

P2 Balanceo de Carga

Mario Rodriguez Lopez

March 2024

Índice

B1.CREACIÓN DEL ESPACIO DE TRABAJO SSL	2
B2. CREACIÓN DE CERTIFICADOS SSL	2
B3. CONFIGURACIÓN DE SERVIDORES WEB APACHE CON SSL	3
B4. CONFIGURACIÓN DEL BALANCEADOR DE CARGA NGINX CON SSL	5
B5. DOCKER COMPOSE PARA LA GRANJA WEB CON SSL	5
B6. VERIFICACIÓN Y PRUEBAS DEL ESCENARIO CON SSL	8
A1. CREACIÓN DE CERTIFICADOS SSL	10
A2. OPTIMIZACIÓN DE CONFIGURACIÓN SSL EN LOS SERVIDORES WEB	12
A3. CONFIGURACIÓN DE CACHÉ DE SESIONES SSL Y TICKETS DE SESION EN EL BALANCEADOR	14
A4. OPTIMIZACIÓN DE CONEXIONES HTTPS Y CIFRADO EN EL BALANCEADOR	16

B1.CREACIÓN DEL ESPACIO DE TRABAJO SSL

Creamos los directorios como sigue:

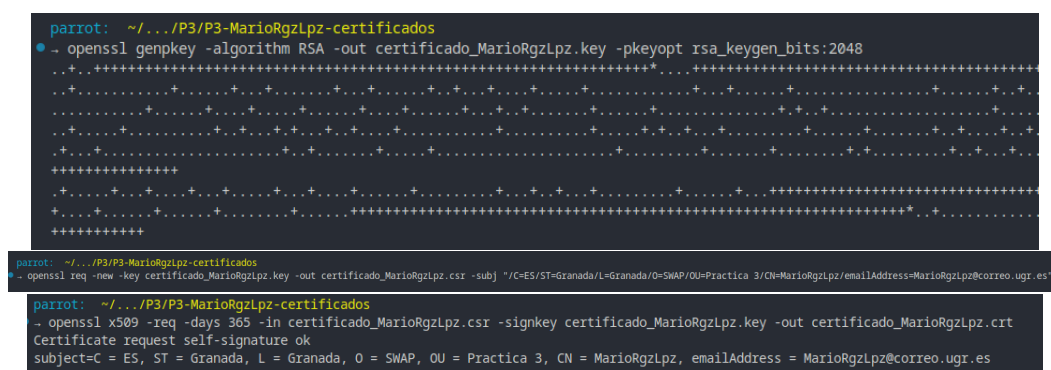
```
mkdir P3-MarioRgzLpz-nginx
mkdir P3-MarioRgzLpz-apache
mkdir P3-MarioRgzLpz-certificados
```

No he usado IA para realizar esto puesto que es muy simple.

B2. CREACIÓN DE CERTIFICADOS SSL

B2.1. Propuesta final

Simplemente he seguido los pasos que me ha dado ChatGPT adaptando a mi nombre de usuario y he visto que funcionasen correctamente. Aquí muestro el resultado de ejecutarlos.



```
parrot: ~/.../P3/P3-MarioRgzLpz-certificados
└─ openssl genpkey -algorithm RSA -out certificado_MarioRgzLpz.key -pkeyopt rsa_keygen_bits:2048
.....
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = SWAP, OU = Practica 3, CN = MarioRgzLpz, emailAddress = MarioRgzLpz@correo.ugr.es

parrot: ~/.../P3/P3-MarioRgzLpz-certificados
└─ openssl req -new -key certificado_MarioRgzLpz.key -out certificado_MarioRgzLpz.csr -subj "/C=ES/ST=Granada/L=Granada/O=SWAP/OU=Practica 3/CN=MarioRgzLpz/emailAddress=MarioRgzLpz@correo.ugr.es"

parrot: ~/.../P3/P3-MarioRgzLpz-certificados
└─ openssl x509 -req -days 365 -in certificado_MarioRgzLpz.csr -signkey certificado_MarioRgzLpz.key -out certificado_MarioRgzLpz.crt
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = SWAP, OU = Practica 3, CN = MarioRgzLpz, emailAddress = MarioRgzLpz@correo.ugr.es
```

B2.2 Analisis Propuesta IA

Pregunta a ChatGPT

En esta parte crearemos los archivos necesarios para el certificado SSL: clave privada y certificado autofirmado con OpenSSL en el directorio P3-tuusuariougr-certificados. El certificado SSL deberá tener las siguientes especificaciones:

- Generar un certificado: Generar un certificado autofirmado en lugar de una solicitud de firma de certificado.
- Validez de 1 año: El certificado debe ser válido por 365 días desde el momento de su creación.
- Encriptación RSA de 2048 bits: La clave privada asociada con el certificado debe ser una clave RSA con una longitud de 2048 bits para asegurar una encriptación fuerte.
- Sin necesidad de passphrase: La clave privada no debe requerir una passphrase para facilitar su uso en automatizaciones y evitar la intervención manual durante el reinicio de los servicios.
- Autofirmado: El certificado debe ser autofirmado, lo que significa que la misma entidad que lo crea, lo firma, lo que es adecuado para entornos de prueba.
- Clave privada: El archivo con la clave privada debe llamarse certificado_tuusuariougr.key
- Certificado autofirmado: El archivo con el certificado autofirmado debe llamarse certificado_tuusuariougr.crt
- Datos para el certificado de dominio:
- Nombre de país: ES
- Provincia: Granada
- Localidad: Granada
- Organización: SWAP
- Organización sección: Práctica 3
- Nombre: "tu usuario_ugr"
- Email: "email_ugr"

Respuesta ChatGPT

Para crear los archivos necesarios para el certificado SSL con las especificaciones dadas utilizando OpenSSL, sigue estos pasos:

Abre una terminal o línea de comandos.

Navega al directorio donde deseas crear los archivos (P3-tuusuariougr-certificados).

Ejecuta los siguientes comandos uno por uno:

```
# Generar la clave privada RSA de 2048 bits
openssl genpkey -algorithm RSA -out certificado_tuusuariougr.key -pkeyopt rsa_keygen_bits:2048

# Generar la solicitud de certificado autofirmado
openssl req -new -key certificado_tuusuariougr.key -out certificado_tuusuariougr.csr -subj "/C=ES/ST=Granada/L=Granada/O=SWAP/OU=Practica 3/CN=tu_usuario_ugr/emailAddress=email_ugr"

# Crear el certificado autofirmado valido por 1 ano
openssl x509 -req -days 365 -in certificado_tuusuariougr.csr -signkey certificado_tuusuariougr.key -out certificado_tuusuariougr.crt

# Eliminar la solicitud de certificado (opcional)
rm certificado_tuusuariougr.csr
```

Esto generará dos archivos en el directorio especificado:

certificado_tuusuariougr.key: La clave privada.

certificado_tuusuariougr.crt: El certificado autofirmado.

