERFACE']</code>	CGI/1.1
<code>\$_SERVER['REDIRECT_URL']</code>	/info.php
<code>\$_SERVER['REMOTE_PORT']</code>	53545
<code>\$_SERVER['SCRIPT_FILENAME']</code>	/var/www/html/info.php
<code>\$_SERVER['SERVER_ADMIN']</code>	webmaster@localhost
<code>\$_SERVER['CONTEXT_DOCUMENT_ROOT']</code>	/usr/lib/cgi-bin/php-fcgi
<code>\$_SERVER['CONTEXT_PREFIX']</code>	/php-fcgi
<code>\$_SERVER['REQUEST_SCHEME']</code>	http
<code>\$_SERVER['DOCUMENT_ROOT']</code>	/var/www/html
<code>\$_SERVER['REMOTE_ADDR']</code>	192.168.1.21
<code>\$_SERVER['SERVER_PORT']</code>	8080
<code>\$_SERVER['SERVER_ADDR']</code>	192.168.1.40
<code>\$_SERVER['SERVER_NAME']</code>	192.168.1.40
<code>\$_SERVER['SERVER_SOFTWARE']</code>	Apache/2.4.41 (Ubuntu)
<code>\$_SERVER['SERVER_SIGNATURE']</code>	<address>Apache/2.4.41 (Ubuntu) Server at 192.168.1.40 Port 8080</address>

# Instalación y configuración de Nginx

---

```
sudo apt-get install nginx
```

Eliminamos el enlace simbólico del host virtual predeterminado porque no lo vamos a usar más.

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

# Instalación y configuración de Nginx

---

Crearemos nuestro propio sitio predeterminado más adelante (example.com).

Ahora creamos hosts virtuales para Nginx

```
sudo mkdir -v /usr/share/nginx/example.com  
/usr/share/nginx/sample.org
```

Mantendremos los sitios web de Nginx en /usr/share/nginx, que es donde Nginx los quiere por defecto.

# Instalación y configuración de Nginx

---

Creamos los archivos index y phpinfo() para probar después de completar la configuración:

```
echo "<h1 style='color: green; ' >Example.com</h1>" | sudo tee  
/usr/share/nginx/example.com/index.html
```

```
echo "<h1 style='color: red; ' >Sample.org</h1>" | sudo tee  
/usr/share/nginx/sample.org/index.html
```

```
echo "<?php phpinfo(); ?>" | sudo tee  
/usr/share/nginx/example.com/info.php
```

```
echo "<?php phpinfo(); ?>" | sudo tee  
/usr/share/nginx/sample.org/info.php
```



# Instalación y configuración de Nginx

---

Creamos un archivo de host virtual para el dominio `example.com`

```
sudo nano /etc/nginx/sites-available/example.com
```

Nginx llama a áreas `server { . . . }` de **\*\*bloques de servidor\*\*** de un archivo de configuración. Cree un bloque de servidor para el host virtual primario, *example.com*. La directiva de configuración *default\_server* lo convierte en el host virtual predeterminado que procesa solicitudes HTTP que no coinciden con ningún otro host virtual

# Instalación y configuración de Nginx

---

```
server {  
    listen 80 default_server;  
  
    root /usr/share/nginx/example.com;  
    index index.php index.html index.htm;  
  
    server_name example.com www.example.com;  
    location / {  
        try_files $uri $uri/ /index.php;  
    }  
  
    location ~ \.php$ {  
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;  
        include snippets/fastcgi-php.conf;  
    }  
}
```

# Instalación y configuración de Nginx

---

Guarda y cierre el archivo. Ahora, cree un archivo de host virtual para el segundo dominio de Nginx, sample.org:

```
sudo nano etc/nginx/sites-available/sample.org
```

Añadimos lo siguiente al archivo:

```
server {
    root /usr/share/nginx/sample.org;
    index index.php index.html index.htm;

    server_name sample.org www.sample.org;
    location / {
        try_files $uri $uri/ /index.php;
    }

    location ~ \.php$ {
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;
        include snippets/fastcgi-php.conf;
    }
}
```

# Instalación y configuración de Nginx

---

Habilitamos ambos sitios creando enlaces simbólicos al directorio sites-enabled:

```
sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/example.com  
sudo ln -s /etc/nginx/sites-available/sample.org /etc/nginx/sites-enabled/sample.org
```

probamos la configuración de Nginx

```
sudo nginx -t
```

si no hay errores, volvemos a cargar Nginx:

```
sudo systemctl reload nginx
```