

# TEMARIO: ARQUITECTURA EN LA NUBE - ADMINISTRACIÓN REMOTA DE SERVIDORES WEB EN AWS A TRAVÉS DE SSH

## COMANDOS Y PROCEDIMIENTOS

### PARTE 0: PREPARACIÓN DEL ENTORNO LOCAL (WSL/VM)

#### 1. Verificar WSL2

`wsl --version`

**Descripción:** Verifica que WSL2 esté instalado y funcionando correctamente.

```
Wslman Shell commandLine, version 0.2.1

USAGE: wsl COMMAND [PARAMS...]

COMMANDS:
identify - WS-Identify
enum     - WS-Enumerate
get      - WS-Get
put      - WS-Put
invoke   - WS-Invoke
xclean   - Delete all files generated by this tool set
xcred    - Create or display credential file
xcert    - Get server certificate (saved to <IPADDRESS>.crt)

PARAMS specification is specific to a COMMAND.

Output will be saved to ./response.xml. If you want to run parallel
executions in the same directory, define RTFILEPREFIX in the environment.
Doing so may significantly increase files generated.

Requires: curl, xmllint, GNU core utilities.
Optional: xsltproc for output formatting, gpg for encrypted credential.
Optional: wget as alternate for curl when not available.
```

## 2. Crear directorio para las claves SSH

```
mkdir -p ~/.ssh
```

```
chmod 700 ~/.ssh
```

**Descripción:** Crea el directorio .ssh con permisos correctos para almacenar claves.

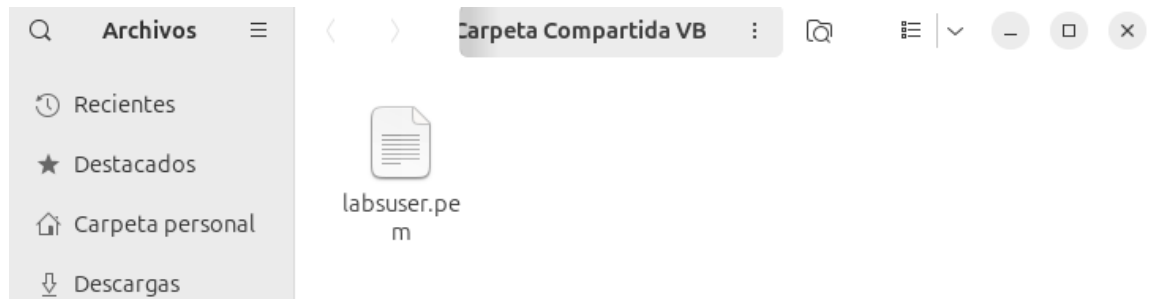
```
root@santiago-VirtualBox:/home/santiago# mkdir -p ~/.ssh
root@santiago-VirtualBox:/home/santiago# chmod 700 ~/.ssh
```

## PARTE 1: CONFIGURACIÓN EN AWS (Interfaz Visual)

### 1. Descargar la clave PEM desde la página del laboratorio

1. Localiza el botón o enlace “**Download key**” o “**Download .pem**” 2. Se descargará un archivo con extensión **.pem** (por ejemplo: **labsuser.pem**)

3. **Importante:** Guarda este archivo en una ubicación segura, preferiblemente en WSL



## Pasos para mover la clave a WSL:

Si descargaste el archivo en Windows:

```
cp /mnt/c/Users/TU-USUARIO/Downloads/labsuser.pem
```

~/ssh/ O si prefieres descargarlo directamente en WSL:

```
cp ~/Downloads/labsuser.pem ~/.ssh/
```

## 2. Configurar permisos de la clave PEM

```
chmod 400 ~/.ssh/labsuser.pem
```

Verifica los permisos:

```
ls -la ~/.ssh/labsuser.pem
```

# Deberías ver: -r-----

```
root@santiago-VirtualBox:/home/santiago# mkdir -p ~/.ssh
root@santiago-VirtualBox:/home/santiago# chmod 700 ~/.ssh
root@santiago-VirtualBox:/home/santiago# cd Descargas
root@santiago-VirtualBox:/home/santiago/Descargas# cp labsuser.pem ~/.ssh/
root@santiago-VirtualBox:/home/santiago/Descargas# chmod 400 ~/.ssh/labsuser.pem
root@santiago-VirtualBox:/home/santiago/Descargas# ls -la ~/.ssh/labsuser.pem
-r----- 1 root root 1674 oct 31 10:59 /root/.ssh/labsuser.pem
```