Estos archivos cumplirán con todas las especificaciones dadas

Análisis Lo que me proporciona ChatGPT es correcto por tanto me basta con usar eso para crear tanto el certificado autofirmado como la llave.

B3. CONFIGURACIÓN DE SERVIDORES WEB APACHE CON SSL

B3.1. Propuesta final

Para este apartado he seguido el guion de la práctica casi para todo. He copiado la configuración que tenía de la P1 para que en los siguientes apartados me funcione todo de manera correcta. Me ha dado problema la creación de la imagen por como funciona docker con los directorios y para crear la imagen he tenido que hacerlo desde la carpeta principal y no desde el directorio P3-apache, la línea siguiente es como lo he hecho:

```
docker build -t mariorgzlpz-apache-image:p3 -f P3-MarioRgzLpz-apache/DockerfileApacheP3 .
```

Lo que muestro ahora es la imagen que me queda del DockerFile de Apache:

```
# Use an Alpine Linux base image
FROM alpine:3.15.0

# Install Apache, PHP, and networking tools
RUN apk add --no-cache \
    bash \
    php8-apache2 \
    php8 \
    apache2 \
    apache2-ssl

# Copy certificate and private key
COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt /etc/apache2/ssl/certificado_MarioRgzLpz.crt
COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key /etc/apache2/ssl/certificado_MarioRgzLpz.key

# Copy Apache configuration files
COPY /P3-MarioRgzLpz-apache/configsapache/httpd.conf /etc/apache2/httpd.conf
COPY /P3-MarioRgzLpz-apache/SSL-MarioRgzLpz-apache.conf /etc/apache2/conf.d/SSL-MarioRgzLpz-apache.conf

# Enable SSL module
RUN sed -i 's/^#LoadModule ssl_module/s^#/g' /etc/apache2/httpd.conf

# Expose port 80 for Apache
EXPOSE 80
EXPOSE 443
```

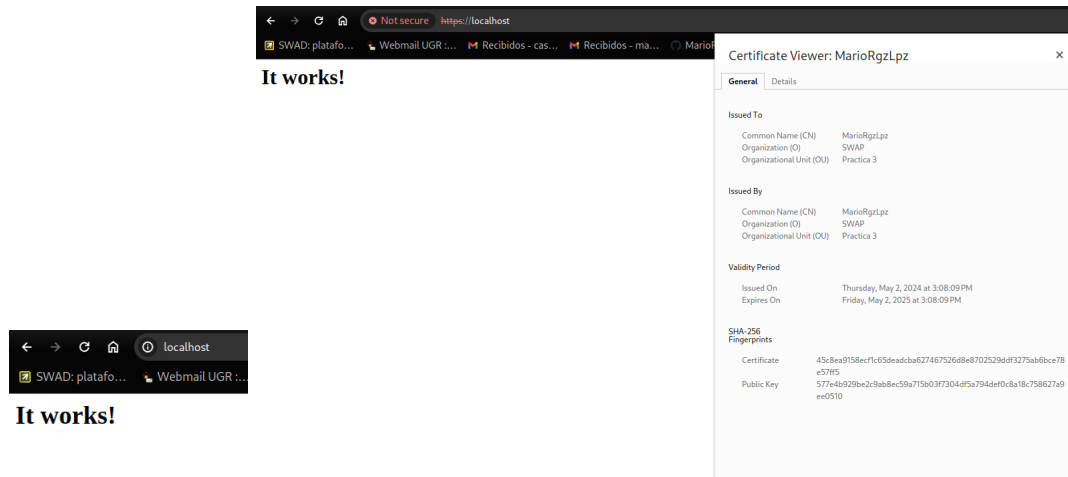
```
# Command to start Apache in the foreground
CMD [ "httpd" , "-D" , "FOREGROUND" ]
```

Aqui he modificado varias cosas una para activar el modulo ssl en apache2 que para esto primero he preguntado a ChatGPT y no me lo ha solucionado y después he buscado en google .alpine apache2 sslz me ha aparecido como primer enlace un paquete de alpine que es apache2-ssl y tras ponerlo en el DockerFile ya me funcionaba perfectamente.

ChatGPT es el que me ha proporcionado la linea con la que pongo que se cargue el modulo ssl en la configuracion.

Como no he copiado el fichero de web porque pretendo montar un volumen en el docker compose me aparece la pagina por defecto de que esta funcionando apache2 tanto en http como en https.

Asi me queda el acceso a localhost en http y https



B3.2. Analisis Propuesta IA

Preguntas a ChatGPT —Cuando lanzo un contenedor con la imagen creada a partir de este DockerFile (dockerfile de la P1 con los certificados y la configuracion) me aparece lo siguiente en los logs

docker logs prueba-apache

```
AH00526: Syntax error on line 3 of /etc/apache2/conf.d/SSL-MarioRgzLpz-apache.conf:
Invalid command 'SSEngine', perhaps misspelled or defined by a module not included
in the server configuration
```

—Me sigue fallando.

Respuestas ChatGPT —Instead of uncommenting the LoadModule rewrite_module line, let's directly enable the SSL module using the a2enmod command. Here's how you can modify your Dockerfile:

```
# Use an Alpine Linux base image
FROM alpine:3.15.0

# Install Apache, PHP, and networking tools
RUN apk add --no-cache \
    bash \
    php8-apache2 \
    php8 \
    apache2

# Enable SSL module
RUN sed -i '/^#LoadModule ssl_module/s/^#//g' /etc/apache2/httpd.conf &&

# Copy Apache configuration files
COPY P3-MarioRgzLpz-apache/configs/apache/httpd.conf /etc/apache2/httpd.conf
COPY P3-MarioRgzLpz-apache/SSL-MarioRgzLpz-apache.conf /etc/apache2/conf.d/SSL-MarioRgzLpz-apache.conf

# Copy certificate and private key
COPY P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt /etc/apache2/ssl/certificado_MarioRgzLpz.crt
```

```

COPY P3-MarioRgzLpz-certificados/certificado-MarioRgzLpz.key /etc/apache2/ssl/
certificado-MarioRgzLpz.key

# Expose port 80 for Apache
EXPOSE 80

# Command to start Apache in the foreground
CMD [ "httpd", "-D", "FOREGROUND" ]

```

—Me vuelve a dar la misma respuesta que antes y que tampoco me funciona. Aquí es cuando busco en google y encuentro la solución.

