MEDAC
Instituto Oficial de Formación Profesional

DESPLIEGUE DE APLICACIONES WEB
TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Administración de servidores web

03

ÍNDICE

OBJETIVOS

Conocer el protocolo HTTPS

Aplicar configuraciones avanzadas en un servidor web

Descubrir los módulos

Analizar el rendimiento de un servidor web

Optimizar un servidor web

/ 1. Introducción y contextualización práctica

En este tema nos centraremos en la administración de servidores web. Analizaremos los diferentes ficheros de configuración necesarios para aplicar cambios y añadiremos funcionalidades utilizando módulos.

Haremos pruebas de rendimiento valorando las diferentes configuraciones y securizaremos las conexiones entre el cliente y el servidor utilizando el protocolo HTTPS.

Por último, instalaremos aplicaciones web para poner en práctica todo lo aprendido.

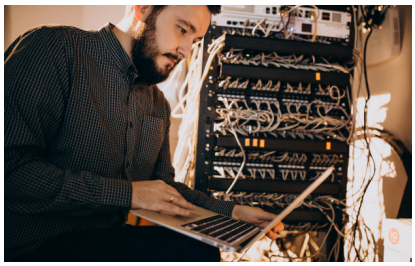

Fig. 1. Administración de servidores web

Planteamiento del caso práctico inicial

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado “Resumen y Resolución del caso práctico”.

Audio Intro. “Problemas con la subida de archivos”
https://bit.ly/32kIVZy

/ 2. Configuraciones necesarias y seguridad

Detallaremos a continuación las configuraciones avanzadas de aplicación a servidores web, además de las consideraciones sobre seguridad a tener en cuenta.

2.1. El protocolo HTTPS

El protocolo HTTPS permite establecer comunicaciones seguras entre cliente y servidor utilizando el puerto TCP 443. De esta manera, si alguien estuviese capturando paquetes en la red, no podría obtener información sensible en texto plano. Para ello, HTTPS cifra los datos utilizando el protocolo seguro SSL (Secure Socker Layer) utilizado por ejemplo por la aplicación SSH (Secure Shell) para tener conexiones remotas encriptadas.

No obstante, a cambio de tener una mayor seguridad, utilizar HTTPS tiene algunos puntos negativos:

- Necesita un **certificado válido firmado por una CA** (Certification Autority), lo que normalmente conlleva un coste. Aunque existen soluciones gratuitas, como por ejemplo certbot.
- Posibilidad de un **menor rendimiento** causado por la cantidad de procesos que ejecutará el servidor para cifrar los paquetes.
- El **tiempo de respuesta** podría aumentar debido al rendimiento del servidor.

Audio 1. "Riesgos de usar HTTP"
https://bit.ly/3ewAg8w

2.2. Configuración de PHP

Cuando administramos un servidor web es necesario conocer cómo modificar valores del componente PHP para optimizar su funcionamiento, añadir extensiones o depurar errores.

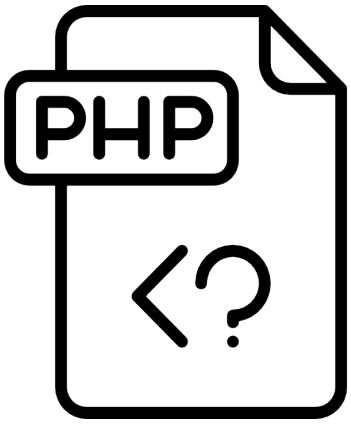

Fig. 2. PHP

En el fichero **php.ini** se define la configuración de PHP. Entre sus variables, encontramos las siguientes:

- **memory_limit:** establece el límite de memoria que un script podrá consumir. Se mide en Megabytes.
- **upload_max_filesize:** limita el tamaño máximo permitido para subir ficheros al servidor. Se mide en Megabytes.
- **post_max_size:** limita el tamaño máximo permitido para peticiones POST.

Teniendo en cuenta los puntos anteriores, la modificación de estos dos últimos parámetros debe hacerse de manera consecuente, y POST_MAX_SIZE debe ser mayor o igual que UPLOAD_MAX_FILESIZE.

