Puesta en Producción Segura

Práctica 4: DevSecOps

Objetivo

El objetivo de esta práctica es que el alumno comprenda los fundamentos de las DevOps seguras, también conocidas como DevSecOps.

Corrección de la práctica

El alumno deberá entregar una memoria con las soluciones a cada apartado y los pasos detallados que ha seguido en el proceso. Se valorará:

- La calidad de la memoria. Se deberán explicar y justificar los procedimientos y resultados,
- así como de dar soporte a lo descrito mediante capturas de pantalla.
- La corrección del proceso descrito.
- La corrección de los comandos empleados y de las opciones.
- La correcta justificación de los pasos seguidos.
- La propuesta de soluciones alternativas, documentando incluso aquellas que siendo razonables no han logrado su propósito.

Forma de entrega: Vía Moodle. Se habilitará una tarea a tal efecto.

Docker

1. Crea un Dockerfile que partiendo de una imagen PHP genera una imagen que:
    1. Copia una aplicación en PHP a un directorio del contenedor. Esta aplicación se debe copiar directamente desde un directorio del anfitrión. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).

Este es el contenido del Dockerfile:

```
usuario@pps:~$ cat Dockerfile
# Usa una imagen base de PHP
FROM php:7.4-cli

# Copia la aplicación PHP desde el directorio del host al contenedor
COPY ./mi_aplicacion_php /var/www/html/

# Expone el puerto 80 para que puedas acceder al servicio (en caso de que agregues un servidor web)
EXPOSE 80

# Comando por defecto para ejecutar la aplicación PHP
CMD ["php", "-S", "0.0.0.0:80", "-t", "/var/www/html"]
usuario@pps:~$
```

Este es el contenido del fichero index.php que se encuentra en el directorio /mi_aplicación_php

```
usuario@pps:~$ cat ./mi_aplicacion_php/index.php
<?php
echo "Hola Rafa, este es mi contenedor de Docker con php";
?>
usuario@pps:~$
```

2. Crea un Dockerfile que partiendo de una imagen Ubuntu genera una imagen que:
    1. Instala Apache, de forma que se exponga el puerto 80.
    2. Instala PHP.
    3. Copia una aplicación web en PHP al directorio de Apache que expone las páginas web. Esta aplicación se debe descargar automáticamente mediante algún comando como git clone o curl. Para facilitar las cosas, debe de ser una aplicación sencilla que no emplee bases de datos (ya que si no también habría que instalar un MySQL).

Este es el contenido del archivo Dockerfile2

```
usuario@pps:~$ cat Dockerfile2
#Imagen base
FROM ubuntu:20.04

#Configurar el entorno no interactivo para evitar problemas con tzdata
ENV DEBIAN_FRONTEND=noninteractive

#Actualizar e instalar Apache, PHP, Git y Curl
RUN apt-get update && \
    apt-get install -y apache2 php git curl && \
    apt-get clean

# Clonar en un directorio temporal
RUN git clone https://github.com/banago/simple-php-website.git /tmp/app \
    && rm -rf /var/www/html/* \
    && cp -r /tmp/app/* /var/www/html/

#Establecer los permisos adecuados
RUN chown -R www-data:www-data /var/www/html

#Exponer el puerto 80
EXPOSE 80

#Iniciar Apache en primer plano
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]

usuario@pps:~$
```