## 3. Crear instancia EC2

1. **Nombre:** servidor-web-practica
2. **AMI:** Ubuntu 22.04 LTS o Ubuntu 24.04 LTS
3. **Tipo:** t2/3.micro (*Free Tier*)
4. **Red:** VPC y Subred por defecto
5. **IP pública:** Habilitada
6. **Almacenamiento:** 8 GiB, tipo gp2
7. **Etiquetas (*opcional*):** Curso → ArquitecturaNube
8. **Grupo de Seguridad:** sg-servidores-web

#### 4. Configurar Security Group (*Reglas de entrada*)

Puerto	Protocolo	Tipo	Descripción
22	TCP	SSH	Acceso SSH remoto
8080	TCP	HTTP personalizado	Apache HTTP
8081	TCP	HTTP personalizado	Nginx
8082	TCP	HTTP personalizado	Caddy
8443	TCP	HTTPS personalizado	Apache HTTPS (SSL)

#### 5. Especificar la clave PEM al lanzar la instancia

Selecciona **Use existing key pair** y elige la clave **labsuser.pem**.

Si la instancia fue creada sin clave, termina la instancia y crea una nueva seleccionando la clave correcta.

## PARTE 2: CONEXIÓN SSH DESDE WSL A AWS

### 1. Obtener la IP pública de la instancia

En AWS Console:

**EC2 → Instances → Public IPv4 address**

### 2. Conectar por SSH

```
ssh -i ~/.ssh/labsuser.pem ubuntu@TU-IP-PUBLICA
```

Ejemplo:

```
ssh -i ~/.ssh/labsuser.pem ubuntu@54.123.45.67
```

```
root@santiago-VirtualBox:/# ssh -i ~/.ssh/labsuser.pem ubuntu@34.233.126.185
The authenticity of host '34.233.126.185 (34.233.126.185)' can't be established.
ED25519 key fingerprint is SHA256:dckaQHJGKHRKo/b4Nn/ah9IAWCVVEEPqxVvo3ztyM10.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '34.233.126.185' (ED25519) to the list of known hosts
.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Fri Oct 31 10:05:03 UTC 2025

System load:  0.0           Temperature:   -273.1 C
Usage of /:   32.8% of 6.71GB Processes:    115
Memory usage: 23%          Users logged in: 0
Swap usage:   0%           IPv4 address for ens5: 172.31.26.129

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.
```

# 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.

```
ubuntu@ip-172-31-26-129:~$ sudo apt update && sudo apt upgrade -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1573 kB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [297 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.4 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1498 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [303 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.4 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted a
```

## 2. Instalar Apache2

Comando:

```
sudo apt install apache2 -y
```

**Descripción:** Instala el servidor web Apache en tu sistema.

```
ubuntu@ip-172-31-26-129:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-aws-6.14-headers-6.14.0-1011 linux-aws-6.14-tools-6.14.0-1011
  linux-headers-6.14.0-1011-aws linux-image-6.14.0-1011-aws
  linux-modules-6.14.0-1011-aws linux-tools-6.14.0-1011-aws
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1t64 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libaprutil1t64 liblua5.4-0 ssl-cert
```

## 3. Configurar Apache en puerto 8080

Comando:

```
sudo nano /etc/apache2/ports.conf
```

**Descripción:** Abre el archivo de configuración de puertos. Cambia **Listen 8** por **Listen 8080**.

```
GNU nano 7.2 /etc/apache2/ports.conf *
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 8080
```

#### 4. Modificar el VirtualHost

Comando:

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

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

```
GNU nano 7.2 /etc/apache2/sites-available/000-default.conf *
<VirtualHost *:8080>
    # The ServerName directive sets the request scheme, hostname and port that
    # 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) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com
```

