EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MEXICALI ®
La tecnología para el bien de la humanidad

Instituto tecnológico de Mexicali.

Ingeniería en sistemas computacionales.

Materia.
Desarrollo de aplicaciones web.

Investigación.
Archivo Nginx.conf

Docente.
Jose Ramon Bogarin Valenzuela.

Alumno.
Hernández López José Carlos-22150084.

¿Qué es Nginx?

Nginx es un software de servidor web de alto rendimiento. Su popularidad se debe a su arquitectura asíncrona y basada en eventos.

A diferencia de servidores más antiguos, Nginx no crea un nuevo hilo de trabajo por cada usuario. En su lugar, maneja miles de conexiones simultáneas en un solo proceso, lo que lo hace increíblemente rápido y eficiente con la memoria RAM.

En un despliegue de producción, Nginx actúa como el "recepcionista principal" de nuestro servidor, cumpliendo estas funciones clave:

- **Proxy Inverso:** Es su rol más común. Se sienta frente a nuestra aplicación (Node.js, Python, Java, etc.). Recibe las peticiones del usuario y las reenvía de forma segura a nuestra aplicación (que corre en localhost:3000, por ejemplo), ocultándola del Internet directo.
- **Servidor de Archivos Estáticos:** Es extremadamente rápido sirviendo archivos que no cambian (CSS, JavaScript, imágenes, HTML).
- **Terminación SSL/TLS:** Se encarga de todo el trabajo pesado del cifrado y descifrado HTTPS.
- **Balanceador de Carga:** Si nuestra aplicación crece, Nginx puede repartir el tráfico entre múltiples instancias de nuestra app para evitar sobrecargas.

Estructura de Configuración Profesional.

Nunca se debe editar el archivo principal /etc/nginx/nginx.conf.

Crear Configuración: Creamos nuestro archivo en /etc/nginx/sites-available/mi-app.conf.

Activar Configuración: Creamos un enlace simbólico (symlink) para activarlo:

sudo ln -s /etc/nginx/sites-available/mi-app.conf /etc/nginx/sites-enabled/

Ocultar Información Sensible.

Para evitar que los atacantes sepan qué versión de Nginx usamos y sus vulnerabilidades, editamos el archivo /etc/nginx/nginx.conf y nos aseguramos de que esta línea exista dentro del bloque http {}:

```
# /etc/nginx/nginx.conf
http {
    server_tokens off;
    # ...
}
```

SSL/TLS (HTTPS) como Prioridad Absoluta.

Un sitio en producción sin HTTPS no es una opción.

- Método Fácil: Usar Certbot (Let's Encrypt). Es un script que configura y renueva automáticamente los certificados SSL.

  sudo certbot –nginx

- Método Manual: Nos aseguramos de deshabilitar protocolos antiguos y peligrosos (SSLv3, TLS 1.0, 1.1).

  ```
  listen 443 ssl http2;
  listen [::]:443 ssl http2;

  ssl_certificate /ruta/a/fullchain.pem;
  ssl_certificate_key /ruta/a/privkey.pem;

  # Solo permite protocolos modernos y seguros
  ssl_protocols TLSv1.2 TLSv1.3;
  ssl_prefer_server_ciphers on;
  ```

Redirección Forzosa de HTTP a HTTPS.

Debemos crear un bloque server separado que capture todo el tráfico del puerto 80 (HTTP) y lo redirija permanentemente (301) a HTTPS.

```
server {
    listen 80;
    listen [::]:80;
    server_name mi-app.com www.mi-app.com;

    # Redirección 301 (permanente) a la versión segura
    return 301 https://$host$request_uri;
}
```

Cabeceras de Seguridad (Security Headers).

Estas se añaden dentro del bloque server (el del puerto 443) para instruir al navegador del usuario sobre cómo comportarse y mitigar ataques.

```
# 1. HSTS (Strict-Transport-Security): Fuerza al navegador a usar HTTPS por 6 meses.
add_header Strict-Transport-Security "max-age=15768000; includeSubDomains; preload" always;

# 2. X-Frame-Options: Evita que tu sitio sea incrustado en un <iframe> (previene Clickjacking).
add_header X-Frame-Options "SAMEORIGIN" always;

# 3. X-Content-Type-Options: Evita que el navegador "adivine" el tipo de contenido (MIME-sniffing).
add_header X-Content-Type-Options "nosniff" always;
```

```
# 4. Content-Security-Policy (CSP): (Opcional pero recomendado)
# Define una "lista blanca" de dónde se pueden cargar scripts, estilos, etc.
add_header Content-Security-Policy "default-src 'self'; script-src 'self'; object-src 'none';" always;
```

Plantilla para Servidor como Proxy Inverso (Para Backend: Node, Python, etc.).

