MÓDULO 1(anexos):

LAMP

(y LEMP)

HTTP (HyperText Transfer Protocol)

Es un protocolo de comunicación utilizado para transferir datos en la web. Funciona sobre el puerto 80 y no cifra la información, lo que significa que los datos viajan en texto plano. Esto lo hace vulnerable a interceptaciones y ataques.

HTTPS (HyperText Transfer Protocol Secure)

Es una versión segura de HTTP que utiliza cifrado para proteger la comunicación entre el cliente (navegador) y el servidor. Funciona sobre el puerto 443 y requiere un certificado SSL/TLS para establecer una conexión segura.

Certificados SSL/TLS

Un certificado SSL/TLS es un archivo digital que vincula una clave pública con la identidad de una entidad (como un sitio web). Sirve para:

- Autenticar la identidad del servidor.
- Cifrar la comunicación entre el cliente y el servidor.

Los certificados son emitidos por Autoridades de Certificación (CA) confiables, como Let's Encrypt, o pueden ser autofirmados (generados localmente).

Certificados autofirmados

Son certificados generados por el propio servidor sin la intervención de una CA. Son útiles para pruebas internas o entornos de desarrollo, pero no son confiables para uso público porque los navegadores mostrarán advertencias de seguridad.

Certbot y Let's Encrypt

- **Let's Encrypt** es una CA gratuita y automatizada que proporciona certificados SSL/TLS válidos para sitios web.
- **Certbot** es una herramienta que facilita la obtención y renovación de certificados de Let's Encrypt.

Cuándo usar certificados autofirmados o Let's Encrypt

- **Autofirmados** : Ideal para entornos de desarrollo, intranets o pruebas internas.
- **Let's Encrypt** : Para sitios públicos donde la confianza del usuario es crucial.

Generación de certificados autofirmados (X.509) usando OpenSSL

1. **Instalar OpenSSL** (si no está instalado):

   sudo apt install openssl # En Ubuntu
   sudo dnf install openssl # En Rocky

2. **Generar una clave privada** :

   openssl genpkey -algorithm RSA -out server.key

3. **Generar una solicitud de firma de certificado (CSR)** :

   openssl req -new -key server.key -out server.csr

   Durante este proceso, se pedirá información como el nombre común (CN), que debe coincidir con el dominio del servidor.

4. **Generar el certificado autofirmado** :

   openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt

Esto generará un certificado válido por 365 días.

Instalación de un sistema LAMP en Ubuntu 24.04 y Rocky 9.4

LAMP (Linux, Apache, MySQL/MariaDB, PHP)
Es una pila de software utilizada para alojar sitios web dinámicos.

Paso a paso en Ubuntu 24.04

1. **Actualizar el sistema** :

   sudo apt update && sudo apt upgrade -y

2. **Instalar Apache** :

sudo apt install apache2 -y
systemctl start apache2
systemctl enable apache2

3. **Instalar MariaDB** :

sudo apt install mariadb-server mariadb-client -y
systemctl start mariadb
systemctl enable mariadb

4. **Configurar MariaDB** :

sudo mysql_secure_installation

5. **Instalar PHP** :

sudo apt install php libapache2-mod-php php-mysql -y
systemctl restart apache2

6. **Configurar el firewall** :

sudo ufw allow 'Apache Full'
sudo ufw enable

7. **Activar el módulo SSL** :

sudo a2enmod ssl
sudo systemctl restart apache2

8. **Configurar HTTPS** :

- Copiar los certificados (`server.key` y `server.crt`) al directorio `/etc/ssl/`.
- Editar el archivo de configuración de Apache (`/etc/apache2/sites-available/default-ssl.conf`):

<VirtualHost *:443>
ServerAdmin webmaster@localhost
DocumentRoot /var/www/html

SSLEngine on

```
SSLCertificateFile /etc/ssl/server.crt
SSLCertificateKeyFile /etc/ssl/server.key
</VirtualHost>
```

- Habilitar el sitio SSL:

```
sudo a2ensite default-ssl.conf
sudo systemctl reload apache2
```

Paso a paso en Rocky 9.4

1. **Actualizar el sistema** :

```
sudo dnf update -y
```

2. **Instalar Apache** :

```
sudo dnf install httpd -y
systemctl start httpd
systemctl enable httpd
```

3. **Instalar MariaDB** :

```
sudo dnf install mariadb-server mariadb -y
systemctl start mariadb
systemctl enable mariadb
```

4. **Configurar MariaDB** :

```
sudo mysql_secure_installation
```

5. **Instalar PHP** :

```
sudo dnf install php php-mysqlnd -y
systemctl restart httpd
```

6. **Configurar el firewall** :

sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload

7. **Activar el módulo SSL** :
    - Editar el archivo `/etc/httpd/conf.d/ssl.conf`:

      <VirtualHost *:443>
      DocumentRoot "/var/www/html"
      SSLEngine on
      SSLCertificateFile /etc/pki/tls/certs/server.crt
      SSLCertificateKeyFile /etc/pki/tls/private/server.key
      </VirtualHost>

    - Reiniciar Apache:
      bash
      1
      systemctl restart httpd

¿Qué es NGINX?

NGINX es un servidor web de alto rendimiento que también puede actuar como proxy inverso, balanceador de carga y servidor de correo. Es conocido por su eficiencia en el manejo de conexiones simultáneas.

