TEMARIO: ARQUITECTURA EN LA NUBE - CONFIGURACIÓN AVANZADA DE SERVIDORES WEB Y HTTPS

CONCEPTOS BÁSICOS PARA LA CLASE

Las Herramientas que Vamos a Usar

- **Apache (*httpd*):** Es el servidor web más veterano y popular del mundo. Como un chef experimentado, puede hacer de todo y es muy configurable. Lleva décadas funcionando en millones de sitios web.
- **Nginx:** Es el servidor web moderno y eficiente. Como un camarero rápido, puede atender muchas peticiones a la vez sin cansarse. Es perfecto para sitios con mucho tráfico.
- **Caddy:** Es el servidor web más nuevo y fácil de usar. Su superpoder es que configura HTTPS automáticamente, algo que con otros servidores requiere esfuerzo manual. Es como tener un mayordomo que lo hace todo por ti.
- **Certbot:** Es una herramienta que obtiene certificados SSL/TLS gratuitos de Let's Encrypt. Estos certificados son como el candado que ves en tu navegador cuando una página es segura (*HTTPS*).

¿Qué es HTTPS y por qué es importante?

- **HTTP** es como enviar postales: cualquiera puede leer lo que pones.
- **HTTPS** es como enviar cartas en sobres cerrados con sello de lacre: nadie puede leer el contenido ni falsificarlo.

Cuando ves el candado en tu navegador, significa que la comunicación está cifrada y es segura.

¿Por qué hacer esta práctica?

En el mundo real, las empresas no usan un solo servidor web. A veces tienen varios funcionando al mismo tiempo para diferentes propósitos:

- Uno para la página principal
- Otro para aplicaciones internas
- Otro como proxy para distribuir la carga

Además, **TODOS los sitios profesionales usan HTTPS.** Aprender a configurarlo es esencial para cualquier profesional de IT.

Al final de esta práctica comprenderás cómo funcionan los servidores web profesionales, cómo pueden coexistir en la misma máquina y cómo proteger las comunicaciones con **HTTPS.**

COMANDOS Y PROCEDIMIENTOS

PARTE 1: INSTALACIÓN Y CONFIGURACIÓN DE APACHE

1. Actualizar el sistema

Comando:

sudo apt update && sudo apt upgrade -y

Descripción: Actualiza la lista de paquetes y mejora el sistema a las últimas versiones.

---

2. Instalar Apache2

Comando:

sudo apt install apache2 -y

Descripción: Instala el servidor web Apache en tu sistema.

---

3. Configurar Apache en puerto 8080

Comando:

sudo nano /etc/apache2/ports.conf

Descripción: Abre el archivo de configuración de puertos. Cambia Listen 80 por Listen 8080.

---

4. Modificar el VirtualHost

Comando:

sudo nano /etc/apache2/sites-available/000-default.conf

Descripción: Cambia <VirtualHost *:80> por <VirtualHost *:8080>.

---

5. Instalar PHP

Comando:

sudo apt install php libapache2-mod-php -y

Descripción: Instala PHP y su módulo para funcionar con Apache.

---

6. Reiniciar Apache

Comando:

sudo systemctl restart apache2

Descripción: Reinicia Apache para aplicar los cambios.

---

7. Verificar estado de Apache

Comando:

sudo systemctl status apache2

Descripción: Comprueba que Apache está funcionando correctamente.

Comando adicional para verificar puerto:

sudo netstat -tulpn | grep apache2

Descripción: Verifica en qué puerto está escuchando Apache (*debe mostrar 8080*).

---

8. Crear archivo PHP de prueba

Comando:

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

Descripción: Crea un archivo que muestra información del PHP instalado.

---

9. Probar Apache desde terminal

Comando:

curl http://localhost:8080/info.php

Descripción: Verifica que Apache sirve correctamente el contenido PHP.

PARTE 2: INSTALACIÓN Y CONFIGURACIÓN DE NGINX

1. Instalar Nginx

Comando:

sudo apt install nginx -y

Descripción: Instala el servidor web Nginx en tu sistema.

---

2. Configurar Nginx en puerto 8081

Comando:

```
sudo nano /etc/nginx/sites-available/default
```

Descripción: Abre la configuración por defecto. Cambia listen 80 por listen 8081.

---

3. Crear página HTML personalizada

Comando:

```
echo "<h1>Servidor Nginx</h1><p>Funcionando en puerto 8081</p>" | sudo tee /usr/share/nginx/html/index.html
```

Descripción: Crea una página HTML identificable para Nginx.