#### 5. Instalar PHP

Comando:

```
sudo apt install php libapache2-mod-php -y
```

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

```
ubuntu@ip-172-31-26-129:~$ sudo apt install php libapache2-mod-php -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-aws-6.14-headers-6.14.0-1011 linux-aws-6.14-tools-6.14.0-1011
  linux-headers-6.14.0-1011-aws linux-image-6.14.0-1011-aws
  linux-modules-6.14.0-1011-aws linux-tools-6.14.0-1011-aws
```

#### 6. Reiniciar Apache

Comando:

```
sudo systemctl restart apache2
```

Descripción: Reinicia Apache para aplicar los cambios.

```
ubuntu@ip-172-31-26-129:~$ sudo systemctl restart apache2
```

## 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*).

```
ubuntu@ip-172-31-26-129:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: >
   Active: active (running) since Fri 2025-10-31 10:15:27 UTC; 34s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 18721 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/S>
  Main PID: 18724 (apache2)
    Tasks: 6 (limit: 1017)
   Memory: 10.7M (peak: 11.1M)
      CPU: 53ms
   CGroup: /system.slice/apache2.service
           └─18724 /usr/sbin/apache2 -k start
             └─18726 /usr/sbin/apache2 -k start
               └─18727 /usr/sbin/apache2 -k start
                 └─18728 /usr/sbin/apache2 -k start
                   └─18729 /usr/sbin/apache2 -k start
                     └─18730 /usr/sbin/apache2 -k start
```

## 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.

```
ubuntu@ip-172-31-26-129:~$ echo "<?php phpinfo(); ?>" | sudo tee /var/www/html/info.php
<?php phpinfo(); ?>
```

## 9. Probar Apache desde terminal

Comando:


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

Descripción: Verifica que Apache sirve correctamente el contenido PHP.

Recibid | Página | Segund | Cuarta | Lanzam | Detalle | Arquite | PH | x | + | - | □ | ×

← | → | ↻ | Not secure | 34.233.126.185:8080/info.php | ☆ | 🔒 | 🛡️ | 📄 | 📥 | 📤 | ⋮


PHP Version 8.3.6



System	Linux ip-172-31-26-129 6.14.0-1015-aws #15~24.04.1-Ubuntu SMP Tue Sep 23 22:44:48 UTC 2025 x86_64
Build Date	Jul 14 2025 18:30:55
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/apache2
Loaded Configuration File	/etc/php/8.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/apache2/conf.d
Additional .ini files parsed	/etc/php/8.3/apache2/conf.d/10-opcache.ini, /etc/php/8.3/apache2/conf.d/10-pdo.ini, /etc/php/8.3/apache2/conf.d/20-calendar.ini, /etc/php/8.3/apache2/conf.d/20-ctype.ini, /etc/php/8.3/apache2/conf.d/20-exif.ini, /etc/php/8.3/apache2/conf.d/20-fileinfo.ini, /etc/php/8.3/apache2/conf.d/20-ftp.ini, /etc/php/8.3/apache2/conf.d/20-gettext.ini, /etc/php/8.3/apache2/conf.d/20-iconv.ini, /etc/php/8.3/apache2/conf.d/20-phar.ini, /etc/php/8.3/apache2/conf.d/20-posix.ini, /etc/php/8.3/apache2/conf.d/20-readline.ini, /etc/php/8.3/apache2/conf.d/20-shmop.ini, /etc/php/8.3/apache2/conf.d/20-sockets.ini, /etc/php/8.3/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.3/apache2/conf.d/20-sysvsem.ini, /etc/php/8.3/apache2/conf.d/20-sysvshm.ini, /etc/php/8.3/apache2/conf.d/20-tokenizer.ini
PHP API	20230831
PHP Extension	20230831
Zend Extension	420230831
Zend Extension Build	API420230831,NTS
PHP Extension Build	API20230831,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
Zend Max Execution Timers	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:  
Zend Engine v4.3.6, Copyright (c) Zend Technologies with Zend OPcache v8.3.6, Copyright (c), by Zend Technologies

zendengine

Configuration

## 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.

```
root@santiago-VirtualBox:/# sudo apt install nginx -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  nginx-common
Paquetes sugeridos:
  fcgiwrap nginx-doc
Se instalarán los siguientes paquetes NUEVOS:
  nginx nginx-common
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 80 no actualizados.
Se necesita descargar 564 kB de archivos.
Se utilizarán 1.596 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx-common all 1.2
4.0-2ubuntu7.5 [43,4 kB]
Des:2 http://es.archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx amd64 1.24.0-2
ubuntu7.5 [520 kB]
```

### 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**.

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
```

### 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.