Análisis

Las respuestas de ChatGPT me han servido para activar el modulo pero mi problema no solo estaba ahí si no que me faltaba tener instalado dicho módulo ya que por defecto en el paquete de apache2 de alpine no viene. Tras ese par de cambios lo configuro de manera correcta.

B4. CONFIGURACIÓN DEL BALANCEADOR DE CARGA NGINX CON SSL

B4.1. Propuesta final

Siguiendo los pasos del guion me queda el siguiente DockerFile:

```

#Imagen base
FROM nginx:latest

# Crear directorio para SSL
RUN mkdir -p /etc/nginx/ssl

# Copiar certificados SSL al contenedor
COPY /P3-MarioRgzLpz-certificados/certificado-MarioRgzLpz.crt /etc/nginx/ssl/
certificado-MarioRgzLpz.crt
COPY /P3-MarioRgzLpz-certificados/certificado-MarioRgzLpz.key /etc/nginx/ssl/
certificado-MarioRgzLpz.key

# Incluir configuracion de Nginx para SSL
COPY /P3-MarioRgzLpz-nginx/MarioRgzLpz-nginx-ssl.conf /etc/nginx/nginx.conf
# Exponer el puerto HTTPS
EXPOSE 80 443

```

Con las siguientes lineas creo la imagen de nginx y el contendor

```

docker build -t mariorgzlpz-nginx-image:p3 -f P3-MarioRgzLpz-nginx/DockerFileNginxP3 .
docker run --name prueba-nginx -p 80:80 -p 443:443 -d mariorgzlpz-nginx-image:p3

```

Usando el docker compose de la P1, que me creaba los contenedores (sin un orden específico) y las redes, y el siguiente comando para conectar el contenedor nginx a la red web

```

docker network connect p1_red_web prueba-nginx

```

Me queda un resultado como el que muestro en el siguiente video [video drive](#)

B4.2 Analisis Propuesta IA

Para este apartado no he usado nada la IA porque siguiendo el guion no me ha dado ningún problema y me ha funcionado a la primera.

B5. DOCKER COMPOSE PARA LA GRANJA WEB CON SSL

B5.1. Propuesta final

En este apartado he modificado los DockerFile anteriores para que en lugar de copiar los certificados y las configuraciones montar volúmenes en el docker compose. Los DockerFiles me quedan así:

```

# Use an Alpine Linux base image
FROM alpine:3.15.0

# Install Apache, PHP, and networking tools
RUN apk add --no-cache \
    bash \
    php8-apache2 \
    php8 \
    apache2 \
    apache2-ssl

# Copy certificate and private key
# COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt /etc/apache2/ssl/
#     certificado_MarioRgzLpz.crt
# COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key /etc/apache2/ssl/
#     certificado_MarioRgzLpz.key

# Copy Apache configuration files
COPY /P3-MarioRgzLpz-apache/configs/apache/httpd.conf /etc/apache2/httpd.conf
# COPY /P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf /etc/apache2/conf.d/MarioRgzLpz
#     -apache-ssl.conf

# Enable SSL module
RUN sed -i '/LoadModule ssl_module/s/^#/g' /etc/apache2/httpd.conf

# Expose port 80 for Apache
EXPOSE 80
EXPOSE 443

# Command to start Apache in the foreground
CMD ["httpd", "-D", "FOREGROUND"]

#Imagen base
FROM nginx:latest

# Crear directorio para SSL
RUN mkdir -p /etc/nginx/ssl

# Copiar certificados SSL al contenedor
# COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt /etc/nginx/ssl/
#     certificado_MarioRgzLpz.crt
# COPY /P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key /etc/nginx/ssl/
#     certificado_MarioRgzLpz.key

# Incluir configuracion de Nginx para SSL
# COPY /P3-MarioRgzLpz-nginx/MarioRgzLpz-nginx-ssl.conf /etc/nginx/nginx.conf

# Exponer el puerto HTTPS
EXPOSE 80 443

```

El docker compose que he creado parte del docker compose de la P2 de nginx en el que añado los volúmenes para cada uno de los certificados, para las configuraciones y también le añado el puerto 443 al balanceador. Así me ha quedado:

```

version: '4.0'

services:
  web1:
    image: mariorgzlpz-apache-image:p3
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2

  web2:
    image: mariorgzlpz-apache-image:p3

```

```

volumes:
  - ./web_MarioRgzLpz:/var/www/localhost/htdocs
  - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/
MarioRgzLpz-apache-ssl.conf
  - ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
container_name: web2
networks:
  red_web:
    ipv4_address: 192.168.10.3
  red_servicios:
    ipv4_address: 192.168.20.3

web3:
  image: mariorgzlpz-apache-image:p3
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/
MarioRgzLpz-apache-ssl.conf
    - ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web3
  networks:
    red_web:
      ipv4_address: 192.168.10.4
    red_servicios:
      ipv4_address: 192.168.20.4

web4:
  image: mariorgzlpz-apache-image:p3
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/
MarioRgzLpz-apache-ssl.conf
    - ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web4
  networks:
    red_web:
      ipv4_address: 192.168.10.5
    red_servicios:
      ipv4_address: 192.168.20.5

web5:
  image: mariorgzlpz-apache-image:p3
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/
MarioRgzLpz-apache-ssl.conf
    - ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web5
  networks:
    red_web:
      ipv4_address: 192.168.10.6
    red_servicios:
      ipv4_address: 192.168.20.6

web6:
  image: mariorgzlpz-apache-image:p3
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/
MarioRgzLpz-apache-ssl.conf
    - ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
  container_name: web6
  networks:
    red_web:
      ipv4_address: 192.168.10.7
    red_servicios:
      ipv4_address: 192.168.20.7

web7:
  image: mariorgzlpz-apache-image:p3
  volumes:
    - ./web_MarioRgzLpz:/var/www/localhost/htdocs
    - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/

```

```

MarioRgzLpz-apache-ssl.conf
- ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
container.name: web7
networks:
  red_web:
    ipv4_address: 192.168.10.8
  red_servicios:
    ipv4_address: 192.168.20.8

web8:
  image: mariorgzlpz-apache-image:p3
  volumes:
    - ./web-MarioRgzLpz:/var/www/localhost/htdocs
    - ./P3-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/
MarioRgzLpz-apache-ssl.conf
- ./P3-MarioRgzLpz-certificados:/etc/apache2/ssl
container.name: web8
networks:
  red_web:
    ipv4_address: 192.168.10.9
  red_servicios:
    ipv4_address: 192.168.20.9

balanceador-nginx:
  image: mariorgzlpz-nginx-image:p3
  volumes:
    - ./P3-MarioRgzLpz-nginx/MarioRgzLpz-nginx-ssl.conf:/etc/nginx/nginx.conf
    - ./P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt:/etc/nginx/ssl/
certificado_MarioRgzLpz.crt
    - ./P3-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key:/etc/nginx/ssl/
certificado_MarioRgzLpz.key
container.name: balanceador-nginx
ports:
  - 80:80
  - 443:443
networks:
  red_web:
    ipv4_address: 192.168.10.50

networks:
  red_web:
    driver: bridge
  ipam:
    config:
      - subnet: 192.168.10.0/24
        gateway: 192.168.10.1
  red_servicios:
    driver: bridge
  ipam:
    config:
      - subnet: 192.168.20.0/24
        gateway: 192.168.20.1

```

B5.2 Analisis Propuesta IA

No he usado ChatGPT en este apartado porque el docker compose me ha funcionado correctamente sin ningún trabajo.