Encontrarás más información en la web oficial de PHP:

https://www.php.net/manual/es/ini.core.php

2.3. Configuración MySQL

MySQL almacena los archivos de configuración en la carpeta /etc/mysql, que contiene el fichero mysql.cfg. Sus parámetros más importantes son los siguientes:

- **Bind-address:** establece la dirección desde la cual se podrá acceder a MySQL. El valor por defecto es 127.0.0.1 (localhost). Modificando este valor, se permitiría acceder a MySQL desde otro host. Una mala configuración podría provocar brechas en la seguridad del servidor.
- **Key_buffer_size:** permite modificar el tamaño del buffer.
- **Max_connections:** permite modificar el número de conexiones simultaneas autorizadas.

En el siguiente enlace encontrarás un script en Python muy útil para detectar problemas de configuración o de seguridad en MySQL.

Fig. 3. Enlace a GitHub de MySQL Tunner

2.4. Configuración Nginx

Nginx gestiona los sitios utilizando la estructura de sites-available y sites-enabled y como vimos en el tema anterior, para cada web que alojemos en el servidor, debemos activar PHP de forma explícita en la configuración del sitio.

La configuración general del servidor web se puede modificar editando el fichero nginx.conf que se encuentra en /etc/nginx/.

En este archivo podremos realizar ajustes para mejorar el rendimiento, editar la versión de TLS que queremos utilizar o establecer las rutas de los logs.

Algunas de las variables que podemos modificar son:

- **worker_processes:** este parámetro define la cantidad de worker o hilos del procesador que utilizará Nginx para gestionar peticiones. Por defecto, se configurará de forma automática y tiene relación de un worker por cada core que tenga la CPU.
- **worker_connections:** este valor define la cantidad de peticiones que un worker puede gestionar simultáneamente.

Nginx también almacena información de los sucesos que se producen y los guarda en dos ficheros access.log y error.log. Por defecto, los logs de Nginx se almacenan en /var/log/nginx.

```
usuario@lnxsrv-web-01:~$ cat /var/log/nginx/access.log
192.168.1.26 - - [25/Jun/2020:16:20:40 +0000] "GET / HTTP/1.1" 200 396 "-" "Mozilla/5
.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4
103.116 Safari/537.36"
192.168.1.26 - - [25/Jun/2020:16:20:40 +0000] "GET /favicon.ico HTTP/1.1" 404 209 "ht
tp://192.168.1.117/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (K
```

Fig. 4. Log de acceso de Nginx

En el siguiente enlace encontrarás información para optimizar el rendimiento de un servidor web Nginx.

https://docs.bluehosting.cl/tutoriales/servidores/como-configurar-nginx-para-obtener-un-rendimiento-optimo.html

2.5. Configuración HTTPS en Nginx

A la hora de configurar sitios web seguros con HTTPS necesitaremos certificados digitales para cifrar la conexión. Por defecto los servidores webs utilizan unos certificados autofirmados genéricos llamados snake-oil, pero podremos generar otros con OpenSSL o adquirir certificados válidos a través de una Autoridad de Certificación (CA) que los genere y certifique.

En caso de querer activar SSL en Nginx utilizando certificados personalizados, el primer paso será generarlos con open SS.

```
usuario@lnxsrv-web-01:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/
private/certnginx.key -out /etc/ssl/certs/certnginx.crt
```

Fig. 5. Generación de certificados SSL

A continuación, crearemos un nuevo fichero de configuración en la carpeta sites-available con el siguiente contenido:

```
server {
        # SSL configuration
        listen 443 ssl;
        server_name sslNginx.local;

        ssl on;
        ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
        ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;

        root /var/www/html;

       # Add index.php to the list if you are using PHP
        index index.html index.htm index.nginx-debian.html;
}
```

Fig. 6. Configuración Virtual Block SSL Nginx

Una vez creado el Virtual Block procederemos a su activación creando un enlace simbólico en la carpeta sites-enabled. Comprobaremos posteriormente la configuración de Nginx, y reiniciaremos el servicio.

Al acceder al sitio utilizando el protocolo HTTPS, comprobaremos que el navegador lo marca como "sitio no seguro". Esto es debido a que hemos utilizado un certificado autofirmado.

```
usuario@lnxsrv-web-01:~$ sudo ln -s /etc/nginx/sites-available/sslNginx /etc/nginx/sites-enabled/.
usuario@lnxsrv-web-01:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
usuario@lnxsrv-web-01:~$ sudo service nginx restart
```

Fig. 7. Activación de sitio y recarga del servicio Nginx

En los siguientes enlaces encontrarás la documentación oficial de Nginx con consejos para activar SSL y una guía para configurar HTTPS.