```
root@santiago-VirtualBox:/# echo "<h1>Servidor Nginx</h1><p>Funcionando en puerto 8081<  
/p>" | sudo tee /usr/share/nginx/html/index.html  
<h1>Servidor Nginx</h1><p>Funcionando en puerto 8081</p>
```

### 4. Reiniciar Nginx

Comando:

```
sudo systemctl restart nginx
```

**Descripción:** Reinicia Nginx para aplicar los cambios de configuración.

```
root@santiago-VirtualBox:/# sudo systemctl restart nginx
```

## 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*).

```
root@santiago-VirtualBox:/# sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Fri 2025-11-07 09:02:09 CET; 1min 10s ago
     Docs: man:nginx(8)
   Process: 5491 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; >
   Process: 5495 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exi>
  Main PID: 5496 (nginx)
    Tasks: 5 (limit: 4602)
   Memory: 3.7M (peak: 3.9M)
      CPU: 18ms
   CGroup: /system.slice/nginx.service
           └─5496 "nginx: master process /usr/sbin/nginx -g daemon on; master_proces>
             └─5497 "nginx: worker process"
               └─5498 "nginx: worker process"
                 └─5499 "nginx: worker process"
                   └─5500 "nginx: worker process"

nov 07 09:02:09 santiago-VirtualBox systemd[1]: Starting nginx.service - A high perfor>
nov 07 09:02:09 santiago-VirtualBox systemd[1]: Started nginx.service - A high perform>
lines 1-19/19 (END)
```

```
tcp        LISTEN      0          511          [::]:80          [::]:*
users:(("nginx",pid=5500,fd=6),("nginx",pid=5499,fd=6),("nginx",pid=5498,fd=6),("nginx",pid=5497,fd=6),("nginx",pid=5496,fd=6))
```

## 6. Probar Nginx desde terminal

Comando:

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

**Descripción:** Verifica que Nginx sirve correctamente el contenido HTML.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</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.

```
root@santiago-VirtualBox:/# sudo apt install -y debian-keyring debian-archive-keyring a
pt-transport-https curl
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
curl ya está en su versión más reciente (8.5.0-2ubuntu10.6).
fijado curl como instalado manualmente.
Se instalarán los siguientes paquetes NUEVOS:
  apt-transport-https debian-archive-keyring debian-keyring
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 80 no actualizados.
Se necesita descargar 31,5 MB de archivos.
Se utilizarán 33,4 MB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu noble-updates/universe amd64 apt-transport-ht
tps all 2.8.3 [3.970 B]
Des:2 http://es.archive.ubuntu.com/ubuntu noble/universe amd64 debian-archive-keyring a
ll 2023.4ubuntu1 [168 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu noble/universe amd64 debian-keyring all 2023.
```

## 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.

```
root@Ubuntu1:/home/vboxuser# curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/
gpg.key' | sudo gpg -- dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg
gpg: directory '/root/.gnupg' created
gpg: keybox '/root/.gnupg/pubring.kbx' created
gpg: WARNING: no command supplied. Trying to guess what you mean ...
usage: gpg [options] [filename]

root@Ubuntu1:/home/vboxuser# curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/
debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list
# Source: Caddy
# Site: https://github.com/caddyserver/caddy
# Repository: Caddy / stable
# Description: Fast, multi-platform web server with automatic HTTPS

deb [signed-by=/usr/share/keyrings/caddy-stable-archive-keyring.gpg] https://dl.clouds
mith.io/public/caddy/stable/deb/debian any-version main

deb-src [signed-by=/usr/share/keyrings/caddy-stable-archive-keyring.gpg] https://dl.cl
oudsmith.io/public/caddy/stable/deb/debian any-version main
```

### 3. Actualizar e instalar Caddy

Comando:

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

**Descripción:** Actualiza la lista de paquetes e instala Caddy.