B6. VERIFICACIÓN Y PRUEBAS DEL ESCENARIO CON SSL

B6.1. Propuesta final

Aqui vemos como el docker compose funciona de manera correcta y lanza todos los contenedores web y el balanceador.

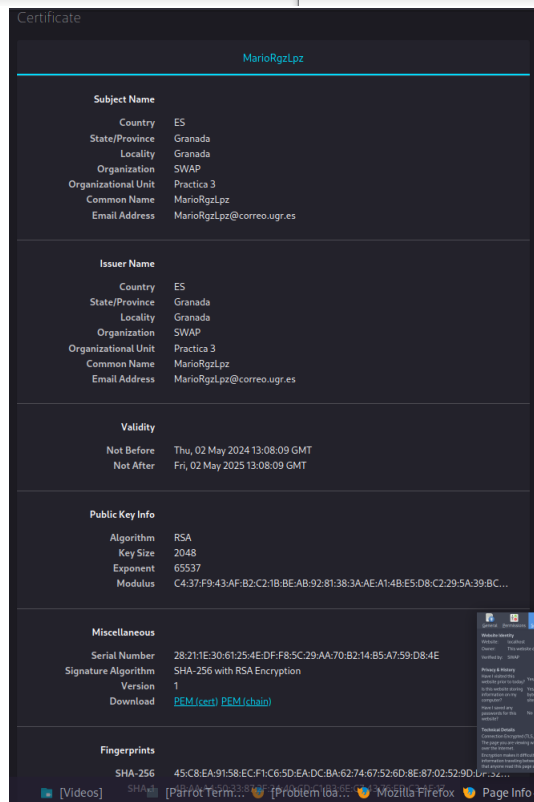
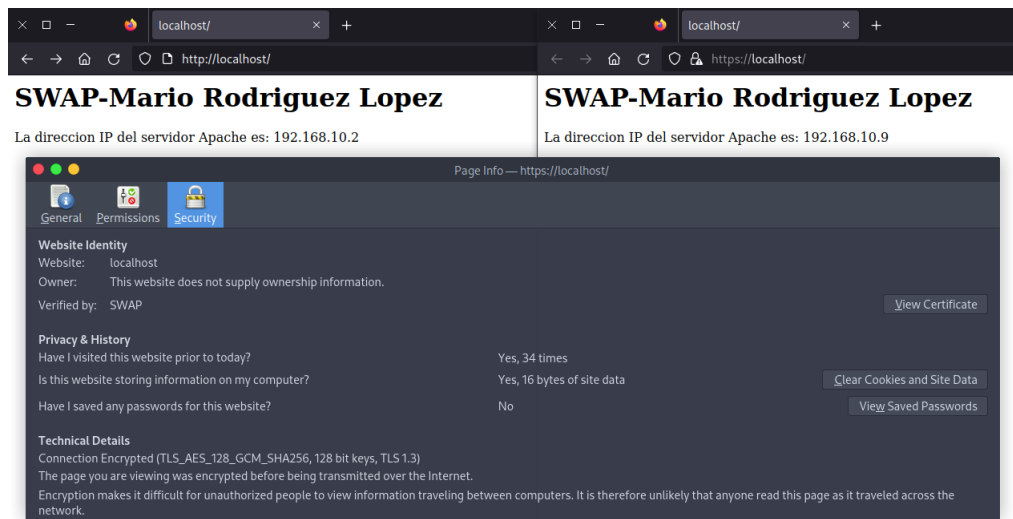

```

parrot: ~/.../SWAP/P3
- docker compose up -d
[+] Running 9/11
  ! Network p3_red_servicios Created 2.3s
  ! Network p3_red_web Created 2.2s
  ! Container web6 Started 2.1s
  ! Container balanceador-nginx Started 1.4s
  ! Container web3 Started 2.8s
  ! Container web1 Started 1.1s
  ! Container web4 Started 1.8s
  ! Container web2 Started 1.1s
  ! Container web8 Started 1.4s
  ! Container web5 Started 1.8s
  ! Container web7 Started 1.9s

parrot: ~/.../SWAP/P3
- docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
bce2c0fa91b6       mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 5 seconds       80/tcp, 443/tcp    web2
7cc0b34b32f5       mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 4 seconds       80/tcp, 443/tcp    web8
77c852b421f5       mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 4 seconds       80/tcp, 443/tcp    web4
084f4fcb91b4       mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 4 seconds       80/tcp, 443/tcp    web3
c3478cc0e27        mariorgzlpz-nginx-image:p3     "/docker-entrypoint..." 7 seconds ago       Up 4 seconds       0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp balanceador-nginx
38cfc2292ee7       mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 4 seconds       80/tcp, 443/tcp    web7
186111abc1e9       mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 5 seconds       80/tcp, 443/tcp    web1
ee239528b5a9       mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 4 seconds       80/tcp, 443/tcp    web5
c5e2aeb5a11        mariorgzlpz-apache-image:p3   "httpd -D FOREGROUND"   7 seconds ago       Up 4 seconds       80/tcp, 443/tcp    web6

```

En estas fotos podemos ver como esta configurado tanto http como https y el certificado.



Y en el siguiente video muestro el correcto funcionamiento del balanceador tanto en http como en https [video drive](#)

B6.2 Analisis Propuesta IA

No he usado ChatGPT en este apartado puesto que era solo realizar comprobaciones.