- **Consejos:** https://nginx.org/en/docs/http/ngx_http_ssl_module.html
- **Guía:** https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-18-04

2.6. Configuración de Apache

El servidor web Apache organiza los sitios webs alojados con un sistema llamado virtualHost, del que hablaremos en profundidad en próximos temas. Los virtualHost se almacenan en un fichero con toda la información referente al sitio web que estamos publicando (Fig. 11).

De la misma manera Apache tiene otros ficheros de configuración para gestionar las propias características del servidor. Dichos ficheros de configuración se almacenan en la carpeta /etc/apache, entre los que destacan:

- **Apache2.conf:** en este fichero se definen opciones generales y de rendimiento.
- **Mods-available y mods-enabled:** en la carpeta mods-available se almacenan todos los módulos de PHP instalados (módulos disponibles activados o no). Al activar un módulo se realizaría un enlace en la carpeta mods-enabled (módulos activados).
- **Ports.conf:** fichero de configuración de los puertos utilizados por Apache para atender peticiones.
- **Sites-available y sites-enabled:** en el directorio sites-available se almacenan los ficheros de configuración de los sitios webs sin activar, así como en el directorio sites-enabled se almacenan los activados.

```
usuario@lnxsrv-web-01:~$ cat /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Fig. 8. Ejemplo de un VirtualHost de Apache

En caso de necesitar investigar un error o un incidente que se haya producido en el servidor, podemos revisar los logs (registros de eventos) del servicio en cuestión.

Apache guarda un registro de los sucesos que tienen lugar. De los ficheros que encontraremos en la ruta /var/log/Apache2 los más importantes son:

- **Access.log:** almacena información de los clientes que han solicitado recursos al servidor web.
- **Error.log:** almacena información de los errores que se producen en el servidor web.

```
usuario@lnxsrv-web-01:~$ cat /var/log/apache2/error.log
[Thu Jun 25 15:31:18.749613 2020] [mpm_event:notice] [pid 7190:tid 140403180985280] AH00489: Apache/2
.4.29 (Ubuntu) configured -- resuming normal operations
[Thu Jun 25 15:31:18.749804 2020] [core:notice] [pid 7190:tid 140403180985280] AH00094: Command line:
 '/usr/sbin/apache2'
[Thu Jun 25 15:31:47.386892 2020] [mpm_event:notice] [pid 7190:tid 140403180985280] AH00491: caught S
IGTERM, shutting down
```

Fig. 9. Log de errores de Apache

2.7. Configuración HTTPS en Apache

La configuración sobre Apache se realiza modificando el virtualHost de nuestro sitio web para añadir las siguientes opciones:

```
SSLEngine on

SSLCertificateFile      /ruta/carpeta/certificado/certicado.pem
SSLCertificateKeyFile           /ruta/carpeta/certificado/certicado.key
```

Código 1. Verificación HTTPs con certificado inválido

Por defecto, Apache viene con un sitio desactivado que utiliza HTTPS, llamado default-ssl.conf. Para activarlo es suficiente con realizar un enlace simbólico del archivo a la carpeta sites-enabled.

