Ayuda 1 Práctica 1: crear imagen semejante a “Apache sobre Ubuntu”

- **Preliminares**: nos situamos en el directorio donde vayamos a crear el Dockerfile , y creamos los .conf que precisemos en base a nuestras pretensiones (verbigracia en el Dockerfile siguiente: apache-config.conf) u otras creaciones que consideremos apropiadas en base al enfoque que le demos.

- **Dockerfile**

```
# Usa la imagen oficial de Apache 2.4 desde Docker Hub como base
FROM httpd:2.4

# Configura el DocumentRoot mediante variable de entorno
ENV APACHE_DOCUMENT_ROOT /var/www/html
# o Copia tus archivos de la aplicación web: COPY my-website/ /usr/local/apache2/htdocs/

ENV HTTPD_PREFIX /usr/local/apache2

# Copia la configuración personalizada de Apache
COPY apache-config.conf $HTTPD_PREFIX/conf/httpd.conf

# Cambia los permisos del DocumentRoot
RUN chown -R www-data:www-data $APACHE_DOCUMENT_ROOT && \
   chmod -R 755 $APACHE_DOCUMENT_ROOT

# Crea el directorio del DocumentRoot
RUN mkdir -p $APACHE_DOCUMENT_ROOT/software.com $APACHE_DOCUMENT_ROOT/hardware.com

# Configura software.com (no seguro - HTTP)
RUN echo "Listen 80" > $HTTPD_PREFIX/conf/extra/software.com.conf
RUN echo "<VirtualHost *:80>" >> $HTTPD_PREFIX/conf/extra/software.com.conf
RUN echo "    DocumentRoot $APACHE_DOCUMENT_ROOT/software.com" >> $HTTPD_PREFIX/conf/extra/software.com.conf
RUN echo "    ServerName software.com" >> $HTTPD_PREFIX/conf/extra/software.com.conf
RUN echo "</VirtualHost>" >> $HTTPD_PREFIX/conf/extra/software.com.conf
# También podríamos haber creado software.com.conf y hacer un “include” de este sobre httpd.conf
# Configura software.com (no seguro - HTTP)
# COPY software.com.conf $HTTPD_PREFIX/conf/extra/
# RUN echo "Include conf/extra/software.com.conf" >> $HTTPD_PREFIX/conf/httpd.conf
```

```
# Configura hardware.com (seguro - HTTPS)
RUN echo "Listen 443" > $HTTPD_PREFIX/conf/extra/hardware.com.conf
RUN echo "<VirtualHost *:443>" >> $HTTPD_PREFIX/conf/extra/hardware.com.conf
RUN echo "    DocumentRoot $APACHE_DOCUMENT_ROOT/hardware.com" >> $HTTPD_PREFIX/conf/extra/hardware.com.conf
RUN echo "    ServerName hardware.com" >> $HTTPD_PREFIX/conf/extra/hardware.com.conf
RUN echo "    SSLEngine on" >> $HTTPD_PREFIX/conf/extra/hardware.com.conf
RUN echo "    SSLCertificateFile /usr/local/apache2/conf/server.crt" >> $HTTPD_PREFIX/conf/extra/hardware.com.conf
RUN echo "    SSLCertificateKeyFile /usr/local/apache2/conf/server.key" >> $HTTPD_PREFIX/conf/extra/hardware.com.conf
RUN echo "</VirtualHost>" >> $HTTPD_PREFIX/conf/extra/hardware.com.conf
# También podríamos haber creado hardware.com.conf y hacer un “include” de este sobre httpd-ssl.conf
# Configura hardware.com (seguro - HTTPS)
# COPY hardware.com.conf $HTTPD_PREFIX/conf/extra/
# RUN echo "Include conf/extra/hardware.com.conf" >> $HTTPD_PREFIX/conf/httpd.conf

# Genera certificado autofirmado para hardware.com (de esta forma o cómo hicimos en clase)
RUN openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout $HTTPD_PREFIX/conf/server.key \
    -out $HTTPD_PREFIX/conf/server.crt \
    -subj "/C=US/ST=CA/L=Cieza/O=Albares/OU=A26/CN=hardware.com"

# Expone los puertos 80 y 443 para que puedan ser accesibles desde el host
EXPOSE 80
EXPOSE 443

# Ejecuta httpd-foreground como comando predeterminado
CMD ["httpd-foreground"]
```

- **Construimos y ejecutamos el contenedor**

Desde el directorio que contiene el Dockerfile, ejecutamos el siguiente comando para construir la imagen:

```
docker build -t imagen_tras_dockerfile .
```

- "**Persistimos**": utilizamos **directorios enlazados o volúmenes** para vincular los directorios en los contenedores que instancien esta última imagen (la creada tras el Dockerfile) con los directorios correspondientes en el sistema anfitrión

```
docker run -d -p 7777:80 -p 9999:443 --name=mi-apache-persi -v /ruta/anfitrion/software.com:/usr/local/apache2/htdocs/software.com -v /ruta/anfitrion/hardware.com:/usr/local/apache2/htdocs/hardware.com imagen_tras_dockerfile
```

```
docker run -d -p 7777:80 -p 9999:443 --name=mi-apache-persi \
   --mount type=bind,source=/ruta/anfitrion/software.com,target=/usr/local/apache2/htdocs/software.com \
   --mount type=bind,source=/ruta/anfitrion/hardware.com,target=/usr/local/apache2/htdocs/hardware.com \
   imagen_tras_dockerfile
```

- Después de confirmar que el contenedor se comporta como deseas, debes hacer un **commit para crear la nueva imagen final demandada** a instanciar en contenedores

```
docker commit mi-apache-persi imagen_apache_ubuntu
```

- Y ya **empezar a crear contenedores a partir de la imagen final buscada** y ya creada (dándole **ip** o averiguando la que coge)

```
docker run -d --name mi-apache-1 -p 7777:80 -p 9999:443 imagen_apache_ubuntu
```

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mi-apache-1
```

```
# o ip a
```

```
docker run -d --name mi-apache-1 --ip 192.168.0.120 -p 7777:80 -p 9999:443 imagen_apache_ubuntu
```

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mi-apache-1
```

- **Probarlo**, dando previamente de alta en **/etc/hosts** (y/o en la reglas de entrada del **router**)

IP software.com
IP hardware.com

verbigracia IP = 192.169.0.120, 172.17.X.Y, …

Ahora, deberías ser capaz de acceder a:

http://software.com:7777
https://hardware.com:9999

http://localhost:7777
    # en este caso, no necesitarías editar el archivo /etc/hosts
https://localhost:9999
    # en este caso, no necesitarías editar el archivo /etc/hosts

Ayuda 2 Práctica 1: crear imagen semejante a "Apache sobre Ubuntu"

Una variante interesante podría ser como todo lo anterior (Ayuda 1 Práctica 1) pero utilizando el siguiente Dockerfile

```
 # Utiliza la imagen oficial de PHP con Apache desde DockerHub
FROM php:apache

# Desinstala PHP
RUN apt-get update && \
    apt-get remove -y php* && \
    apt-get autoremove -y && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
```