```
root@Ubuntu1:/home/vboxuser# sudo apt update && sudo apt install caddy -y
Hit:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:2 http://es.archive.ubuntu.com/ubuntu noble InRelease
Hit:3 http://es.archive.ubuntu.com/ubuntu noble-updates InRelease
Get:4 https://dl.cloudsmith.io/public/caddy/stable/deb/debian any-version InRelease [14,8 kB]
Hit:5 http://es.archive.ubuntu.com/ubuntu noble-backports InRelease
Get:6 https://dl.cloudsmith.io/public/caddy/stable/deb/debian any-version/main amd64 Packages [4.329 B]
Fetched 19,1 kB in 0s (43,2 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
20 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
```

### 4. Crear directorio para Caddy

Comando:

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

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

```
root@Ubuntu1:/home/vboxuser# sudo mkdir -p /var/www/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.

```
| GNU nano 7.2 prueba.md *
echo "# Bienvenido a Caddy" | sudo tee /var/www/caddy/README.md echo "" | sudo tee -a>
echo "Este servidor está funcionando correctamente." | sudo tee -a /var/www/caddy/RE>
echo "" | sudo tee -a /var/www/caddy/README.md
echo "## Características" | sudo tee -a /var/www/caddy/README.md echo "- Servidor mod>
```

## 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.

```
root@Ubuntu1:/home/vboxuser# curl -o /tmp/test-image.jpg "https://www.python.org/stati
c/apple-touch-icon-144x144-precomposed.png"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 7382 100 7382    0     0 234k      0 --:--:-- --:--:-- --:--:-- 240k
root@Ubuntu1:/home/vboxuser# sudo mv /tmp/test-image.jpg /var/www/caddy/test.jpg
```

## 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
```

```
}
```

```
GNU nano 7.2 /etc/caddy/Caddyfile *
# To use your own domain name (with automatic HTTPS), first make
# sure your domain's A/AAAA DNS records are properly pointed to
# this machine's public IP, then replace ":80" below with your
# domain name.

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

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

# Refer to the Caddy docs for more information:
# https://caddyserver.com/docs/caddyfile
```

## 8. Reiniciar Caddy

Comando:

```
sudo systemctl restart caddy
```

**Descripción:** Reinicia Caddy para aplicar la nueva configuración.

```
root@Ubuntu1:/home/vboxuser# sudo systemctl restart caddy
```

## 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*).

```
root@Ubuntu1:/home/vboxuser# sudo systemctl status caddy
● caddy.service - Caddy
   Loaded: loaded (/usr/lib/systemd/system/caddy.service; enabled; preset: enabled)
   Active: active (running) since Fri 2025-10-10 09:29:18 UTC; 33s ago
     Docs: https://caddyserver.com/docs/
   Main PID: 10569 (caddy)
    Tasks: 8 (limit: 4604)
   Memory: 11.9M (peak: 12.5M)
      CPU: 73ms
   CGroup: /system.slice/caddy.service
           └─10569 /usr/bin/caddy run --environ --config /etc/caddy/Caddyfile
```

## 10. Probar Caddy desde terminal

Comando:

```
curl http://localhost:8082/
```

**Descripción:** Lista los archivos disponibles en el servidor Caddy.

```
root@Ubuntu1:/home/vboxuser# curl http://localhost:8082/

<!DOCTYPE html>
<html>
  <head>
    <title>/</title>
    <link rel="canonical" href="/" />
    <meta charset="utf-8">
    <meta name="color-scheme" content="light dark">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style nonce="0d043bab-d8cb-4e43-838e-422d02e36b0a">
  * { padding: 0; margin: 0; box-sizing: border-box; }
  body {
    font-family: Inter, system-ui, sans-serif;
    font-size: 16px;
    text-rendering: optimizespeed;
    background-color: #f3f6f7;
    min-height: 100vh;
  }
```

## 11. Probar archivo Markdown

Comando:

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

**Descripción:** Verifica que Caddy sirve correctamente archivos Markdown.

