Sesión 5 — Introducción a los servidores web: Apache y Nginx

1. Introducción: ¿Qué es un servidor web?

Un **servidor web** es un software que **recibe, interpreta y responde** a las peticiones que envían los clientes (normalmente navegadores web).

Su principal función es **entregar recursos** (páginas HTML, imágenes, vídeos o resultados dinámicos) mediante los protocolos **HTTP** o **HTTPS**.

El término *servidor* se usa en dos sentidos:

1. **Hardware:** el ordenador físico que almacena los datos y ejecuta el servicio.
2. **Software:** el programa que procesa las solicitudes (como Apache o Nginx).

Conceptos básicos:

- **Servidor web:** Programa que recibe peticiones (por ejemplo, de un navegador) y responde enviando contenido (páginas, imágenes, vídeos, etc.).
- **Cliente**: Dispositivo o programa que solicita la información.
    - Ejemplo: un navegador.
- **HTTP / HTTPS:** Protocolo de comunicación entre cliente y servidor. HTTPS incluye seguridad con certificados TLS.
- **Puerto**: Puerta lógica usada para comunicarse (80 para HTTP y 443 para HTTPS).
- **Contenido estático:** Archivos que no cambian (HTML, CSS, imágenes).
- **Contenido dinámico:** Archivos que cambian según el usuario (PHP, formularios, bases de datos).
- **Proxy inverso**: Servidor que recibe las peticiones de los usuarios y las reparte entre varios servidores internos.
- **Balanceo de carga:** Técnica para repartir las peticiones entre varios servidores.
- **Logs**: Archivos donde el servidor guarda accesos y errores.
- **Host local (localhost):** El propio ordenador actuando como servidor (IP 127.0.0.1).

Ciclo básico de funcionamiento

1. **El cliente (navegador)** realiza una solicitud HTTP, por ejemplo:
   → GET /index.html HTTP/1.1

2. **El servidor web** recibe esa petición, busca el archivo solicitado y lo procesa.
3. Si el recurso existe, responde con:→ HTTP/1.1 200 OK + contenido.
4. El navegador interpreta el contenido y lo muestra al usuario.

Ejemplo práctico:

Cuando visitas https://www.wikipedia.org, tu navegador envía una petición al servidor de Wikipedia.

Este localiza el recurso solicitado (HTML, CSS, imágenes...) y lo devuelve como respuesta, que se renderiza en tu pantalla.

Componentes básicos de un servidor web

- **Motor HTTP:** interpreta peticiones y genera respuestas.
- **Módulos o extensiones:** añaden funciones extra (seguridad, compresión, estadísticas).
- **Sistema de archivos:** almacena los recursos servidos.
- **Configuración:** define puertos, dominios, permisos, logs, etc.
- **Registro de actividad (logs):** guardan todas las solicitudes y errores.

2. Importancia de los servidores web

Los servidores web son el **punto de encuentro entre usuarios y aplicaciones**. Sin ellos, no existirían los servicios que usamos cada día: tiendas online, redes sociales, correo, blogs o plataformas de streaming.

Los servidores web son responsables de:

- **Atender millones de usuarios simultáneamente.**
- **Gestionar seguridad** (certificados SSL/TLS).
- **Distribuir contenido estático y dinámico.**
- **Equilibrar la carga entre varios servidores.**
- **Optimizar la velocidad y el consumo de recursos.**

Ejemplo:

Cuando haces una compra en Amazon, tu navegador envía docenas de solicitudes simultáneas a los servidores de Amazon: para mostrar productos, cargar imágenes, verificar tu sesión, y procesar el pago.
Todo eso pasa por servidores web configurados para responder con eficiencia y seguridad.

3. Protocolos y funcionamiento interno

Los servidores web operan principalmente con dos protocolos:

- **HTTP (HyperText Transfer Protocol):** estándar de comunicación entre cliente y servidor.
- **HTTPS (HTTP Secure):** versión cifrada con SSL/TLS, garantiza confidencialidad e integridad.

Código de estado más comunes:

- 200 OK → Respuesta correcta.
- 301 Moved Permanently → Redirección permanente.
- 404 Not Found → Recurso inexistente.
- 500 Internal Server Error → Error en el servidor.