---

4. Reiniciar Nginx

Comando:

```
sudo systemctl restart nginx
```

Descripción: Reinicia Nginx para aplicar los cambios de configuración.

---

5. Verificar estado de Nginx

Comando:

```
sudo systemctl status nginx
```

Descripción: Comprueba que Nginx está funcionando correctamente.

Comando adicional para verificar puerto:

```
sudo netstat -tulpn | grep nginx
```

Descripción: Verifica en qué puerto está escuchando Nginx (*debe mostrar 8081*).

---

6. Probar Nginx desde terminal

Comando:

```
curl http://localhost:8081
```

Descripción: Verifica que Nginx sirve correctamente el contenido HTML.

PARTE 3: INSTALACIÓN Y CONFIGURACIÓN DE CADDY

1. Instalar dependencias necesarias

Comando:

```
sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl
```

Descripción: Instala herramientas necesarias para añadir repositorios externos.

---

2. Agregar repositorio de Caddy

Comando:

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
```

```
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list
```

Descripción: Añade el repositorio oficial de Caddy a tu sistema.

---

3. Actualizar e instalar Caddy

Comando:

```
sudo apt update && sudo apt install caddy -y
```

Descripción: Actualiza la lista de paquetes e instala Caddy.

---

4. Crear directorio para Caddy

Comando:

```
sudo mkdir -p /var/www/caddy
```

Descripción: Crea un directorio específico para los archivos de Caddy.

---

5. Crear archivo Markdown de prueba

Comando:

```
echo "# Bienvenido a Caddy" | sudo tee /var/www/caddy/README.md
```

```
echo "" | sudo tee -a /var/www/caddy/README.md
```

```
echo "Este servidor está funcionando correctamente." | sudo tee -a /var/www/caddy/README.md
```

```
echo "" | sudo tee -a /var/www/caddy/README.md
```

```
echo "## Características" | sudo tee -a /var/www/caddy/README.md
echo "- Servidor moderno" | sudo tee -a /var/www/caddy/README.md
echo "- HTTPS automático" | sudo tee -a /var/www/caddy/README.md
echo "- Fácil configuración" | sudo tee -a /var/www/caddy/README.md
```

Descripción: Crea un archivo Markdown con contenido de ejemplo.

---

6. Crear imagen de prueba (cuidado WSL hay que hacer ajustes)

Comando:

```
curl -o /tmp/test-image.jpg "https://www.python.org/static/apple-touch-icon-144x144-precomposed.png"
sudo mv /tmp/test-image.jpg /var/www/caddy/test.jpg
```

Descripción: Descarga una imagen de prueba para verificar que Caddy sirve archivos estáticos.

---

7. Crear Caddyfile personalizado

Comando:

```
sudo nano /etc/caddy/Caddyfile
```

Descripción: Abre el archivo de configuración de Caddy.

Escribe el siguiente contenido:

```
:8082 {
    root * /var/www/caddy
    file_server browse

    @markdown path *.md
    header @markdown Content-Type text/plain
}
```

---

8. Reiniciar Caddy

Comando:

sudo systemctl restart caddy

Descripción: Reinicia Caddy para aplicar la nueva configuración.

---

9. Verificar estado de Caddy

Comando:

sudo systemctl status caddy

Descripción: Comprueba que Caddy está funcionando correctamente.

Comando adicional para verificar puerto:

sudo netstat -tulpn | grep caddy

Descripción: Verifica en qué puerto está escuchando Caddy (*debe mostrar 8082*).

---

10. Probar Caddy desde terminal

Comando:

curl http://localhost:8082/

Descripción: Lista los archivos disponibles en el servidor Caddy.

---

11. Probar archivo Markdown

Comando:

curl http://localhost:8082/README.md

Descripción: Verifica que Caddy sirve correctamente archivos Markdown.

PARTE 4: CONFIGURACIÓN DE HTTPS CON CERTBOT EN APACHE

1. Instalar Certbot y el plugin de Apache

Comando:

sudo apt install certbot python3-certbot-apache -y

Descripción: Instala Certbot y su integración con Apache para gestionar certificados SSL.

---

2. Verificar dominio o usar localhost

Nota: Para obtener certificados reales de Let's Encrypt necesitas un dominio público. Para esta práctica usaremos certificados autofirmados.

Comando:

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt

Descripción: Crea un certificado autofirmado para practicar HTTPS localmente. Completa los campos solicitados (*puedes usar valores por defecto o responder: ES, Madrid, Madrid, vacío, vacío, localhost y ejemplo@gmail.com*).

Nota sobre snakeoil: Los certificados "snakeoil" NO se crean automáticamente al generar certificados self-signed. Son certificados de ejemplo que vienen preinstalados con algunos paquetes SSL (*como ssl-cert en Debian/Ubuntu*) y se encuentran en /etc/ssl/certs/ssl-cert-snakeoil.pem y /etc/ssl/private/ssl-cert-snakeoil.key. Son certificados genéricos que se pueden usar para pruebas, pero cuando ejecutas el comando openssl req estás creando tus propios certificados self-signed personalizados, completamente independientes de los snakeoil.

---

3. Habilitar módulo SSL en Apache

Comando:

sudo a2enmod ssl

Descripción: Activa el módulo SSL necesario para HTTPS en Apache.

---

4. Cambiar puerto SSL

Comando:

sudo nano /etc/apache2/ports.conf

Descripción: Añade la línea Listen 8443 para que Apache escuche HTTPS en puerto 8443.

---

5. Modificar VirtualHost SSL

Comando:

sudo nano /etc/apache2/sites-available/default-ssl.conf

Descripción: Cambia <VirtualHost *:443> por <VirtualHost *:8443>.

---

6. Habilitar sitio SSL

Comando:

sudo a2ensite default-ssl.conf

Descripción: Activa la configuración SSL en Apache.

---

7. Reiniciar Apache

Comando:

sudo systemctl restart apache2

Descripción: Aplica todos los cambios de configuración SSL.

---

8. Verificar HTTPS

Comando:

curl -i -k https://localhost:8443

Descripción: Prueba la conexión HTTPS (*el flag -k ignora el aviso del certificado autofirmado*).

PARTE 5: VERIFICACIÓN FINAL DE LOS TRES SERVIDORES

1. Verificar que todos los servicios están activos

Comando para verificar Apache:

sudo systemctl status apache2

Comando para verificar puerto de Apache:

sudo netstat -tulpn | grep apache2

Comando para verificar Nginx:

sudo systemctl status nginx

Comando para verificar puerto de Nginx:

sudo netstat -tulpn | grep nginx

Comando para verificar Caddy:

sudo systemctl status caddy

Comando para verificar puerto de Caddy:

sudo netstat -tulpn | grep caddy

Descripción: Estos comandos muestran el estado de cada servidor individualmente y verifican en qué puertos están escuchando.

---

2. Verificar puertos en uso (*todos simultáneamente*)

Comando:

sudo netstat -tulpn | grep -E '8080|8081|8082|8443'

Descripción: Lista los cuatro puertos donde están escuchando los servidores simultáneamente.

---

3. Probar todos los servidores

Comandos:

curl http://localhost:8080

curl http://localhost:8081

curl http://localhost:8082

curl -i -k https://localhost:8443

Descripción: Verifica que cada servidor responde correctamente en su puerto asignado.

PROPUESTA DE EJERCICIO PRÁCTICO

Laboratorio: Configuración Simultánea de Múltiples Servidores Web y HTTPS

Objetivos:

- Configurar Apache, Nginx y Caddy funcionando simultáneamente en puertos diferentes.
- Verificar el funcionamiento de cada servidor con contenido específico.
- Implementar HTTPS con certificados autofirmados en Apache.
- Comprender las diferencias entre servidores web.