3. Crea un contenedor para cada una de esas imágenes y verifica que funciona. Para y borra dicho contenedor.

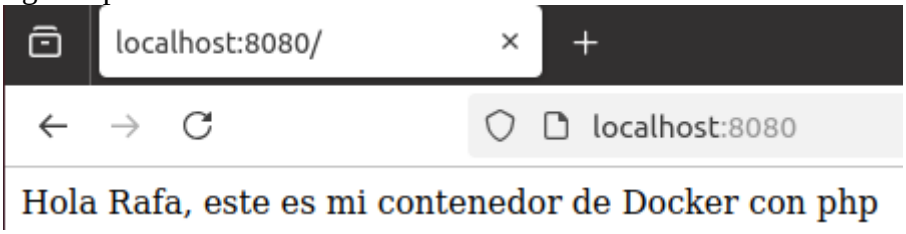
Creamos el primer contenedor

```
usuario@pps:~$ docker build -t imagen_php .
[+] Building 1.8s (7/7) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 418B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/php:7.4-cli                                                                1.8s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [internal] load build context                                                                                             0.0s
 => => transferring context: 163B                                                                                             0.0s
 => CACHED [1/2] FROM docker.io/library/php:7.4-cli@sha256:620a6b9f4d4feef2210026172570465e9d0c1de79766418d3affd09190a7fda5  0.0s
 => [2/2] COPY ./mi_aplicacion_php /var/www/html/                                                                             0.0s
 => exporting to image                                                                                                        0.0s
 => => exporting layers                                                                                                       0.0s
 => => writing image sha256:26f513755c682b41d183f19fcd249299f5f0e4189780f34ba3567d21f78d8c5a                                  0.0s
 => => naming to docker.io/library/imagen_php                                                                                 0.0s
usuario@pps:~$
```

Arrancamos el contenedor en el puerto 8080

```
usuario@pps:~$ docker run -d -p 8080:80 imagen_php
9756ac7c638ed65b3141ce337cc7acac1fc0612680bd600e704ddbfcda43dbed
usuario@pps:~$
```

Abrimos el navegador para ver el resultado



Paramos el contenedor

```
usuario@pps:~$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS                                          NAMES
9756ac7c638e   imagen_php   "docker-php-entrypoi…"   4 minutes ago   Up 4 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   exciting_heyrovsky
usuario@pps:~$ docker stop exciting_heyrovsky
exciting_heyrovsky
usuario@pps:~$
```

Creamos el segundo contenedor

```
usuario@pps:~$ docker build -f Dockerfile2 -t mi_aplicacion_web_php .
[+] Building 0.0s (8/8) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile2                                                                        0.0s
 => => transferring dockerfile: 695B                                                                                         0.0s
 => [internal] load metadata for docker.io/library/ubuntu:20.04                                                              0.0s
 => [internal] load .dockerignore                                                                                            0.0s
 => => transferring context: 2B                                                                                              0.0s
 => [1/4] FROM docker.io/library/ubuntu:20.04                                                                                0.0s
 => CACHED [2/4] RUN apt-get update &&     apt-get install -y apache2 php git curl &&     apt-get clean                    0.0s
 => CACHED [3/4] RUN git clone https://github.com/banago/simple-php-website.git /tmp/app     && rm -rf /var/www/html/*     && cp -r /tmp/ap  0.0s
 => CACHED [4/4] RUN chown -R www-data:www-data /var/www/html                                                                0.0s
 => exporting to image                                                                                                       0.0s
 => => exporting layers                                                                                                      0.0s
 => => writing image sha256:0bec5915aa48082af28ad3e25faca705c7ed1d0bebf398beb8a93686500a09d1                                 0.0s
 => => naming to docker.io/library/mi_aplicacion_web_php                                                                     0.0s
usuario@pps:~$
```

Arrancamos el contenedor que creamos en el paso anterior

```
usuario@pps:~$ docker run -d -p 8080:80 mi_aplicacion_web_php
15f980e3baa19299fb1aec5f7e84e4084b1810aac0ee2bfa1271023adcf3d94b
usuario@pps:~$
```

Abrimos el navegador y este es el resultado:

Por último, paramos el contenedor

```
usuario@pps:~$ docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS         PORTS                                       NAMES
15f980e3baa1   mi_aplicacion_web_php   "/usr/sbin/apache2ct…"   4 minutes ago   Up 4 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   hopeful_j
ohnson
usuario@pps:~$ docker stop hopeful_johnson
hopeful_johnson
usuario@pps:~$
```

4. Emplea un comando para lanzar 20 contenedores de la segunda imagen, cada uno mapeado en un puerto distinto del anfitrión. Cuando veas que funcionan, para y borra dichos contenedores.