NGINX es un servidor web de alto rendimiento que también puede actuar como:

- **Servidor HTTP** : Sirve contenido estático y dinámico.
- **Proxy inverso** : Distribuye solicitudes a otros servidores backend.
- **Balanceador de carga** : Distribuye el tráfico entre múltiples servidores para mejorar la disponibilidad y el rendimiento.
- **Servidor de correo** : Proporciona servicios de proxy para protocolos de correo como IMAP, POP3 y SMTP.

Diferencias clave entre NGINX y Apache

Característica	NGINX	Apache
Arquitectura	Basado en eventos asíncronos (event-driven).	Basado en procesos por solicitud (process/thread-based).
Rendimiento	Excelente para servir contenido estático y manejar muchas conexiones.	Mejor para contenido dinámico con configuraciones flexibles.

Característica	NGINX	Apache
Configuración	Archivos de configuración centralizados (`/etc/nginx/nginx.conf` y más).	Configuración distribuida en archivos `.conf` dentro de `/etc/apache2/`.
Sintaxis	Más compacta y directa.	Más detallada y flexible con `.htaccess`.
Escalabilidad	Ideal para aplicaciones de alta concurrencia.	Adecuado para servidores pequeños o medianos.

Escenarios de uso de NGINX frente a Apache

1. **NGINX** :
    - Sitios web con alto tráfico y muchas conexiones simultáneas.
    - Servidores que necesitan balancear carga o actuar como proxy inverso.
    - Aplicaciones que requieren servir contenido estático rápidamente.
    - APIs RESTful o microservicios.
2. **Apache** :
    - Sitios web con configuraciones complejas y personalizaciones avanzadas.
    - Entornos donde se necesita soporte para `.htaccess`.
    - Servidores con menos tráfico pero más configuraciones específicas.

Paso a paso: Instalación de LEMP (Linux, NGINX, MariaDB, PHP) en Ubuntu 24.04

1. Actualizar el sistema

```
sudo apt update && sudo apt upgrade -y
```

2. Instalar NGINX

```
sudo apt install nginx -y
systemctl start nginx
systemctl enable nginx
```

3. Instalar MariaDB

```
sudo apt install mariadb-server mariadb-client -y
systemctl start mariadb
```

systemctl enable mariadb

4. Configurar MariaDB

sudo mysql_secure_installation

5. Instalar PHP y módulos necesarios

sudo apt install php-fpm php-mysql -y

6. Configurar PHP-FPM

- Editar el archivo `/etc/php/8.x/fpm/pool.d/www.conf` (ajustar según la versión de PHP):

  listen = /run/php/php8.x-fpm.sock

- Reiniciar PHP-FPM:

  systemctl restart php8.x-fpm

7. Configurar NGINX para usar PHP

- Editar el archivo de configuración predeterminado de NGINX (`/etc/nginx/sites-available/default`):

  server {
  listen 80;
  server_name _;

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

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

  location ~ \.php$ {
  include snippets/fastcgi-php.conf;

```
fastcgi_pass unix:/run/php/php8.x-fpm.sock;
}

location ~ /\.ht {
deny all;
}
}
```

- Probar la configuración de NGINX:

  sudo nginx -t

- Reiniciar NGINX:

  systemctl restart nginx

8. Configurar el firewall

sudo ufw allow 'Nginx Full'
sudo ufw enable

Paso a paso: Instalación de LEMP en Rocky 9.4

1. Actualizar el sistema

sudo dnf update -y

2. Instalar NGINX

sudo dnf install nginx -y
systemctl start nginx
systemctl enable nginx

3. Instalar MariaDB

sudo dnf install mariadb-server mariadb -y
systemctl start mariadb
systemctl enable mariadb

4. Configurar MariaDB

sudo mysql_secure_installation

5. Instalar PHP y módulos necesarios

sudo dnf install php php-fpm php-mysqlnd -y

6. Configurar PHP-FPM

- Editar el archivo `/etc/php-fpm.d/www.conf`:

  listen = /run/php-fpm/www.sock

- Reiniciar PHP-FPM:

  systemctl restart php-fpm

7. Configurar NGINX para usar PHP

- Editar el archivo de configuración predeterminado de NGINX (`/etc/nginx/nginx.conf` o `/etc/nginx/conf.d/default.conf`):

```
server {
listen 80;
server_name _;

root /usr/share/nginx/html;
index index.php index.html index.htm;

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

location ~ \.php$ {
include fastcgi_params;
fastcgi_pass unix:/run/php-fpm/www.sock;
fastcgi_index index.php;
fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
```

```
}

location ~ /\.ht {
deny all;
}
}
```

- Probar la configuración de NGINX:

  sudo nginx -t

- Reiniciar NGINX:

  systemctl restart nginx

8. Configurar el firewall

```
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload
```

Ficheros de configuración y sintaxis de NGINX

Estructura básica de un archivo de configuración de NGINX

```
server {
listen 80; # Puerto en el que escucha
server_name example.com; # Nombre del servidor

root /var/www/html; # Directorio raíz
index index.html index.php; # Archivos de índice

location / { # Bloque de ubicación
try_files $uri $uri/ =404; # Manejo de errores
}

location ~ \.php$ { # Manejo de archivos PHP
include fastcgi_params;
fastcgi_pass unix:/run/php/php8.x-fpm.sock;
```

}

location ~ /\.ht { # Denegar acceso a archivos .htaccess
deny all;
}
}

Diferencias clave con Apache

- **Directivas** : NGINX usa `location`, `server`, y `upstream`, mientras que Apache usa `<Directory>`, `<VirtualHost>`, y `.htaccess`.
- **Módulos** : NGINX no tiene `.htaccess`; todas las configuraciones deben estar en los archivos principales.
- **Rendimiento** : NGINX es más eficiente en entornos de alta concurrencia.