A1. CREACIÓN DE CERTIFICADOS SSL

A1.1. Propuesta final

Aqui muestro como he ido siguiendo los pasos que me ha dicho ChatGPT para ver que me genera de manera correcta tanto las claves privadas como los certificados de la CA y la CA intermedia que usaremos para firmar nuestro certificado en lugar de que sea autofirmado.

```
parrot: ~/.../P3/P3-MarioRgzLpz-certificados
→ openssl genrsa -out certificado_CA.key 2048

parrot: ~/.../P3/P3-MarioRgzLpz-certificados
→ openssl req -new -key certificado_CA.key -out certificado_CA.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CA SWAP
Organizational Unit Name (eg, section) []:Practica 3
Common Name (e.g. server FQDN or YOUR name) []:MarioRgzLpzCA
Email Address []:MarioRgzLpz@correo.ugr.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

parrot: ~/.../P3/P3-MarioRgzLpz-certificados
→ openssl x509 -req -in certificado_CA.csr -signkey certificado_CA.key -out certificado_CA.crt
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = CA SWAP, OU = Practica 3, CN = MarioRgzLpzCA, emailAddress = MarioRgzLpz@correo.ugr.es

parrot: ~/.../P3/P3-MarioRgzLpz-certificados
→ openssl genrsa -out certificado_subCA.key 2048

parrot: ~/.../P3/P3-MarioRgzLpz-certificados
→ openssl req -new -key certificado_subCA.key -out certificado_subCA.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:subCA SWAP
Organizational Unit Name (eg, section) []:Practica 3
Common Name (e.g. server FQDN or YOUR name) []:MarioRgzLpzsubCA
Email Address []:MarioRgzLpz@correo.ugr.es

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

parrot: ~/.../P3/P3-MarioRgzLpz-certificados
→ openssl x509 -req -in certificado_subCA.csr -CA certificado_CA.crt -CAkey certificado_CA.key -CAcreateserial
-out certificado_subCA.crt -days 365
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = subCA SWAP, OU = Practica 3, CN = MarioRgzLpzsubCA, emailAddress
= MarioRgzLpz@correo.ugr.es
```

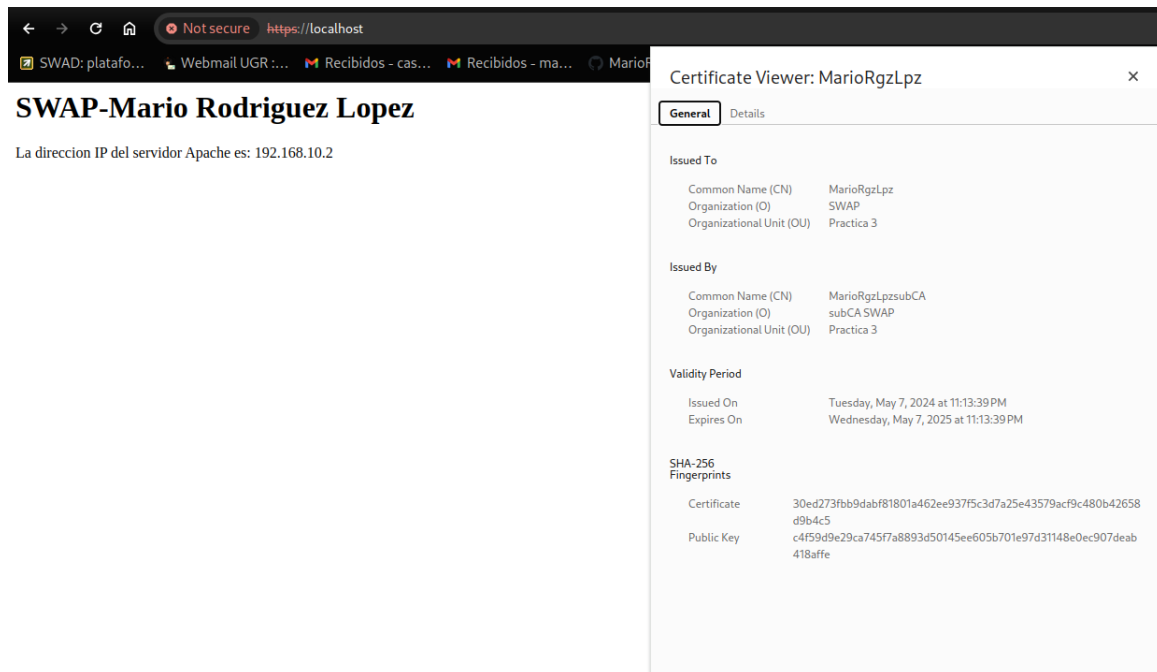
Ya con todo lo necesario solo hace falta crear la certificación de la B2 de nuevo usando la entidad certificado subCA en lugar de que sea autofirmado.

```

parrot: ~/.../P3-MarioRgzLpz-certificados
. openssl x509 -req -in certificado_subCA.csr -CA certificado_CA.crt -CAkey certificado_CA.key -CAcreateserial -out certificado_subCA.crt -days 365
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = subCA SWAP, OU = Practica 3, CN = MarioRgzLpzsubCA, emailAddress = MarioRgzLpz@correo.ugr.es
. openssl x509 -req -in certificado_MarioRgzLpz.csr -CA certificado_subCA.crt -CAkey certificado_subCA.key -CAcreateserial -out certificado_MarioRgzLpz.crt -days 365
Certificate request self-signature ok
subject=C = ES, ST = Granada, L = Granada, O = SWAP, OU = Practica 3, CN = MarioRgzLpz, emailAddress = MarioRgzLpz@correo.ugr.es

```

Y comprobamos que se vea el nuevo certificado en la dirección https del localhost.



Aquí vemos como la certificación del servidor está firmada por nuestra CA intermedia por lo que ya tendríamos nuestro certificado del servidor firmado por una entidad certificadora simulando una situación real.

A1.2 Analisis Propuesta IA

Pregunta a ChatGPT Generar un certificado raíz (CA) y uno o más certificados intermedios (subCA) para entender cómo se construye una cadena de confianza. Utilizar el certificado intermedio para firmar los certificados de los servidores, simulando una estructura más realista y segura que se encuentra en entornos de producción.

Recomendaciones: Primero, genera clave privada de la CA y úsala para crear un certificado autofirmado que servirá como CA raíz. Segundo, crea una Autoridad Certificadora Intermedia (subCA) similar a la CA, crea una clave privada para la subCA y utilízala para generar una solicitud de certificado (CSR) de la subCA que será firmada por la CA raíz, estableciendo una cadena de confianza.

