JÓVENES BICENTENARIO 3.0 – CERTUS

Docente: EDWIN MARAVI PÉREZ

POR: CARRION HUACANI GEAN CARLO

ACTIVIDAD ASINCRÓNICA N° 07

DOCKER

1. Mejores prácticas de Seguridad en Docker

Nunca expongas el puerto 2375 a Internet. Esto es aún más importante para los servidores privados virtuales que normalmente exponen todos los puertos. El puerto 2375 solo debería exponerse a la red interna (127.0.0.1:2375). Si no se refuerzan las vulnerabilidades de seguridad de la ventana acoplable, puede provocar un desastre. A continuación, se muestran algunos escenarios de ejemplo:

- Ejecutar un contenedor como root o privilegios elevados puede abrir la puerta para que una aplicación se haga cargo de su host Docker.
- Las imágenes de Docker que no son de confianza pueden tener código malicioso que podría comprometer datos confidenciales o incluso exponerlos intencionalmente.
- Los servicios de Docker pueden consumir intencional o involuntariamente los recursos de su host, lo que provoca fallas o que los recursos no estén disponibles para otras aplicaciones.
- Exponer el socket de docker, que es propiedad de root, a contenedores puede llevar a una toma de control de todo el sistema. Por ejemplo, Traefik requiere acceso al socket de la ventana acoplable. Entonces, si usa Traefik y si Traefik está comprometido, entonces su sistema también está comprometido.
- Una protección inadecuada (por ejemplo, un sistema de autenticación débil) puede poner en peligro sus aplicaciones web.
- El malware en Docker puede utilizar tus recursos para fines no deseados (por ejemplo, minería de cifrado). (elhacker.NET, 2021)

Asegurar el host de Docker

- Mantener actualizado el host de Docker
- NO Exponer socket del demonio Docker
- Usar un usuario para docker
- Limitar capacidades (Otorgar solo capacidades específicas, necesarias para un contenedor)
- Agregar el indicador –no-new-privileges
- Deshabilitar la comunicación entre contenedores (--icc = false)
- Limitar recursos (memoria, CPU, descriptores de archivos, procesos, reinicios)
- Establecer el sistema de archivos y los volúmenes en solo lectura
- Utilizar un cortafuego
- Utilizar un proxy inverso

Seguridad en Docker – Hardening

- No cambies la propiedad de Docker Socket
- No ejecutar contenedores Docker como root
- Utilizar el modo privilegiado con cuidado
- Utilizar imágenes de Docker de confianza
- Utilizar los secretos de Docker
- Utilizar los secretos en los servicios de Docker
- Controlar el uso de recursos de Docker
- Utilizar Herramientas para Auditar la Seguridad Contenedores Docker

2. Cómo establecer un orden de inicio en el archivo docker-compose.yaml

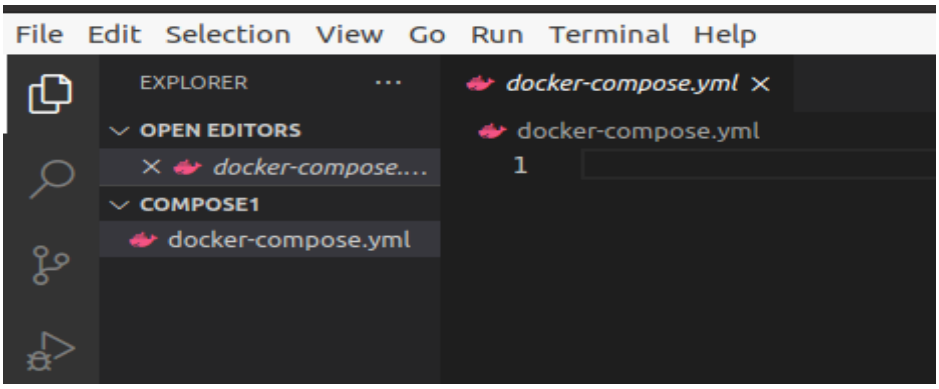
Una vez realizada la instalación de Docker Compose , se configura nuestro ficheros ***yml*** y al igual que hicimos con *Dockerfile* necesitaremos:

- Un espacio de trabajo /home/manu/projects/compose1
- Un editor de texto.