Esta es la configuración ideal para una aplicación con backend. Nginx maneja SSL y los archivos estáticos, y pasa el resto a la aplicación.

```
# ========================================
# CONFIGURACIÓN DE PROXY INVERSO SEGURO
# (Para Node.js, Python, Java, Go, etc.)
# ========================================

# 1. Redirección de HTTP a HTTPS (Obligatoria)
server {
    listen 80;
    server_name tu-dominio.com;
    return 301 https://$host$request_uri;
}

# 2. Servidor Principal HTTPS
server {
    listen 443 ssl http2;
    server_name tu-dominio.com;

    # --- Configuración SSL ---
    # (Estas rutas son generalmente configuradas por Certbot)
    ssl_certificate /etc/letsencrypt/live/tu-dominio.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/tu-dominio.com/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;

    # --- Cabeceras de Seguridad (Hardening) ---
    add_header Strict-Transport-Security "max-age=15768000; includeSubDomains; preload" always;
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;

    # --- Configuración de Logs ---
    access_log /var/log/nginx/tu-dominio.access.log;
    error_log /var/log/nginx/tu-dominio.error.log;

    # --- Bloqueo de Archivos Sensibles ---
    location ~ /\. {
        deny all;
    }

    # --- Servir Archivos Estáticos (Opcional pero recomendado) ---
    # Si tus archivos estáticos están en /var/www/public
    location /static/ {
        root /var/www/public;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # --- Proxy Inverso a la Aplicación ---
    location / {
        # Pasa la solicitud a tu app (ej: Node.js en el puerto 3000)
        proxy_pass http://localhost:3000;

        # --- Cabeceras de Proxy Esenciales ---
        # Pasa la IP real del cliente a tu app
        proxy_set_header X-Real-IP $remote_addr;
        # Lista de IPs (útil si hay más proxies)
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        # Informa al backend que la conexión original fue HTTPS
        proxy_set_header X-Forwarded-Proto $scheme;
        # Pasa el Host original
        proxy_set_header Host $host;
    }
}
```

Plantilla 2: Servidor de SPA (Para Frontend: React, Vue, Angular).

Esta es la configuración ideal para servir una Single Page Application (SPA). No hay proxy, pero se añade la lógica de try_files y un bloqueo de archivos más estricto.

```
# ==========================================
# CONFIGURACIÓN DE SPA (React, Vue, Angular) SEGURA
# ==========================================

# 1. Redirección de HTTP a HTTPS (Obligatoria)
server {
    listen 80;
    server_name tu-spa.com;
    return 301 https://$host$request_uri;
}

# 2. Servidor Principal HTTPS
server {
    listen 443 ssl http2;
    server_name tu-spa.com;

    # --- Configuración SSL ---
    # (Estas rutas son generalmente configuradas por Certbot)
    ssl_certificate /etc/letsencrypt/live/tu-spa.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/tu-spa.com/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_prefer_server_ciphers on;

    # --- Cabeceras de Seguridad (Hardening) ---
    add_header Strict-Transport-Security "max-age=15768000; includeSubDomains; preload" always;
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Content-Security-Policy "default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self'; object-src 'none';" always;

    # --- Raíz y Logs ---
    # Directorio donde está el 'build' de tu SPA
    root /var/www/mi-spa/build;
    index index.html;

    access_log /var/log/nginx/tu-spa.access.log;
    error_log /var/log/nginx/tu-spa.error.log;

    # --- Compresión (Rendimiento) ---
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_types text/plain text/css text/xml text/javascript application/x-javascript application/javascript application/json;

    # --- Bloqueo Granular de Archivos Sensibles ---

    # Bloquear acceso a archivos de entorno
    location ~ /\.env { deny all; }

    # Bloquear acceso a archivos git
    location ~ /\.git { deny all; }

    # Bloquear Source Maps (¡Muy importante!)
    location ~ \.map$ { deny all; }

    # Bloquear directorios de desarrollo (si se subieron por error)
    location ~ ^/(node_modules|src) { deny all; }

    # Bloquear archivos de código fuente (si se subieron por error)
    location ~ \.(ts|tsx|jsx|vue)$ { deny all; }

    # Bloquear archivos de configuración del proyecto
    location ~ /(README|package\.json|package-lock\.json|yarn\.lock|vite\.config\.js) {
        deny all;
    }

    # --- Caché para Archivos Estáticos (Rendimiento) ---
    # Los archivos con hash (ej. main.a4b8c1.js) pueden cachearse indefinidamente
    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg|woff|woff2)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # --- ¡LA MAGIA DE LA SPA! ---
    # Esta es la directiva clave para React/Vue Router.
    # 1. Intenta servir el archivo ($uri)
    # 2. Intenta servir un directorio ($uri/)
    # 3. Si falla, sirve /index.html y deja que el router del cliente decida.
    location / {
        try_files $uri $uri/ /index.html;
    }
}
```