Respuesta ChatGPT

Aquí tienes un proceso paso a paso para generar un certificado raíz (CA) y uno o más certificados intermedios (subCA) utilizando OpenSSL:

Paso 1: Generar clave privada y certificado autofirmado para la CA raíz Genera la clave privada para la CA raíz:

```
openssl genrsa -out ca.key 2048
```

Genera la solicitud de certificado (CSR) para la CA raíz:

```
openssl req -new -key ca.key -out ca.csr
```

Firma el certificado con la propia clave privada para crear el certificado autofirmado de la CA raíz

```
openssl x509 -req -in ca.csr -signkey ca.key -out ca.crt
```

Paso 2: Generar clave privada y CSR para la subCA Genera la clave privada para la subCA:

```
openssl genrsa -out subca.key 2048
```

Genera la solicitud de certificado (CSR) para la subCA:

```
openssl req -new -key subca.key -out subca.csr
```

Paso 3: Firmar la CSR de la subCA con la clave privada de la CA raíz Firma la CSR de la subCA con la clave privada de la CA raíz para establecer la cadena de confianza:

```
openssl x509 -req -in subca.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out subca.crt -days 365
```

Ahora tienes una CA raíz y una subCA con una cadena de confianza establecida entre ellas. Puedes usar la subCA para firmar certificados para servidores individuales.

Análisis Lo que me proporciona ChatGPT es correcto y solo sería necesario repetir el último paso con la solicitud de firma creada en el apartado B2 para crear el certificado que usamos en nuestro servidor.

A2. OPTIMIZACIÓN DE CONFIGURACIÓN SSL EN LOS SERVIDORES WEB

A2.1. Propuesta final

Confirmando lo que me dice ChatGPT veo que también son inseguros los protocolos SSL 2.0 y SSL 3.0 además de que el protocolo TLS v1.3 que es del 2018 no lo menciona a pesar de ser el más seguro, por lo que yo he optado por deshabilitar todos los protocolos y permitir solo TLS v1.2 y TLS v1.3 que son los más seguros. La información la he extraído de varias fuentes que nombro aquí, [wikipedia para ver la seguridad de los protocolos](#), [tutorial para activar solo los protocolos TLS v1.2 y TLS v1.3](#)

Además para el cifrado también he comprobado que los cifrados MD5, RC4 y 3DES no son seguros y que por lo tanto es mejor no permitirlos. Pero basta con usar la opción HIGH con unas directivas para lograr que solo se permitan cifrados con un alto grado de protección. Para esto he consultado las siguientes web [documentación de apache](#), [tutorial para desactivar versiones inseguras y cifrados inseguros](#)

Por tanto el fichero de configuración tomando en cuenta las consideraciones anteriores me quedaría como sigue:

```
<VirtualHost *:443>
    DocumentRoot /var/www/localhost/htdocs
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/certificado_MarioRgzLpz.crt
    SSLCertificateKeyFile /etc/apache2/ssl/certificado_MarioRgzLpz.key

    # Configuración de protocolos SSL/TLS permitidos
    SSLProtocol -all +TLSv1.2 +TLSv1.3

    # Configuración de cifrados SSL permitidos
    SSLCipherSuite HIGH:!aNULL:!MD5:!3DES:!RC4

</VirtualHost>
```

Comprobamos que el fichero se está cargando en nuestros contenedores apache simplemente usando el editor vi con un docker exec. Usando el siguiente comando:

```
docker exec -it web1 vi /etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
```

Y como vemos la configuración está cambiada:

```
VirtualHost *:443>
DocumentRoot /var/www/localhost/htdocs
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/certificado_MarioRgzLpz.crt
SSLCertificateKeyFile /etc/apache2/ssl/certificado_MarioRgzLpz.key

# Configuración de protocolos SSL/TLS permitidos
SSLProtocol -all +TLSv1.2 +TLSv1.3

# Configuración de cifrados SSL permitidos
SSLCipherSuite HIGH:!aNULL:!MD5:!3DES:!RC4

</VirtualHost>
~
~
~
~
- /etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf 1/13 7%
```

A2.2 Analisis Propuesta IA

Pregunta a ChatGPT Pregunta 1 Optimizar la configuración SSL en Apache para mejorar tanto la seguridad como el rendimiento de las conexiones seguras. Para ello, deshabilitar protocolos inseguros y cifrados débiles que pueden ser explotados por ataques. Por ejemplo, permitir o denegar protocolo TLS v1, TLS v1.1 y TLS v1.2 así como cifrados MD5, RC4 y 3DES. Justifica la elección de cada uno.

Pregunta 2 Como puedo implementar esto en mi configuracion teniendo en cuenta que mi fichero es el siguiente (le paso el fichero de configuracion de apache de la B2)

Respuesta ChatGPT Respuesta 1 Para optimizar la configuración SSL en Apache y mejorar tanto la seguridad como el rendimiento de las conexiones seguras, es importante deshabilitar protocolos inseguros y cifrados débiles que pueden ser explotados por ataques. Aquí hay algunas recomendaciones y justificaciones para cada una:

Protocolos SSL/TLS TLS v1: Se debe deshabilitar, ya que es vulnerable a varios ataques y ha sido superado por versiones posteriores de TLS. TLS v1 no proporciona las últimas mejoras de seguridad y puede exponer al servidor a riesgos. TLS v1.1: También se debe deshabilitar, ya que ha sido superado por versiones más recientes de TLS. TLS v1.1 tiene algunas vulnerabilidades conocidas y no proporciona la seguridad óptima ofrecida por versiones posteriores. TLS v1.2: Se debe permitir, ya que es una versión sólida y ampliamente aceptada de TLS que proporciona una buena seguridad y compatibilidad con los clientes. TLS v1.2 es ampliamente adoptado y ofrece mejoras significativas sobre versiones anteriores. **Cifrados débiles MD5:** Debe ser deshabilitado, ya que es vulnerable a ataques de colisión y se considera débil y no seguro para su uso en conexiones SSL/TLS. **RC4:** Debe ser deshabilitado, ya que ha sido comprometido y se considera inseguro. RC4 ha sido objeto de varios ataques y ya no se recomienda para su uso en conexiones seguras. **3DES:** Debe ser deshabilitado, ya que es vulnerable a ataques de fuerza bruta y se considera obsoleto. 3DES tiene problemas de seguridad conocidos y ha sido reemplazado por algoritmos de cifrado más seguros. **Justificación Seguridad:** Al deshabilitar protocolos inseguros y cifrados débiles, se reduce el riesgo de que los ataques exploren vulnerabilidades conocidas y comprometan la seguridad del servidor y de los datos transmitidos. **Compatibilidad:** Permitir sólo protocolos y cifrados seguros garantiza una mejor compatibilidad con los clientes modernos que implementan las últimas versiones de TLS y los algoritmos de cifrado más seguros. Los clientes más antiguos que no sean compatibles con estas configuraciones deberían ser actualizados o deshabilitados para garantizar la seguridad. **Rendimiento:** Al eliminar protocolos y cifrados innecesarios, se puede mejorar el rendimiento del servidor al reducir la sobrecarga asociada con la negociación de conexiones SSL/TLS y el procesamiento de cifrados débiles. Al implementar estas recomendaciones, se puede lograr un equilibrio óptimo entre seguridad y rendimiento en las conexiones seguras gestionadas por Apache.

