MEMORIA DE PRÁCTICAS SERVIDOR WEB (HTTPS)

Pilar Guzmán Cabezas
2ºDAW

Introducción

En esta práctica se indicará cómo puedes configurar tu contenedor Docker para que se pueda acceder a una página web por https.

El contenedor docker tendrá que estar basado en la imagen httpd de dockerhub y usar un bind mount para publicar la página web.

Creación Bind mount y Docker https

Se utilizará un **Bind mount** para poder tener acceso en local a los archivos de la web que crearemos y estarán en el contenedor y poder modificarlos con los programas instalados en nuestro ordenador como el Visual Studio Code.

Creamos una **carpeta** en el Home donde iremos trabajando en esta práctica, en mi caso se llama "**memoria**".

Ejecutamos el siguiente **comando** en la terminal:

```
docker run --name HTTPS -p 80:80 -p 443:443 --mount type=bind,src=/home/estudiante/Home/memoria,dst=/usr/local/apache 2/htdocs httpd
```

Se indica lo siguiente:

indicamos su nombre ("HTTPS")
el mapeo de puertos (-p 80:80 -p 443:443)
el bind mount y la imagen que va a usar… (podemos probar que accedemos con normalidad por el puerto 80 pero no por el 443
el nombre del contenedor (httpd)

Con este comando se ha creado el contenedor y se ha iniciado al usar la orden "run".

Configuración

Accedemos al contenedor para configurarlo mediante el comando `docker exec`

```
estudiante@DAW1:~/memoria$ docker exec -it HTTPS /bin/bash
```

Una vez dentro del contenedor lo actualizamos con `apt-get update`

```
root@4044190da23c:/usr/local/apache2# apt-get update
Get:1 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 http://deb.debian.org/debian-security bullseye-security InRelease [48.4 kB]
Get:3 http://deb.debian.org/debian bullseye-updates InRelease [44.1 kB]
Get:4 http://deb.debian.org/debian bullseye/main amd64 Packages [8184 kB]
Get:5 http://deb.debian.org/debian-security bullseye-security/main amd64 Packages [194 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [14.6 kB]
Fetched 8600 kB in 3s (3014 kB/s)
Reading package lists... Done
```

Certificado SSL

Generar certificado SSL

Instalamos/usamos Openssl… para generar un certificado X509:

`apt-get install openssl`

```
root@4044190da23c:/usr/local/apache2# apt-get install openssl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssl is already the newest version (1.1.1n-0+deb11u3).
openssl set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
root@4044190da23c:/usr/local/apache2#
```

Nos ubicamos en la carpeta de configuración del contenedor mediante “cd”
y ejecutamos:

`openssl req -x509 -days 365 -newkey rsa:2048 -keyout server.key -out server.crt`

```
root@4044190da23c:/usr/local/apache2# cd ./
bin/     build/   cgi-bin/ conf/    error/   htdocs/  icons/   include/ logs/    modules/
root@4044190da23c:/usr/local/apache2# cd ./conf/
root@4044190da23c:/usr/local/apache2/conf# openssl req -x509 -days 365 -newkey rsa:2048 -keyout server.ke
y -out server.crt
Generating a RSA private key
.....+++++
.............................+++++
writing new private key to 'server.key'
Enter PEM pass phrase:
```

Nos pide unos datos para generar la clave:

```
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
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]:sevilla
Locality Name (eg, city) []:brenes
Organization Name (eg, company) [Internet Widgits Pty Ltd]:jacaranda
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []:pilar
Email Address []:.
root@4044190da23c:/usr/local/apache2/conf#
```

Configuración

Entramos al archivo de configuración: `/usr/local/apache2/conf/httpd.conf` haciendo uso de “nano”(*nano conf/httpd.conf*) y buscamos tres líneas específicas que hay que descomentar, son las siguientes:

```
#LoadModule file_cache_module modules/mod_file_cache.so
#LoadModule cache_module modules/mod_cache.so
#LoadModule cache_disk_module modules/mod_cache_disk.so
#LoadModule cache_socache_module modules/mod_cache_socache.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
#LoadModule socache_dbm_module modules/mod_socache_dbm.so
#LoadModule socache_memcache_module modules/mod_socache_memcache.so
#LoadModule socache_redis_module modules/mod_socache_redis.so
#LoadModule watchdog_module modules/mod_watchdog.so
```

```
#LoadModule session_dbd_module modules/mod_session_dbd.so
#LoadModule slotmem_shm_module modules/mod_slotmem_shm.so
#LoadModule slotmem_plain_module modules/mod_slotmem_plain.so
LoadModule ssl_module modules/mod_ssl.so
#LoadModule optional_hook_export_module modules/mod_optional_hook_e
#LoadModule optional_hook_import_module modules/mod_optional_hook_i
#LoadModule optional_fn_import_module modules/mod_optional_fn_impor
#LoadModule optional_fn_export_module modules/mod_optional_fn_expor
```

```
</IfModule>

# Secure (SSL/TLS) connections
Include conf/extra/httpd-ssl.conf
#
# Note: The following must must be present to support
```

Guardamos los cambios y **salimos**.

Reiniciamos el contenedor para que se guarden los cambios con:
`apachectl restart`

Ya podemos acceder a la web por https://ip:443 (nos marcará **Not secure** por motivos de seguridad)