1. Los grandes protagonistas: Apache y Nginx

Hoy en día, los dos servidores más populares y relevantes del mundo son **Apache HTTP Server** y **Nginx**.

Servidor	Año	Creador / Fundación	Enfoque	Cuota global
Apache HTTP Server	1995	Apache Software Foundation	Flexibilidad, compatibilidad	~33%
Nginx	2004	Igor Sysoev (Rusia)	Rendimiento, alta concurrencia	~31%

Ambos son **software libre y multiplataforma**, funcionan en Linux, Windows y macOS, y pueden trabajar **juntos o por separado**.

4. Apache HTTP Server

Apache es un servidor web **de código abierto y gratuito**, lanzado en 1995.
Su nombre viene de la tribu nativa *Apache*, en honor a su "resistencia" y "modularidad".
Fue el primer servidor en ofrecer configuraciones flexibles, y durante dos décadas fue el estándar de la web.

Arquitectura de Apache

Apache utiliza un modelo **multiproceso o multihilo**, donde cada conexión genera un **nuevo proceso o hilo**.

Esto significa que si 500 usuarios acceden a la vez, Apache creará hasta 500 procesos simultáneos.

- **Ventaja:** cada conexión es independiente; si una falla, no afecta a las demás.
- **Desventaja:** alto consumo de memoria con mucho tráfico.

Ejemplo:

Un sitio educativo con 200 alumnos conectados simultáneamente puede funcionar bien, pero uno con 20.000 alumnos podría saturarse.

Estructura y módulos

Apache está compuesto por un **núcleo (core)** y un conjunto de **módulos** que amplían su funcionalidad.

Algunos de los más usados:

Módulo	Función
mod_rewrite	Redirecciones avanzadas y URLs amigables
mod_ssl	Conexiones seguras HTTPS
mod_proxy	Proxy directo e inverso
mod_php	Ejecución de código PHP
mod_cache	Sistema de caché de contenidos
mod_security	Filtro y cortafuegos para ataques comunes

Archivos .htaccess
Permiten aplicar reglas personalizadas por carpeta sin modificar la configuración principal.
Muy usados en **hosting compartido** o **sitios con múltiples administradores**.

Ejemplo práctico:
RewriteEngine On
RewriteRule ^noticias/(.*)$ noticias.php?id=$1 [L]
Con esta regla, la URL www.sitio.com/noticias/12 redirige internamente a noticias.php?id=12.

Ventajas de Apache
- Muy flexible y modular.
- Perfecta compatibilidad con PHP, Perl, Python.
- Soporte a .htaccess.
- Gran comunidad y documentación.
- Estabilidad probada durante más de 25 años.

Desventajas
- Menor rendimiento con muchas conexiones simultáneas.
- Mayor consumo de memoria.
- Configuraciones más pesadas en entornos de alto tráfico.

Ejemplo real:

La mayoría de las universidades con plataformas Moodle o WordPress usan Apache, porque facilita configuraciones personalizadas y seguridad granular por usuario.

5. Nginx

Nginx (Engine X) fue creado en 2004 por **Igor Sysoev**, con el objetivo de resolver el "problema C10k" (manejar más de 10.000 conexiones simultáneas).

Desde entonces, se ha convertido en el servidor más usado por grandes empresas por su **velocidad, estabilidad y eficiencia**.

Arquitectura de Nginx

A diferencia de Apache, Nginx utiliza un modelo **asíncrono basado en eventos**.
Esto significa que **no crea un proceso por conexión**:
Usa unos pocos procesos "workers" que manejan todas las solicitudes mediante una cola de eventos.

- **Ventaja:** puede gestionar miles de conexiones con muy poco consumo.
- **Desventaja:** no ejecuta PHP directamente (usa php-fpm).

Ejemplo:

- 1 proceso principal ("master") coordina todo.
- 4 procesos "workers" gestionan miles de peticiones simultáneas.

Características destacadas

Función	Descripción
Servidor HTTP	Procesa solicitudes web estándar.
Proxy inverso	Redirige tráfico a servidores backend (Apache, Node.js, etc.).
Balanceador de carga	Distribuye peticiones entre varios servidores.
Servidor de correo	Soporta IMAP/POP3/SMTP.
Caché de contenido	Guarda temporalmente respuestas para acelerar futuras peticiones.