Respuesta 2 Para aplicar las recomendaciones de seguridad en tu configuración de Apache, necesitas agregar las directivas SSLProtocol y SSLCipherSuite dentro de tu bloque VirtualHost correspondiente al puerto 443. Aquí te muestro cómo puedes hacerlo:

```
<VirtualHost *:443>
    DocumentRoot /var/www/localhost/htdocs
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/certificado_MarioRgzLpz.crt
    SSLCertificateKeyFile /etc/apache2/ssl/certificado_MarioRgzLpz.key

    # Configuracion de protocolos SSL/TLS permitidos
    SSLProtocol all -TLSv1 -TLSv1.1

    # Configuracion de cifrados SSL permitidos
    SSLCipherSuite HIGH:!MD5:!RC4:!3DES
</VirtualHost>
```

En este ejemplo, he agregado las directivas SSLProtocol y SSLCipherSuite dentro del bloque VirtualHost para el puerto 443. La directiva SSLProtocol especifica qué protocolos SSL/TLS están permitidos, y la directiva SSLCipherSuite especifica los cifrados que se permiten y se excluyen.

Con estas configuraciones, Apache solo permitirá conexiones SSL/TLS utilizando protocolos seguros (todos excepto TLSv1 y TLSv1.1) y cifrados fuertes (sin MD5, RC4 y 3DES), lo que mejorará tanto la seguridad como el rendimiento de las conexiones seguras en este virtual host específico. Asegúrate de guardar los cambios en tu archivo de configuración y luego reinicia Apache para que los cambios surtan efecto.

Analisis En este pregunta ChatGPT es bastante fiable puesto que nos dice cuales son seguros y cuales no con precisión, información que yo he contrastado consultando wikipedia y además la forma en la que propone la implementación es correcta por lo que bastaría con ajustarlo a los protocolos y mecanismos de cifrado que yo he elegido para que todo sea correcto.

A3. CONFIGURACIÓN DE CACHE DE SESIONES SSL Y TICKETS DE SESION EN EL BALANCEADOR

A3.1. Propuesta final

Siguiendo lo que hace me dice ChatGPT modifíco la configuración de nginx añadiendo las directivas para cache de sesion SSL y los tickets de sesión además de crear el ticket de sesion como sigue:

```
parrot: ~/.../SWAP/P3
→ openssl rand 80 > ticket_sesion_MarioRgzLpz.key
```

También me creo un volumen para cargar el ticket de sesion en el docker compose:

```
- ./ticket_sesion_MarioRgzLpz.key:/etc/nginx/ssl/session_ticket.key
```

Y me queda la siguiente configuración de nginx:

```
events {
    worker_connections 1024;
}

http {
    # Configuración de la cache de sesiones SSL y tickets de sesion
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    upstream backend_MarioRgzLpz {
        server 192.168.10.2 weight=1;
        server 192.168.10.3 weight=1;
        server 192.168.10.4 weight=1;
        server 192.168.10.5 weight=1;
        server 192.168.10.6 weight=1;
        server 192.168.10.7 weight=1;
        server 192.168.10.8 weight=1;
        server 192.168.10.9 weight=1;
    }

    server {
        listen 80;
        server_name nginx_MarioRgzLpz;

        access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
        error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

        location / {
            proxy_pass http://backend_MarioRgzLpz;
            proxy_set_header Cookie $http_cookie;
            proxy_hide_header Set-Cookie;
        }

        location /estadisticas_MarioRgzLpz {
            stub_status on;
        }
    }

    server {
        listen 443 ssl;
        server_name nginx_MarioRgzLpz;

        # Activamos los tickets de sesion y cargamos nuestro ticket
        ssl_session_tickets on;
        ssl_session_ticket_key /etc/nginx/ssl/session_ticket.key;

        ssl_certificate /etc/nginx/ssl/certificado_MarioRgzLpz.crt;
        ssl_certificate_key /etc/nginx/ssl/certificado_MarioRgzLpz.key;

        access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
        error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

        location / {
```

```

        proxy_pass http://backend_MarioRgzLpz;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
    }

    location /estadisticas_MarioRgzLpz {
        stub_status on;
    }
}

```

Y podemos comprobar si el ticket de sesión está en la web o no usando

A3.2 Analisis Propuesta IA

Pregunta a ChatGPT Configurar Nginx para utilizar caché de sesiones SSL y tickets de sesión para mejorar la velocidad de las conexiones seguras repetidas, reduciendo el tiempo necesario para negociar la seguridad de la conexión.

Respuesta ChatGPT Para configurar Nginx para utilizar caché de sesiones SSL y tickets de sesión, puedes seguir estos pasos:

Habilitar el caché de sesiones SSL: El caché de sesiones SSL permite almacenar información de la sesión SSL en la memoria del servidor para reutilizarla en conexiones SSL posteriores del mismo cliente. Esto reduce la sobrecarga asociada con la negociación de la seguridad de la conexión en cada solicitud. Agrega la siguiente directiva dentro de tu bloque de servidor o en el bloque de servidor predeterminado en el archivo de configuración de Nginx (nginx.conf o un archivo de configuración específico):

```

ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;

```

ssl_session_cache shared:SSL 10m; Esta directiva especifica el tipo y el tamaño de la caché de sesiones SSL. En este ejemplo, se usa una caché compartida llamada "SSL" con un tamaño de 10 megabytes. Puedes ajustar el tamaño según tus necesidades. ssl_session_timeout 10m;: Esta directiva establece el tiempo de vida de las entradas en la caché de sesiones SSL. En este caso, se configura en 10 minutos. **Habilitar los tickets de sesión SSL:** Los tickets de sesión SSL permiten al servidor cifrar y almacenar información de sesión SSL en un "ticket" que se envía al cliente. El cliente puede enviar de vuelta este ticket en futuras solicitudes SSL para reanudar la sesión SSL sin necesidad de una nueva negociación completa. Agrega la siguiente directiva dentro del mismo bloque de servidor o en el bloque de servidor predeterminado:

```

ssl_session_tickets on;
ssl_session_ticket_key /etc/nginx/ssl/session_ticket.key;

```

ssl_session_tickets on;: Esta directiva habilita los tickets de sesión SSL. ssl_session_ticket_key /etc/nginx/ssl/session_ticket.key;: Esta directiva especifica la ubicación del archivo de clave que se utilizará para cifrar y descifrar los tickets de sesión SSL. Debes generar y configurar esta clave para habilitar los tickets de sesión SSL. Puedes generar la clave con el siguiente comando:

```

openssl rand 80 > /etc/nginx/ssl/session_ticket.key

```

Reinicia Nginx para que los cambios surtan efecto:

```

sudo systemctl restart nginx

```

Con estas configuraciones, Nginx utilizará la caché de sesiones SSL y los tickets de sesión SSL para mejorar la velocidad de las conexiones seguras repetidas al reducir el tiempo necesario para negociar la seguridad de la conexión en cada solicitud SSL.

