

Introducción a Docker

UD 04. Caso práctico 03 - Apache 2 con PHP desde Alpine



Fons Social Europeu

L'FSE inverteix en el teu futur

Autor: Sergi García Barea

Actualizado Marzo 2023

Licencia




Reconocimiento – NoComercial - CompartirIgual (BY-NC-SA): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 **Importante**

 **Atención**

 **Interesante**

Introducción	3
Descargando la aplicación	3
Preparando el Dockerfile y creando la imagen	3
Probando la imagen	4
Bibliografía	4

UD04. CASO PRÁCTICO 03

1. INTRODUCCIÓN

En este caso práctico vamos a crear una imagen que partiendo de la imagen base “alpine”, una de las imágenes más reconocidas como “imagen ligera”, le instalaremos Apache 2, PHP y pondremos un simple `<?php phpinfo(); ?>` obtenido desde un fichero de Internet.

2. DESCARGANDO LA APLICACIÓN

Antes de empezar, debemos crear un directorio (por ejemplo, “Caso4-3”) y en el descargar la sencilla aplicación con la aplicación que ejecuta el comando `<?php phpinfo(); ?>` que encontraremos tanto en un fichero comprimido “zip” de este curso como en <https://gist.github.com/SyntaxC4/5648247>

Asimismo, de los ficheros proporcionados en el caso práctico obtendremos el script “**start.sh**” que deberá estar situado en el directorio que hemos creado. El contenido de dicho script es el siguiente:

```
#!/bin/sh
#Lanzamos servicio Apache2 en segundo plano
/usr/sbin/httpd -D FOREGROUND
```

3. PREPARANDO EL DOCKERFILE Y CREANDO LA IMAGEN

En el directorio creando anteriormente, o crearemos (o en el caso del “zip” del curso, ya tendremos listo) el siguiente fichero “**Dockerfile**”:

```
FROM alpine
MAINTAINER Sergi <sergi.profesor@gmail.com>
#Actualizamos e instalamos paquetes con APK para Alpine
RUN apk update && apk add apache2 php php-apache2 openrc tar
#Copiamos script para lanzar Apache 2
ADD ./start.sh /start.sh
#Descargamos un ejemplo de <?php phpinfo(); ?> por enseñar como bajar algo de Internet
#Podría haber sido simplemente
#RUN echo "<?php phpinfo(); ?>" > /var/www/localhost/htdocs/index.php
ADD https://gist.githubusercontent.com/SyntaxC4/5648247/raw/94277156638f9c309f2e36e19bff378ba7364907/info.php /var/www/localhost/htdocs/index.php

# Si quisiéramos algo como Wordpress haríamos
#ADD http://wordpress.org/latest.tar.gz
/var/www/localhost/htdocs/wordpress.tar.gz
#RUN tar xvzf /var/www/localhost/htdocs/wordpress.tar.gz && rm -rf
/var/www/localhost/htdocs/wordpress.tar.gz
```

```
# Usamos usuario y grupo www-data. El grupo lo crea Apache, pero si
quisiéramos crear grupo
# Grupo www-data RUN set -x && addgroup -g 82 -S www-data
# Creamos usuario www-data y lo añadimos a ese grupo
RUN adduser -u 82 -D -S -G www-data www-data
# Hacemos todos los ficheros de /var/www propiedad de www-data
# Y damos permisos a esos ficheros y a start.sh
RUN chown -R www-data:www-data /var/www/ && chmod -R 775 /var/www/ &&
chmod 755 /start.sh
#Indicamos puerto a exponer (para otros contenedores) 80
EXPOSE 80
#Comando lanzado por defecto al instalar el contenedor
CMD /start.sh
```

El funcionamiento del propio “Dockerfile” está definido por sus propios comentarios. Una vez preparado, crearemos la imagen con:

```
docker build -t alpineapache ./
```

Con esa línea indicamos que creamos la imagen “**alpineapache**” basándose en el fichero “**Dockerfile**” del directorio actual.

```
sergi@ubuntu:~/Desktop/alpineapache$ docker build -t alpineapache ./
Sending build context to Docker daemon 4.096kB
Step 1/9 : FROM alpine
latest: Pulling from library/alpine
Digest: sha256:a75afd8b57e7f34e4dad8d65e2c7ba2e1975c795ce1ee22fa34f8cf46f96a3be
Status: Downloaded newer image for alpine:latest
```

4. PROBANDO LA IMAGEN

Con el siguiente comando, podremos lanzar la aplicación en el puerto 80 de nuestra máquina:

```
docker run -dp 80:80 alpineapache
```

Una vez hecho esto, podremos probar la aplicación accediendo a <http://localhost/index.html> (donde veremos un mensaje similar a “It works!”) y <http://localhost/index.php> donde veremos el contenido de la llamada a la función “**php_info()**”.

5. BIBLIOGRAFÍA

[1] Docker Docs <https://docs.docker.com/>