```
root@Ubuntu1:/home/vboxuser# curl http://localhost:8082/README.md
# Bienvenido a Caddy
# Bienvenido a Caddy
```

## 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.

```
root@Ubuntu1:/home/vboxuser# sudo apt install certbot python3-certbot-apache -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
```

## 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.

```
Country Name (2 letter code) [AU]:ES  
State or Province Name (full name) [Some-State]:Madrid  
Locality Name (eg, city) []:Madrid  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Internet widgits pty ltd  
Organizational Unit Name (eg, section) []:eg,section  
Common Name (e.g. server FQDN or YOUR name) []:server FQDN  
Email Address []:ronin@gmail.com  
root@Ubuntu1:/home/vboxuser#
```

## 3. Habilitar módulo SSL en Apache

**Comando:**

```
sudo a2enmod ssl
```

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

```
root@Ubuntu1:/home/vboxuser# sudo a2enmod ssl  
Considering dependency mime for ssl:  
Module mime already enabled  
Considering dependency socache_shmcb for ssl:  
Enabling module socache_shmcb.  
Enabling module ssl.  
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-si  
gned certificates.  
To activate the new configuration, you need to run:  
systemctl restart apache2
```

## 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.

```
GNU nano 7.2 /etc/apache2/sites-available/default-ssl.conf
<VirtualHost *:443>

    SSLEngine on

    # A self-signed (snakeoil) certificate can be created by installing
    # the ssl-cert package. See
    # /usr/share/doc/apache2/README.Debian.gz for more info.
    # If both key and certificate are stored in the same file, only the
    # SSLCertificateFile directive is needed.
    SSLCertificateFile      /etc/ssl/certs/ssl-cert-snakeoil.pem
    SSLCertificateKeyFile   /etc/ssl/private/ssl-cert-snakeoil.key
```

## 5. Modificar VirtualHost SSL

Comando:

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

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

```
GNU nano 7.2 /etc/apache2/ports.conf *
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 8080
Listen 8443
<IfModule ssl_module>
    Listen 443

GNU nano 7.2 /etc/apache2/sites-available/default-ssl.conf *
<VirtualHost *:8443>
```

## 6. Habilitar sitio SSL

Comando:

```
sudo a2ensite default-ssl.conf
```

**Descripción:** Activa la configuración SSL en Apache.

```
root@Ubuntu1:/home/vboxuser# sudo a2ensite default-ssl.conf
Site default-ssl already enabled
```

## 7. Reiniciar Apache

Comando:

```
sudo systemctl restart apache2
```

**Descripción:** Aplica todos los cambios de configuración SSL.

```
root@Ubuntu1:/home/vboxuser# sudo systemctl restart apache2
```

## 8. Verificar HTTPS

Comando:

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

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

```
root@Ubuntu1:/home/vboxuser# curl -i -k https://localhost:8443
HTTP/1.1 200 OK
Date: Fri, 10 Oct 2025 10:14:13 GMT
Server: Apache/2.4.58 (Ubuntu)
Last-Modified: Fri, 03 Oct 2025 07:54:29 GMT
ETag: "29af-6403c694cc265"
Accept-Ranges: bytes
Content-Length: 10671
Vary: Accept-Encoding
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/x
```

## **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.

```
root@Ubuntu1:/home/vboxuser# sudo systemctl status apache2 nginx caddy
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enable>
   Active: active (running) since Fri 2025-10-10 10:13:54 UTC; 1min 30s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 11396 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 11399 (apache2)
```

### **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.