Ventajas de Nginx

- Maneja miles de usuarios con bajo consumo.
- Ideal para contenido estático.
- Excelente proxy inverso y balanceador.
- Alta velocidad de respuesta.
- Configuración limpia y clara.

Desventajas

- No tiene .htaccess.
- Requiere php-fpm para contenido dinámico.
- Menos flexible en entornos compartidos.

Ejemplo real:

Netflix, **GitHub**, **WordPress.com**, **Spotify** y **Airbnb** utilizan Nginx por su capacidad para manejar tráfico global con alta eficiencia.

Ejemplo básico de configuración Nginx

```
server {
    listen 80;
    server_name ejemplo.com;

    root /var/www/html;
    index index.html;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

6. Comparativa completa entre Apache y Nginx

Aspecto	Apache	Nginx
Modelo de ejecución	Multiproceso/multihilo	Asíncrono basado en eventos
Consumo de recursos	Alto con muchas conexiones	Bajo y estable
.htaccess	Sí	No
Compatibilidad PHP	mod_php	php-fpm
Rendimiento en contenido estático	Medio	Muy alto
Rendimiento en contenido dinámico	Excelente	Depende del backend
Proxy inverso / balanceo	Limitado	Potente y nativo
Escalabilidad	Media	Muy alta
Facilidad de configuración	Alta	Requiere experiencia
Ideal para	Sitios pequeños o medianos	Sitios grandes, APIs, streaming

7. Casos prácticos y aplicaciones reales

1. Entorno educativo (Apache)
   - Moodle o WordPress para 300 alumnos.
   - .htaccess permite control por curso.
   - Integración con PHP sin complicaciones.
     - **Resultado:** estable, fácil de mantener, buen rendimiento local.

2. Diario digital (Nginx)
   - Miles de visitas simultáneas leyendo artículos.
   - Mucho contenido estático (imágenes, CSS).
     - **Resultado:** carga más rápida, menor uso de CPU.

3. Tienda online (Nginx + Apache)
   - Tráfico alto durante rebajas.
   - Nginx sirve contenido estático y actúa como proxy.
   - Apache procesa los pedidos y páginas PHP.
     - **Resultado:** 50% menos carga de CPU, 40% más velocidad.

4. API de microservicios (Nginx)
   - App móvil con miles de peticiones por segundo.
   - Nginx balancea y protege las conexiones.
     - **Resultado:** rendimiento y disponibilidad óptimos.

8. Cuándo usar Apache, Nginx o ambos

Usar solo Apache cuando:
- Proyecto pequeño o educativo.
- CMS tradicionales (WordPress, Moodle).
- Necesidad de .htaccess.
- Tráfico bajo o medio.

Ejemplo: Blog institucional o plataforma Moodle.

Usar solo Nginx cuando:
- Tráfico muy alto o global.
- Streaming, APIs o contenido estático.
- Escalabilidad en la nube.
- Integración con Node.js, Flask, Django...
-

Ejemplo: Netflix, Twitter, GitHub.

Usar los dos juntos cuando:

- Web con contenido dinámico y mucho tráfico.
- Se necesita equilibrio entre velocidad y compatibilidad.

Cliente → Nginx (proxy inverso)
↓
Apache (backend PHP)
↓
Base de datos

Ventajas combinadas:

- Nginx acelera y protege.
- Apache mantiene compatibilidad con CMS.
- Mejor rendimiento y seguridad.

Ejemplo real:
Wikipedia y **WordPress.com** usan esta arquitectura híbrida.

9. Resumen

Servidor	Puntos fuertes	Ideal para	Ejemplo
Apache	Modular, compatible, flexible	Aulas, CMS	Moodle, Joomla
Nginx	Rápido, eficiente, escalable	APIs, streaming	Netflix, GitHub
Combinación	Equilibrio perfecto	E-commerce, webs globales	Wikipedia, WordPress.com

Analogía:

- **Nginx** es el **portero veloz**: filtra, organiza y acelera la entrada.
- **Apache** es el **empleado especializado**: procesa las peticiones internas.
- **Juntos**, crean un sistema **seguro, eficiente y moderno** para cualquier tipo de aplicación web.