Primer fichero Docker Compose

Dentro del espacio de trabajo se crea un nuevo archivo llamado docker-compose.yml (también es posible hacerlo con la extensión **yaml**)

```
manu@manu-CX61-2QC:~/projects/compose1$ touch docker-compose.yml
manu@manu-CX61-2QC:~/projects/compose1$ code .
```

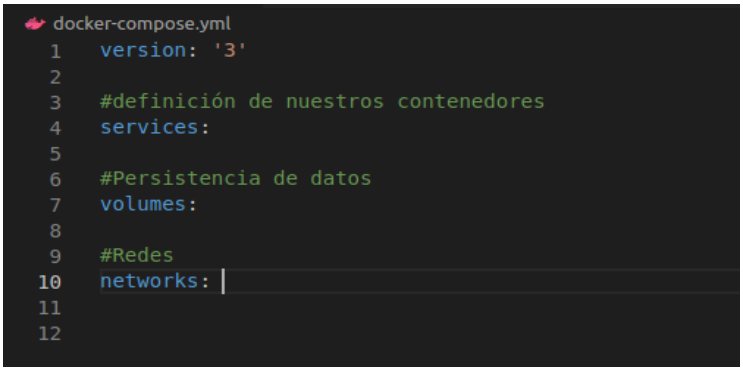


¿De qué secciones consta el fichero?

Cada fichero docker-compose.yml va a estar formado por las siguientes secciones:

- version (**Obligatorio**. Si no se indica una version se trataría de la versión 1 que está en desuso)
- services (**Obligatorio**. Debe incluir al menos un servicio). Que nos servirá para configurar todos nuestros contenedores partiendo de una imagen base, además de poder indicar variables de entorno o establecer ubicaciones.
- volumes (Opcional). Donde podremos definir la persistencia de datos de nuestros contenedores.
- networks (Opcional). Para definir las redes con las que conectar los distintos servicios.

Luego una plantilla de ejemplo que podrían tener nuestros docker-compose.yml sería:



Una cuestión importante que debemos tener en cuenta al trabajar con ficheros yml es que haremos uso de tabulaciones para **indentar** el código que vayamos escribiendo.

Ejemplo sencillo haciendo uso de una imagen base de *nginx:alpine* Referencia Nginx:alpine

```yaml
version: '3.8'

#definición de nuestros contenedores
services:
  nginx: #Nombre del servicio
    image: nginx:alpine #Nombre de la imagen que vamos a usar
    container_name: nginxc #Nombre del Contenedor
```

Y esto sería todo lo necesario para poner en marcha nuestra primera prueba. Desde una terminal podemos ejecutar los comandos que necesitemos, se recomienda estar ubicado dentro de la ruta donde se encuentra nuestro fichero docker-compose.yml, ya que, si no, nos veremos obligados a especificar la ruta absoluta en cada comando que queramos ejecutar.

Comprobemos que nuestra configuración funciona escribiendo: ***docker-compose up***

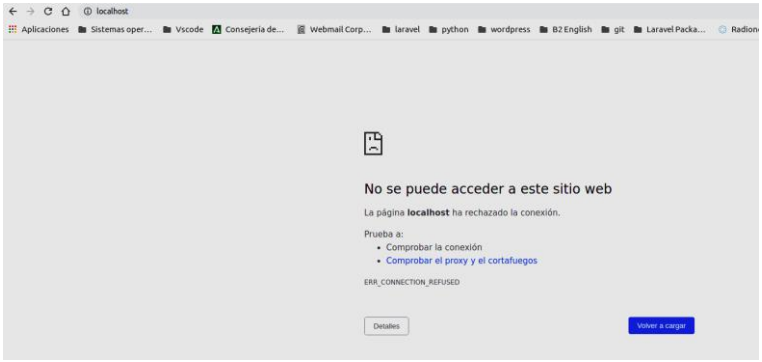
```
manu@manu-CX61-2QC:~/projects/compose1$ docker-compose up
Building with native build. Learn about native build in Compose here: https://docs.docker.com/go/compose-native-build/
Creating network "compose1_default" with the default driver
Pulling nginx (nginx:alpine)...
alpine: Pulling from library/nginx
ba3557a56b15: Pull complete
1a18b9f93d41: Pull complete
38ceab6c6432: Pull complete
6104f3bd82cc: Pull complete
750e0e12d70c: Pull complete
d7c38a871210: Pull complete
Digest: sha256:14536d83ca3128923ee7c2f7f4f285e023abd40f3ccdc8911f56cd4119558506
Status: Downloaded newer image for nginx:alpine
Creating nginxc ... done
Attaching to nginxc
nginxc    | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginxc    | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginxc    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginxc    | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginxc    | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
nginxc    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginxc    | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginxc    | /docker-entrypoint.sh: Configuration complete; ready for start up
```

Si abrimos una nueva terminal y escribimos: ***docker-compose ps***

veremos lo siguiente:

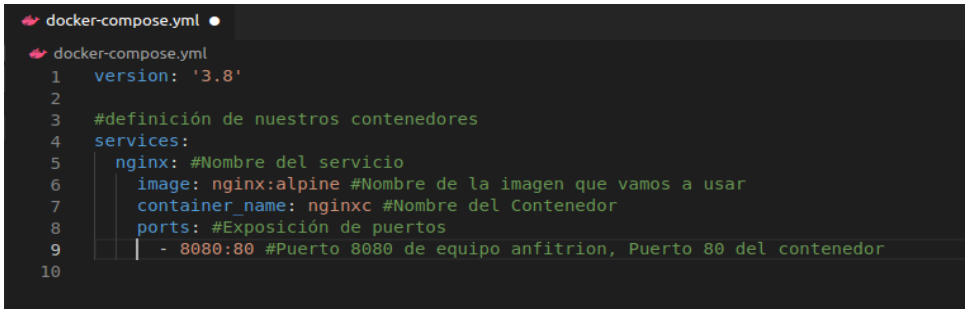
```
manu@manu-CX61-2QC:~/projects/compose1$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS     NAMES
d02b98f6e807   nginx:alpine   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    nginxc
```

Nuestro contenedor está en ejecución, pero si tratamos de acceder mediante: ***https://localhost*** nos dirá que la página no es accesible

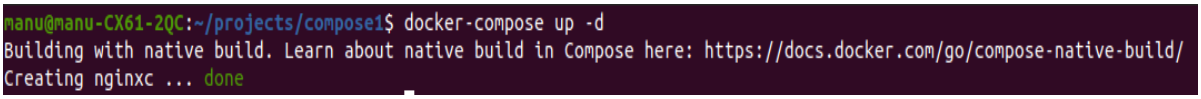


¿Por qué sucede esto?

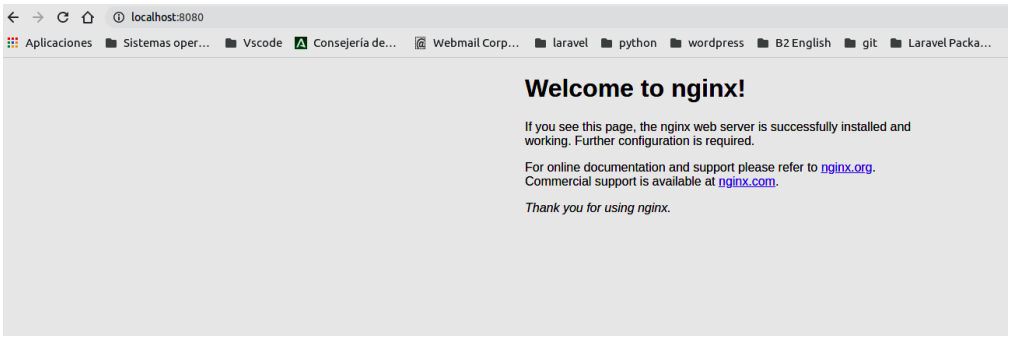
La respuesta es sencilla, el fichero Dockerfile de nuestra imagen base, expone el puerto 80, pero nosotros no le hemos indicado en el fichero docker-compose.yml qué puerto de la máquina anfitrión vamos a mapear para acceder al servicio de nginx. Para ello deberemos incluir en nuestro fichero de configuración lo siguiente:



Ahora si nos tiene que dejar acceder a nuestro servicio nginx, lanzaremos el contenedor de nuevo y para no dejar la terminal capturada añadiremos el flag -d (detach) como hacíamos con docker run: **docker-compose up -d**



Y si tratamos de acceder al servicio a través del puerto 8080 de nuestro equipo:



Referencias

elhacker.NET. (2021). *Mejores prácticas de seguridad en Docker*. Obtenido de https://blog.elhacker.net/2021/03/mejores-practicas-de-seguridad-en-contenedores-docker-hardening.html

I.E.S. Celia Viñas - Ciberseguridad. (s.f.). *ESTRUCTURA DE CONFIGURACIÓN DE UN ARCHIVO DOCKER COMPOSE*. Obtenido de https://iescelia.org/ciberseguridad/serie-docker-estructura-de-configuracion-de-un-archivo-docker-compose/

CPR de Zafra. (2021). *El comando docker-compose*. Obtenido de https://iesgn.github.io/curso_docker_2021/sesion5/comando.html

DockerTips. (2020). *Aprendiendo a utilizar Docker Compose*. Obtenido de https://dockertips.com/utilizando-docker-compose

Linode. (2021). *Nueva guía de fundamentos de seguridad para contenedores Docker*. Obtenido de https://www.linode.com/es/blog/security/new-docker-container-security-essentials-guide/