Lanzamos los 20 contenedores a la vez

```
usuario@pps:~$ for i in {1..20}; do docker run -d --name php_app_$i -p $((8080 + i)):80 mi_aplicacion_web_php; done
aef7c1f700c24cdf695922f6d2c4f3d4b3fec4b620ca2ea9b051b7bd39cee14a
b2e0a862cca2a9cc82a94ff5d4a23cc5f039db96960e8c3c9813aff49dcd2d1d
396a6a4f359777e3d3c20d042cc445d89cac12eb0cb6cbcb5f06ccdf6e3ce659
5bcfa7767a9e26b286d1fc5df6be73b594ec1eaa2920d75cc641dda33504f025
bcb094b7cbdf04e74c959708d62e5befd8a2094e7760dc780b4decd3d050d313
7a85bc63568bb24e7ade071c084633150b7363f5bacdd3d912241541fe5b1bfd
c62af6bc2c922272d5d423065e57b3d17fa97143ac0fd04ff471210e59b52ef9
bfd0283850ae14a08428db1be31bcd924d741484ad58c53ce224d741bb331842
3c582ce458c79abd56bd1a50c8005d6cb4baff90cf68cc4283c66af7ca50e9a3
c7feac8071250540bd1f5661e41182b354c7141182903087469981195eca24dd
eb6aa4af48af48c4f4ff7cbaf474d7f08368e957b8678f3170c3abb6d1d6a52b
79657f4a8d2ddede3e5b2e99b047e822116f6abafdbfd9642a3b77e4e59d4a73
e63f64114599f09923b9bd2aaab96d1d2c1fceeb2def7960e8980388a72876c8
06ae498d0379c5f3ae3a958a0456ab3fc8628941c39682389bafdbcdd6791170
54ceb4e7bb86ae876a8a0551f52e20ee622a442d93f6c317560359c0c8b786d2
7ce334a9cecd34296401f3d74b91678215907d47e4051fbcecb09df178d26bb8
ad35a2a19d9bf419f8d6230b9fd2893dffe436593a540ad7703e30f74033b123
c5ab02c7d32b0361d339b910286a4c931e262bfa7afb2be5a7572d343381a342
faf93b03dd32491f97562c351d762ca758137d3f60cfb61ed5f297a7360fa01e
fc32745f0805de0bebcfa94850c6df146d7f74abccb382388389addb8c9d6750
usuario@pps:~$
```

Verificamos que están corriendo

```
usuario@pps:~$ docker ps --filter "name=php_app_"
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                                   NAMES
fc32745f0805   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8100->80/tcp, [::]:8100->80/tcp   php_app_20
faf93b03dd32   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8099->80/tcp, [::]:8099->80/tcp   php_app_19
c5ab02c7d32b   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8098->80/tcp, [::]:8098->80/tcp   php_app_18
ad35a2a19d9b   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8097->80/tcp, [::]:8097->80/tcp   php_app_17
7ce334a9cecd   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8096->80/tcp, [::]:8096->80/tcp   php_app_16
54ceb4e7bb86   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8095->80/tcp, [::]:8095->80/tcp   php_app_15
06ae498d0379   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8094->80/tcp, [::]:8094->80/tcp   php_app_14
e63f64114599   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8093->80/tcp, [::]:8093->80/tcp   php_app_13
79657f4a8d2d   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8092->80/tcp, [::]:8092->80/tcp   php_app_12
eb6aa4af48af   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8091->80/tcp, [::]:8091->80/tcp   php_app_11
c7feac807125   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8090->80/tcp, [::]:8090->80/tcp   php_app_10
3c582ce458c7   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8089->80/tcp, [::]:8089->80/tcp   php_app_9
bfd0283850ae   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8088->80/tcp, [::]:8088->80/tcp   php_app_8
c62af6bc2c92   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8087->80/tcp, [::]:8087->80/tcp   php_app_7
7a85bc63568b   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8086->80/tcp, [::]:8086->80/tcp   php_app_6
bcb094b7cbdf   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8085->80/tcp, [::]:8085->80/tcp   php_app_5
5bcfa7767a9e   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8084->80/tcp, [::]:8084->80/tcp   php_app_4
396a6a4f3597   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8083->80/tcp, [::]:8083->80/tcp   php_app_3
b2e0a862cca2   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8082->80/tcp, [::]:8082->80/tcp   php_app_2
aef7c1f700c2   mi_aplicacion_web_php  "/usr/sbin/apache2ct…"   3 minutes ago   Up 3 minutes   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp   php_app_1
usuario@pps:~$
```