```
usuario@lnxsrv-web-01:~$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificat
es.
To activate the new configuration, you need to run:
  systemctl restart apache2
usuario@lnxsrv-web-01:~$ sudo ln -s /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-
enabled/.
usuario@lnxsrv-web-01:~$ sudo apache2ctl restart
usuario@lnxsrv-web-01:~$
```

Fig. 10. Activar virtualHost SSL por defecto de Apache

Cuando reiniciemos el servicio, podremos entrar a la IP de nuestro servidor mediante HTTPS. Utilizando las "opciones de desarrollo de Google Chrome", bajo el apartado "Security", podremos comprobar que nuestro sitio web es seguro, pero no tiene un certificado válido.

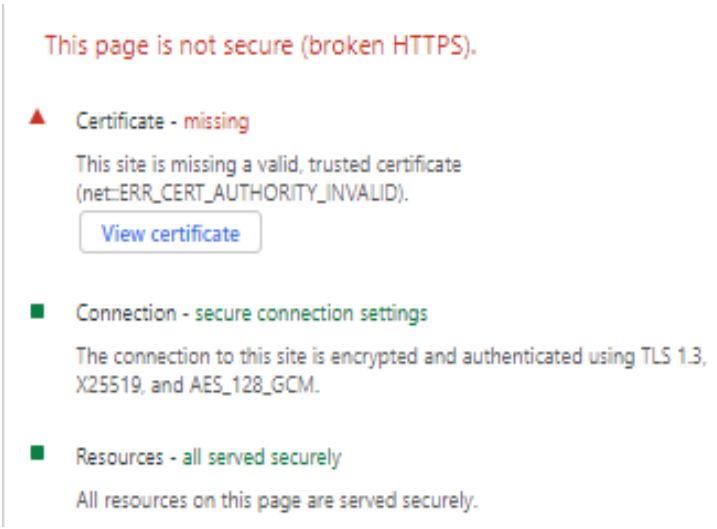

Fig. 11.Verificación HTTPs con certificado inválido

/ 3. Caso práctico 1: "Activar SSL Nginx"

Planteamiento: Pedro trabaja en una empresa de hosting y desarrollo web. Ha preparado un VPS en la nube con Nginx y quiere activar HTTPS para configurar un servidor seguro.

El objetivo es poder ofrecer a aquellos clientes que llevan años sin actualizar sus páginas webs una demostración sobre las ventajas de utilizar sitios seguros para que así, algunos de ellos se decidan a contratar sus servicios.

Nudo: ¿Qué pasos debe seguir para configurar HTTPS? ¿Necesitará crear certificados autofirmados?

Desenlace: El primer paso será instalar el servidor web con el comando apt-get install nginx.

A continuación, comprobaremos que podemos acceder desde un navegador web a la URL de nuestro servidor.

El siguiente paso será crear el certificado, para ello creamos una carpeta para almacenarlo y otorgamos permisos al usuario propietario. Como es un servidor de desarrollo, no es necesario que este certificado deba estar firmado por una CA.

Utilizaremos OpenSSL para crear dos certificados que utilizaremos posteriormente en Nginx.

```
usuario@lnxsrv-web-01:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048
-keyout /etc/ssl/private/certnginx.key -out /etc/ssl/certs/certnginx.crt
```

Fig. 12. Generación de certificados SSL con OpenSSL

Posteriormente, creamos un nuevo sitio en la carpeta sites-available con la configuración SSL y la ruta de los certificados.

```
 GNU nano 2.9.3                    /etc/nginx/sites-available/websrv.prue

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

        # SSL configuration
        listen 443 ssl default_server;
        server_name websrv.prueba;
        ssl on;
        ssl_certificate /etc/ssl/certificados/certi.crt;
        ssl_certificate_key /etc/ssl/certificados/certi.key;
        root /var/www/html;
```

Fig. 13. Configuración del sitio en Nginx

```
usuario@lnxsrv-web-01:~$ sudo ln -s /etc/nginx/sites-available/sslNginx /etc/nginx/sites-enabled/.
usuario@lnxsrv-web-01:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
usuario@lnxsrv-web-01:~$ sudo service nginx restart
```

Fig. 14. Activación de sitio y recarga del servicio Nginx

Finalmente, activamos el sitio y tras comprobar que la configuración es correcta, reiniciamos el servicio Nginx para activar cambios y publicar en sitios HTTPS.

/ 4. Módulos y aplicaciones

Anteriormente hemos visto como un servidor web por sí solo no es suficiente para ser totalmente funcional. Del mismo modo que PHP y MySQL permiten al servidor web ejecutar nuevas tareas, los módulos son otro tipo de software complementario que nos aportará funcionalidades extra, como por ejemplo cifrar datos y servir sitios web seguros usando HTTPS.

En este apartado hablaremos de los diferentes módulos que podemos encontrar tanto para servidores web, como para PHP.

4.1. Módulos PHP

Los módulos de PHP permiten que el preprocesador conecte con otros componentes, como podría ser Apache, Nginx, mysql, un servidor LDAP o un servidor de correo.

En la web oficial de PHP encontraremos un listado completo de sus módulos.

4.2. Módulos Nginx

Los módulos de Nginx se pueden clasificar en cinco grupos: Core, Event, HTTP, Mail y Stream. Para listar los módulos instalados utilizaremos el comando nginx –V.

Module Types	Module Name	Description
Core	ngx_core_module ngx_error_log_module ngx_conf_module	This includes modules that enable network and application protocols. It includes modules that handle logging, encryption, etc. It includes CPU architecture-specific Nginx configuration, it takes care of CPU affinity, thread pool allocation, memory management api, file and sockets IO, etc.
Event	ngx_events_module ngx_events_core_module	This includes connection processing methods that are event driven. Example connection pooling, etc.
HTTP	ngx_http_module ngx_http_core_module ngx_http_log_module ngx_http_upstream_module	This includes websServer functionality in Nginx and takes care of all HTTP requests.
Mail	ngx_mail_module ngx_mail_core_module	This includes mail proxy modules.
Stream	ngx_stream_module ngx_stream_core_module ngx_stream_proxy_module ngx_stream_upstream_module	This includes modules that enable proxy, load balance functionality in Nginx.

Fig. 15. Estructura docker

4.3. Módulos Apache

Apache cuenta con un amplio catálogo de módulos disponibles, entre ellos podemos destacar:

- **mod_ssl:** permite al servidor utilizar protocolos SSL y TSL para servir webs de forma segura usando HTTPS.
- **mod_auth_digest:** permite realizar autenticación de usuarios usando MD5 DigestN.
- **mod_rewrite:** permite activar la redirección de URL. Este módulo es necesario para utilizar los ficheros htaccess, de los que hablaremos en próximos temas.

En este enlace encontrarás un listado completo de los módulos de Apache: https://httpd.apache.org/docs/2.4/es/mod/

/ 5. Caso práctico 2: "Comparativa de rendimiento Nginx vs Apache"

Planteamiento:

En el caso práctico 1 vimos cómo Pedro preparaba un servidor para una demo, y algunos clientes le hicieron preguntas sobre qué servidor web era más potente. Como consecuencia, Pedro tendrá que elaborar un informe de rendimiento para entregar pruebas y así aportar datos reales sobre la velocidad que ofrecen sus servidores.

Nudo:

¿En qué se basará para elaborar el informe?

Desenlace:

Para el desarrollo de este caso práctico Pedro instalará dos VPS con las mismas características. En uno instalará Nginx y en otro Apache además de PHP y MySQL

Una vez instalados los dos servidores web, creará un archivo en el DocumentRoot del servidor con el siguiente código:

```
<h1 align=center>Caso práctico 2</h1>
<?php
phpinfo();
?>
```

Código 3. Creación de un index.php

Utilizando el comando AB (sobre el que profundizaremos a continuación) realizará pruebas y contrastará los datos obtenidos.

El informe que entregará será el mostramos en la siguiente página.

Resultados VPS Nginx

```
usuario@lnxsrv-web-01:~$ ab -k -n 1000 -c100 -H 'Accept-Encoding: gzip,deflate' http://localhost/info.
php/
   Server Software:       nginx/1.14.0
   Server Hostname:       localhost
   Server Port:           80
   Document Path:         /info.php
   Document Length:       80691 bytes
   Concurrency Level:     100
   Time taken for tests:  3.705 seconds
   Complete requests:     1000
   Failed requests:       969
     (Connect: 0, Receive: 0, Length: 969, Exceptions: 0)
   Keep-Alive requests:   0
   Total transferred:     80838777 bytes
   HTML transferred:      80692777 bytes
   Requests per second:   269.89 [#/sec] (mean)
   Time per request:      370.522 [ms] (mean)
   Time per request:      3.705 [ms] (mean, across all concurrent requests)
   Transfer rate:         21306.22 [Kbytes/sec] received
```

Código 3. Pruebas de rendimiento con AB sobre servidor Nginx

Resultados VPS Apache

```
usuario@lnxsrv-web-01:~$ ab -k -n1000 -c100 -H 'Accept-Encoding: gzip,deflate' http://localhost/info.
php
   Server Software:       Apache/2.4.29
   Server Hostname:       localhost
   Server Port:           80
   Document Path:         /info.php
   Document Length:       22855 bytes
   Concurrency Level:     100
   Time taken for tests:  8.928 seconds
   Complete requests:     1000
   Failed requests:       867
     (Connect: 0, Receive: 0, Length: 867, Exceptions: 0)
   Keep-Alive requests:   1000
   Total transferred:     23109178 bytes
   HTML transferred:      22856207 bytes
   Requests per second:   112.01 [#/sec] (mean)
   Time per request:      892.752 [ms] (mean)
   Time per request:      8.928 [ms] (mean, across all concurrent requests)
   Transfer rate:         2527.86 [Kbytes/sec] received
```

Código 4. Pruebas de rendimiento con AB sobre servidor Apache

/ 6. Pruebas de rendimiento

El paquete Apache-utils contiene la aplicación Apache Benchmark que permite testear el rendimiento de un servidor web. A partir de los datos que nos muestre 'AB' podremos tomar decisiones y modificar la configuración para conseguir que nuestro servidor web pueda atender un mayor número de peticiones con tiempos de espera menor. El funcionamiento de Apache Benchmark consiste en realizar una gran cantidad de peticiones a un servidor web.

Con este software podemos analizar cualquier tipo de servidor web, ya sea Apache, Nginx o IIS.

El proceso de instalación y pruebas sería el siguiente: En primer lugar, instalaremos el paquete apache2-utils