Analisis Las indicaciones que me da ChatGPT son las necesarias para configurar caché de sesiones ssl y tickets de sesión, información que he contrastado con la documentación oficial de nginx. [documentacion nginx](#)

A4. OPTIMIZACIÓN DE CONEXIONES HTTPS Y CIFRADO EN EL BALANCEADOR

A4.1. Propuesta final

Las elecciones que hace ChatGPT con respecto a los protocolos son las mismas que las que habia hecho para la configuración de apache, suceso totalmente lógico ya que los protocolos más seguros y mejor optimizados van a ser los más recientes como es el caso de TLS1.3. Por esto en mi configuración de nginx he optado por permitir TLS1.3 y también TLS1.2 por compatibilidad ya que la nueva versión es bastante reciente(2018). Para las suites de cifrado vamos a usar casi siempre el algoritmo de intercambio de claves mediante curvas elípticas, ya demostrado como uno de los más eficientes y seguros a la vez y que mejora el intercambio estandar de claves Diffie-Helman. En conjunto con este algoritmo usaremos como algoritmo de cifrado AES (Advanced Encryption Standard) y una de sus variantes que es más eficiente e igual de segura. Toda la información relacionada con la elección de las suites ha sido extraída de varias fuentes (wikipedia, foros y apuntes de criptografía). Todo esto lo permitimos mediante las dos directivas `ssl_protocols` y `ssl_ciphers` de manera que el archivo de configuración me queda así:

```
events {
    worker_connections 1024;
}

http {
    # Configuración de la cache de sesiones SSL y tickets de sesion
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    upstream backend_MarioRgzLpz {
        server 192.168.10.2 weight=1;
        server 192.168.10.3 weight=1;
        server 192.168.10.4 weight=1;
        server 192.168.10.5 weight=1;
        server 192.168.10.6 weight=1;
        server 192.168.10.7 weight=1;
        server 192.168.10.8 weight=1;
        server 192.168.10.9 weight=1;
    }

    server {
        listen 80;
        server_name nginx_MarioRgzLpz;

        access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
        error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

        location / {
            proxy_pass http://backend_MarioRgzLpz;
            proxy_set_header Cookie $http_cookie;
            proxy_hide_header Set-Cookie;
        }

        location /estadisticas_MarioRgzLpz {
            stub_status on;
        }
    }

    server {
        listen 443 ssl;
        server_name nginx_MarioRgzLpz;

        http2 on;

        # Activamos los tickets de sesion y cargamos nuestro ticket
        ssl_session_tickets on;
        ssl_session_ticket_key /etc/nginx/ssl/session_ticket.key;

        # Configuración de parametros SSL adicionales
        ssl_protocols TLSv1.2 TLSv1.3; #Elección de protocolos SSL
        ssl_prefer_server_ciphers on; # Preferencia de la lista de cifrados del servidor
        sobre la del cliente
    }
}
```



```

    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH'; # Lista de suites
de cifrado

    ssl_certificate /etc/nginx/ssl/certificado_MarioRgzLpz.crt;
    ssl_certificate_key /etc/nginx/ssl/certificado_MarioRgzLpz.key;

    access_log /var/log/nginx/nginx_MarioRgzLpz.access_log;
    error_log /var/log/nginx/nginx_MarioRgzLpz.error_log;

    location / {
        proxy_pass http://backend_MarioRgzLpz;
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
    }

    location /estadisticas_MarioRgzLpz {
        stub_status on;
    }
}

```

A4.2 Analisis Propuesta IA

Pregunta a ChatGPT Personalizar los protocolos SSL/TLS y suites de cifrado para equilibrar seguridad y rendimiento y activar HTTP/2 para mejorar la eficiencia de las conexiones HTTPS. Por ejemplo, protocolos TLSv1.2 y TLSv1.3 así como cifrado EECDH, AESGCM, AES256, etc. Justifica la elección de cada uno.

Respuesta ChatGPT Para personalizar los protocolos SSL/TLS y las suites de cifrado en Nginx con el objetivo de equilibrar seguridad y rendimiento, así como activar HTTP/2 para mejorar la eficiencia de las conexiones HTTPS, aquí tienes una configuración recomendada junto con la justificación de cada elección:

Protocolos SSL/TLS: TLSv1.2 y TLSv1.3: Estos protocolos son versiones más modernas y seguras de SSL/TLS. TLSv1.3 es especialmente deseable debido a sus mejoras en seguridad y rendimiento en comparación con las versiones anteriores. Sin embargo, es importante mantener la compatibilidad con TLSv1.2 para asegurar que los clientes más antiguos aún puedan conectarse.

```
ssl_protocols TLSv1.2 TLSv1.3;
```

Suites de cifrado: EECDH (Elliptic Curve Ephemeral Diffie-Hellman): Los algoritmos de intercambio de claves de curva elíptica, como EECDH, ofrecen un mejor rendimiento y seguridad en comparación con los métodos más antiguos como DHE. AESGCM (Advanced Encryption Standard Galois/Counter Mode): AESGCM es un modo de cifrado de bloques que proporciona autenticación de datos y confidencialidad de manera eficiente, siendo una excelente opción para aplicaciones HTTPS. AES256: Utilizar AES con claves de 256 bits proporciona un alto nivel de seguridad y es ampliamente compatible con la mayoría de los navegadores modernos.

```
ssl_prefer_server_ciphers on;
ssl_ciphers EECDH+AESGCM:EECDH+AES256:EECDH+AES128:EDH+AES128;
```

Activar HTTP/2: HTTP/2 es la versión más reciente del protocolo HTTP y proporciona mejoras significativas en la eficiencia de la transferencia de datos, lo que resulta en un rendimiento mejorado para las conexiones HTTPS.

```
listen 443 ssl http2;
```

Analisis Lo que me proporciona ChatGPT es una configuración válida en la que obtenemos una optimización de las conexiones gracias a usar TLS 1.3 (aunque permitimos TLS 1.2 para tener mayor compatibilidad) y empleamos los cifrados más seguros y mejor optimizados y otros que ofrecen alta compatibilidad. Lo único que modifiqué es la manera en la que activamos http2 ya que la forma que me dice ChatGPT ya está deprecated, simplemente mirando la documentación oficial de nginx me aparece como se activa.