Los paramos

```
usuario@pps:~$ for i in {1..20}; do docker stop php_app_$i; done
php_app_1
php_app_2
php_app_3
php_app_4
php_app_5
php_app_6
php_app_7
php_app_8
php_app_9
php_app_10
php_app_11
php_app_12
php_app_13
php_app_14
php_app_15
php_app_16
php_app_17
php_app_18
php_app_19
php_app_20
usuario@pps:~$
```

Y por último los borramos

```
usuario@pps:~$ for i in {1..20}; do docker rm php_app_$i; done
php_app_1
php_app_2
php_app_3
php_app_4
php_app_5
php_app_6
php_app_7
php_app_8
php_app_9
php_app_10
php_app_11
php_app_12
php_app_13
php_app_14
php_app_15
php_app_16
php_app_17
php_app_18
php_app_19
php_app_20
usuario@pps:~$
```

Git

Mantén un repositorio de Git público con:

- Memoria de la práctica.
- El fichero Dockerfile.

Haz un commit y un push del proyecto cada vez que superes uno de los ítems de la sección anterior.

La primera vez que subimos contenido al repositorio hacemos lo siguiente:

```
usuario@pps:~/docker-ubuntu-php-david$ git config --global user.name "DavidDizCIFP"
usuario@pps:~/docker-ubuntu-php-david$ git config --global user.email "david.diz.nieves@gmail.com"
usuario@pps:~/docker-ubuntu-php-david$ git init
Reinicializado el repositorio Git existente en /home/usuario/docker-ubuntu-php-david/.git/
usuario@pps:~/docker-ubuntu-php-david$ echo "# docker-ubuntu-php-david" >> README.md
usuario@pps:~/docker-ubuntu-php-david$ git add README.md
usuario@pps:~/docker-ubuntu-php-david$ git commit -m "Primer ejercicio"
[master 5442e6c] Primer ejercicio
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
usuario@pps:~/docker-ubuntu-php-david$ git branch -M main
```

Tras esto con push, subimos las cosas

```
usuario@pps:~/docker-ubuntu-php-david$ git push -u origin main
Username for 'https://github.com': DavidDizCIFP
Password for 'https://DavidDizCIFP@github.com':
Enumerando objetos: 6, listo.
Contando objetos: 100% (6/6), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (6/6), 773 bytes | 386.00 KiB/s, listo.
Total 6 (delta 0), reusado 0 (delta 0)
To https://github.com/DavidDizCIFP/docker-ubuntu-php-david.git
 * [new branch]      main -> main
Rama 'main' configurada para hacer seguimiento a la rama remota 'main' de 'origin'.
usuario@pps:~/docker-ubuntu-php-david$
```

Añadimos el segundo archivo Dockerfile que creamos para el segundo ejercicio:

```
usuario@pps:~/docker-ubuntu-php-david$ git add Dockerfile2
usuario@pps:~/docker-ubuntu-php-david$ git commit -m "Segundo ejercicio"
[main f91d52c] Segundo ejercicio
 1 file changed, 25 insertions(+)
 create mode 100644 Dockerfile2
usuario@pps:~/docker-ubuntu-php-david$ git push -u origin main
Username for 'https://github.com': DavidDizCIFP
Password for 'https://DavidDizCIFP@github.com':
Enumerando objetos: 4, listo.
Contando objetos: 100% (4/4), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (3/3), listo.
Escribiendo objetos: 100% (3/3), 736 bytes | 736.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
To https://github.com/DavidDizCIFP/docker-ubuntu-php-david.git
   5442e6c..f91d52c  main -> main
Rama 'main' configurada para hacer seguimiento a la rama remota 'main' de 'origin'.
usuario@pps:~/docker-ubuntu-php-david$
```