```
$sudo apt-get install apache2-utils
```

Código 5. Instalación del paquete apache2-utils

A continuación, ejecutaremos el comando AB con los siguientes parámetros:

- **k:** usar keep Alive
- **n:** número de peticiones que se realizarán
- **c:** cantidad de conexiones concurrentes
- **g ab.csv:** guardará el resultado en un archivo CSV
- **H 'Accept-Encoding:** gzip,deflate': imitará las peticiones que realizaría un navegador web

```
 $sudo ab –k –n2000 –c200 -H 'Accept-Encoding: gzip,deflate' –g ab.csv http://IP_WEBSERVER/
Server Software:        nginx/1.14.0
Server Hostname:        172.20.10.6
Server Port:            80

Document Path:          /
Document Length:        612 bytes

Concurrency Level:      200
Time taken for tests:   0.234 seconds
Complete requests:      2000
Failed requests:        0
Keep-Alive requests:    2000
Total transferred:      1718000 bytes
HTML transferred:       1224000 bytes
Requests per second:    8539.93 [#/sec] (mean)
Time per request:       23.419 [ms] (mean)
Time per request:       0.117 [ms] (mean, across all concurrent requests)
Transfer rate:          7163.87 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    1   4.4      0      16
Processing:     8   20   4.6     21      28
Waiting:        8   20   4.7     21      28
Total:         10   22   3.2     21      29
```

Código 6. Prueba de rendimiento con Apache Benchmark –
Extracto de: https://juantrucupei.wordpress.com/2015/11/10/pruebas-con-ab-apache-benchmark/

Vídeo 2. "Apache Bench + GNUPLOT·
https://bit.ly/3957yuv

/ 7. Resumen y resolución del caso práctico de la unidad

En esta unidad, hemos empezado a conocer qué ficheros tenemos que modificar para aplicar cambios en un servidor web y los ficheros que debemos revisar para detectar errores o monitorizar los accesos a los recursos del servidor.

Además, hemos hablado del protocolo HTTPS y de la manera de securizar el acceso a la información alojada en el servidor, evitando que alguien ajeno pueda capturar paquetes en nuestra red y descubrir información sensible, como credenciales o datos bancarios.

Finalmente, hemos hecho pruebas de rendimiento tanto de Apache como de Nginx, comparando resultados.

Resolución del caso práctico de la unidad

En el caso práctico inicial se plantea un problema de configuración que limita el tamaño máximo de los ficheros que se pueden subir al servidor.

Un posible fallo podría haber sido que un firewall estuviera bloqueando la subida, pero como el cliente nos confirma que archivos más pequeños sí puede, debemos revisar la configuración de nuestro servidor y aumentar los límites de los parámetros POST_MAX_SIZE y UPLOAD_MAX_FILESIZE en la configuración de PHP (php.ini)

Una vez reconfigurado, debemos reiniciar el servicio web y realizar pruebas.

```
; Maximum size of POST data that PHP will accept.
; Its value may be 0 to disable the limit. It is ignored if POST data reading
; is disabled through enable_post_data_reading.
; http://php.net/post-max-size
post_max_size = 8M
```

Fig. 16. Fichero php.ini. Configuración POST_MAX_SIZE

/ 8. Bibliografía

PHP: Descripción de las directivas del núcleo de php.ini - Manual. (s. f.). PHP: Descripción de las directivas del núcleo de php.ini - Manual. https://www.php.net/manual/es/ini.core.php#ini.upload-max-filesize

How To Create a Self-Signed SSL Certificate for Nginx on CentOS 7 | DigitalOcean. (2019, 18 septiembre). digitalocean. https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-on-centos-7

¿Qué es el Protocolo HTTPS y por qué es tan importante? (s. f.). https://es.ryte.com/wiki/HTTPS

Soni, R. (2016). Nginx: From Beginner to Pro. Apress.

Certificado SSL Instalación - Servidor Nginx. (s. f.). Digicert. https://www.digicert.com/es/instalar-certificado-ssl-nginx.htm

Trucupei, J. (2018, 25 mayo). Pruebas de Carga (Stress) con AB Apache Benchmark. Linux Trucupei Blog. https://juantrucupei.wordpress.com/2015/11/10/pruebas-con-ab-apache-benchmark/