- Documentar todo el proceso mediante capturas de pantalla.

Parte 1: Servidor Apache con PHP (*Puerto 8080 y HTTPS en 8443*)

Tareas a realizar:

1. Actualizar el sistema y capturar el resultado.
2. Instalar Apache2 y mostrar confirmación.
3. Configurar Apache en puerto 8080 y capturar archivo ports.conf.
4. Instalar PHP y capturar confirmación.
5. Crear archivo info.php.
6. Acceder a http://localhost:8080/info.php y capturar pantalla.
7. Configurar HTTPS con certificado autofirmado.
8. Acceder a https://localhost:8443 y capturar pantalla con el candado.

Entregables Parte 1:

- Captura de sudo systemctl status apache2 mostrando servicio activo.
- Captura de sudo netstat -tulpn | grep apache2 mostrando puertos 8080 y 8443.
- Captura de la página info.php en puerto 8080.
- Captura del archivo ports.conf mostrando puertos 8080 y 8443.
- Captura de la página HTTPS en puerto 8443 con el candado del navegador.

Parte 2: Servidor Nginx con HTML (*Puerto 8081*)

Tareas a realizar:

1. Instalar Nginx y capturar confirmación.
2. Configurar Nginx en puerto 8081 y capturar archivo de configuración.
3. Crear página HTML personalizada.
4. Reiniciar Nginx y capturar estado del servicio.
5. Acceder a http://localhost:8081 y capturar pantalla.

Entregables Parte 2:

- Captura de sudo systemctl status nginx mostrando servicio activo.
- Captura de sudo netstat -tulpn | grep nginx mostrando puerto 8081.
- Captura del archivo de configuración mostrando puerto 8081.
- Captura de la página HTML en navegador.

Parte 3: Servidor Caddy con Archivos Especiales (*Puerto 8082*)

Tareas a realizar:

1. Instalar Caddy siguiendo el procedimiento completo.
2. Crear directorio /var/www/caddy y capturar confirmación.
3. Crear archivo Markdown (*README.md*) con contenido personalizado.
4. Descargar o crear una imagen de prueba.
5. Configurar Caddyfile para puerto 8082 con file_server browse.
6. Capturar el contenido del Caddyfile.
7. Reiniciar Caddy y verificar estado.
8. Acceder a http://localhost:8082 y capturar el listado de archivos.
9. Acceder a http://localhost:8082/README.md y capturar el contenido.
10. Acceder a http://localhost:8082/test.jpg y capturar la imagen.

Entregables Parte 3:

- Captura de sudo systemctl status caddy mostrando servicio activo.
- Captura de sudo netstat -tulpn | grep caddy mostrando puerto 8082.
- Captura del Caddyfile configurado.
- Captura del navegador de archivos de Caddy.
- Captura del archivo Markdown visualizado.
- Captura de la imagen servida por Caddy.

Parte 4: Verificación Simultánea de los Tres Servidores

Tareas finales:

1. Ejecutar comandos que muestren estado de cada servicio individualmente.
2. Ejecutar comando netstat mostrando los cuatro puertos en uso (*8080, 8081, 8082, 8443*).
3. Abrir cuatro pestañas del navegador simultáneamente:
    - Pestaña 1: Apache HTTP (*8080*)
    - Pestaña 2: Apache HTTPS (*8443*)
    - Pestaña 3: Nginx (*8081*)
    - Pestaña 4: Caddy (*8082*)
4. Capturar pantalla completa mostrando las cuatro pestañas funcionando.

Entregables Parte 4:

- Captura individual del estado de Apache: sudo systemctl status apache2
- Captura individual del estado de Nginx: sudo systemctl status nginx
- Captura individual del estado de Caddy: sudo systemctl status caddy
- Captura individual de puertos de Apache: sudo netstat -tulpn | grep apache2
- Captura individual de puertos de Nginx: sudo netstat -tulpn | grep nginx
- Captura individual de puertos de Caddy: sudo netstat -tulpn | grep caddy
- Captura del comando netstat mostrando los cuatro puertos: sudo netstat -tulpn | grep -E '8080|8081|8082|8443'
- Captura del navegador con las cuatro pestañas abiertas y funcionando.

CRITERIOS DE EVALUACIÓN

1. Funcionamiento Correcto (*70%*)

- Apache sirve correctamente PHP en puerto 8080 (*15%*).
- Apache sirve correctamente HTTPS en puerto 8443 con certificado (*20%*).
- Nginx sirve correctamente HTML en puerto 8081 (*15%*).
- Caddy sirve correctamente archivos Markdown e imágenes en puerto 8082 (*20%*).

2. Documentación (*30%*)

- Capturas claras de todos los comandos ejecutados (*10%*).
- Capturas de archivos de configuración modificados (*10%*).
- Capturas de verificación de los cuatro servicios funcionando simultáneamente (*10%*).

SOLUCIÓN DE PROBLEMAS COMUNES

- **Si un puerto ya está en uso:**
  sudo lsof -i :PUERTO
- **Si un servicio no arranca:**
  sudo journalctl -u SERVICIO -n 50
- **Si Apache/Nginx fallan:** revisa los logs en:
  /var/log/apache2/ o /var/log/nginx/