```
root@Ubuntu1:/home/vboxuser# curl http://localhost:8080
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
      * {
        margin: 0px 0px 0px 0px;
        padding: 0px 0px 0px 0px;
      }

      body, html {
        padding: 3px 3px 3px 3px;

        background-color: #D8DBE2;

        font-family: Ubuntu, Verdana, sans-serif;
```

```
root@Ubuntu1:/home/vboxuser# curl http://localhost:8081
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
      * {
        margin: 0px 0px 0px 0px;
        padding: 0px 0px 0px 0px;
      }

      body, html {
        padding: 3px 3px 3px 3px;
```

```
root@Ubuntu1:/home/vboxuser# curl http://localhost:8082
```

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <link rel="canonical" href="/" />
    <meta charset="utf-8">
    <meta name="color-scheme" content="light dark">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
<style nonce="741b58c1-83ff-4133-a2b5-185a6d8d580a">
* { padding: 0; margin: 0; box-sizing: border-box; }

body {
  font-family: Inter, system-ui, sans-serif;
  font-size: 16px;
  text-rendering: optimizespeed;
  background-color: #f3f6f7;
  min-height: 100vh;
}

img,
svg {
```

```
root@Ubuntu1:/home/vboxuser# curl -k https://localhost:8443
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
* {
  margin: 0px 0px 0px 0px;
  padding: 0px 0px 0px 0px;
}

body, html {
  padding: 3px 3px 3px 3px;

  background-color: #D8DBE2;
```

# EXPLICACIÓN ESCRITA DE CADA CAPTURA

1. **Descarga de la clave PEM:**  
Descargamos la clave .pem para poder autenticarnos por SSH en la instancia EC2.
2. **Mover la clave a .ssh:**  
Movemos la clave al directorio seguro .ssh para protegerla y usarla correctamente.
3. **Cambio de permisos de la clave:**  
Ajustamos los permisos de la clave para que solo el usuario pueda leerla.
4. **Creación de la instancia EC2:**  
Creamos una instancia Ubuntu en AWS que actuará como servidor web.
5. **Configuración del Security Group:**  
Abrimos los puertos necesarios para acceder por SSH y para los servidores web.
6. **Obtención de la IP pública:**  
Localizamos la IP pública de la instancia para conectarnos de forma remota.
7. **Conexión SSH desde WSL:**  
Nos conectamos a la instancia mediante SSH usando la clave privada.

## Parte de Apache (8080)

8. **Instalación de Apache:**  
Instalamos el servidor web Apache en el sistema.
9. **Cambio al puerto 8080:**  
Configuramos Apache para escuchar en el puerto 8080.
10. **Creación del archivo info.php:**  
Creamos una página PHP de prueba para verificar el funcionamiento.
11. **Prueba de Apache con curl:**  
Comprobamos que Apache sirve la página correctamente.

## Parte de Nginx (8081)

12. **Instalación de Nginx:**  
Instalamos el servidor web Nginx en el sistema.
13. **Cambio al puerto 8081:**  
Modificamos la configuración para que Nginx escuche en el puerto 8081.
14. **Página HTML personalizada:**  
Creamos una página HTML para identificar el servidor Nginx.
15. **Prueba de Nginx:**  
Verificamos que Nginx responde correctamente en su puerto.

## Parte de Caddy (8082)

16. **Instalación de Caddy:**  
Instalamos el servidor web Caddy, que gestiona HTTPS automáticamente.
17. **Creación de directorio y archivos:**  
Creamos contenido de prueba para servir desde Caddy.
18. **Configuración del Caddyfile:**  
Configuramos Caddy para servir en el puerto 8082.
19. **Prueba de Caddy:**  
Comprobamos que Caddy lista y muestra correctamente los archivos.

## HTTPS con Apache (8443)

20. **Creación de certificado autofirmado:**  
Generamos un certificado SSL para habilitar HTTPS sin dominio público.
21. **Habilitación de SSL en Apache:**  
Activamos el módulo SSL para permitir conexiones seguras.
22. **Apache escuchando en 8443:**  
Configuramos Apache para servir HTTPS en el puerto 8443.
23. **Prueba de HTTPS:**  
Verificamos la conexión HTTPS ignorando el aviso del certificado.

## **Verificación final**

### **24. Estado de los servicios:**

Comprobamos que Apache, Nginx y Caddy están activos simultáneamente.

### **25. Comprobación de puertos:**

Verificamos que cada servidor escucha correctamente en su puerto asignado.