DOCKER Y CONTAINERS

18 DE NOVIEMBRE DE 2025

ALEJANDRO SAINZ SAINZ
CLOUD COMPUTING

1 SIEMPRE ME PASA ALGO, PERO APRENDO

Sin extenderme mucho. Por alguna razón Docker no iba bien en mi equipo. Ha dado muchos problemas. Quizá el propio docker o el WSL 2. Así que, sin miedo al fracaso. Opto por eliminar al intermediario, preparo una MV de Debian Linux, como buen vegano de la informática, e instalo Docker en ella y alguna que otra cosa más. De todo se aprende.

2 COMENZANDO

Evidentemente al no contar con un entorno gráfico, para alguien que no sabe del todo como funciona esta tecnología, esto se complica un poco. Pero bueno, San Google y Santa Paciencia ayudan. Pero bueno, vamos poco a poco.

Durante esta práctica, además de hacer lo que se pide, más o menos. He intentado añadir alguna otra cosa alternativa. Iré explicando poco a poco cada cosa.

Lo primero, después de crear la MV Debian, es buscar la forma de instalar Docker. De eso no he puesto capturas, pero hay varias formas y alternativas. Docker, propiamente dicho, o Docker Engine, que es parecido, aunque tiene una serie de diferencia. No he profundizado tanto en las diferencias. En este caso sudo apt install docker, después de añadir alguna dependencia, ha hecho el trabajo.

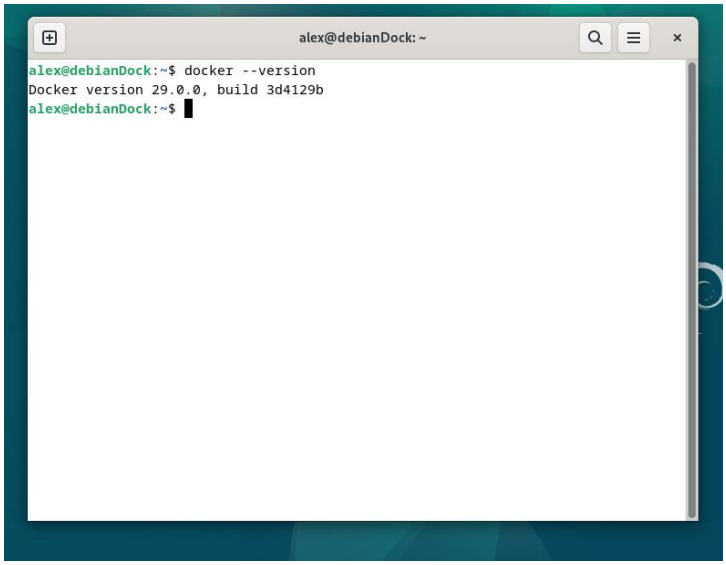

Ilustración 1 COMPROBANDO VERSION

Una vez tenemos instalado Docker, en la terminal, podemos usar el comando docker –version para comprobar que la instalación ha sido exitosa. A partir de aquí, ya todo es relativo a la tarea en cuestión.

Ahora lo que tengo que hacer es crear una página web, sencilla. Yo he creado la siguiente, con su archivo CSS. Pero aquí viene uno de los pequeños inconvenientes.

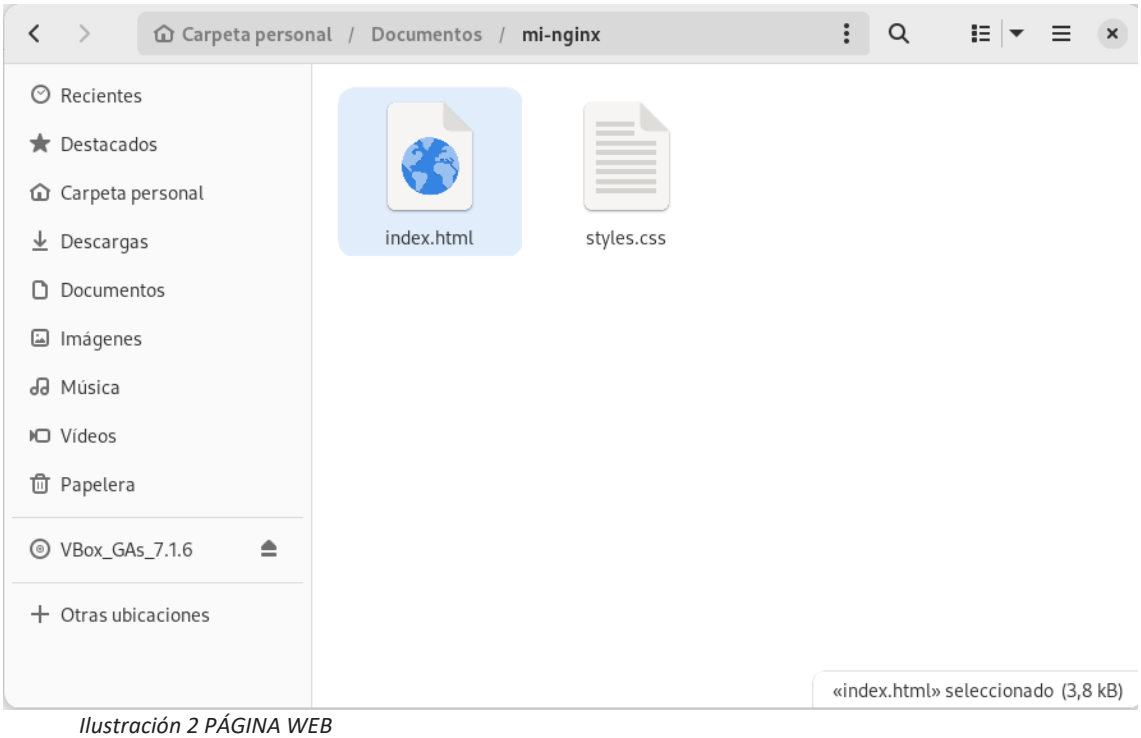

Ilustración 2 PÁGINA WEB

La carpeta mi-nginx, que según la práctica tendría que crearla en C:/ProgramFiles/Docker. Como ya comprenderemos, yo no puedo crearla en esa misma dirección. Tampoco lo voy a usar como excusa, decidí no enredar demasiado y dejarla así mismo, para evitar posibles consecuencias, aunque tuvo algunas que no puede solucionar.

Siguiendo la actividad, debo de ejecutar un comando, que me preparará mi contenedor de nginx gracias a Docker, incluyendo mi archivo html. Hay va a venir el problema.

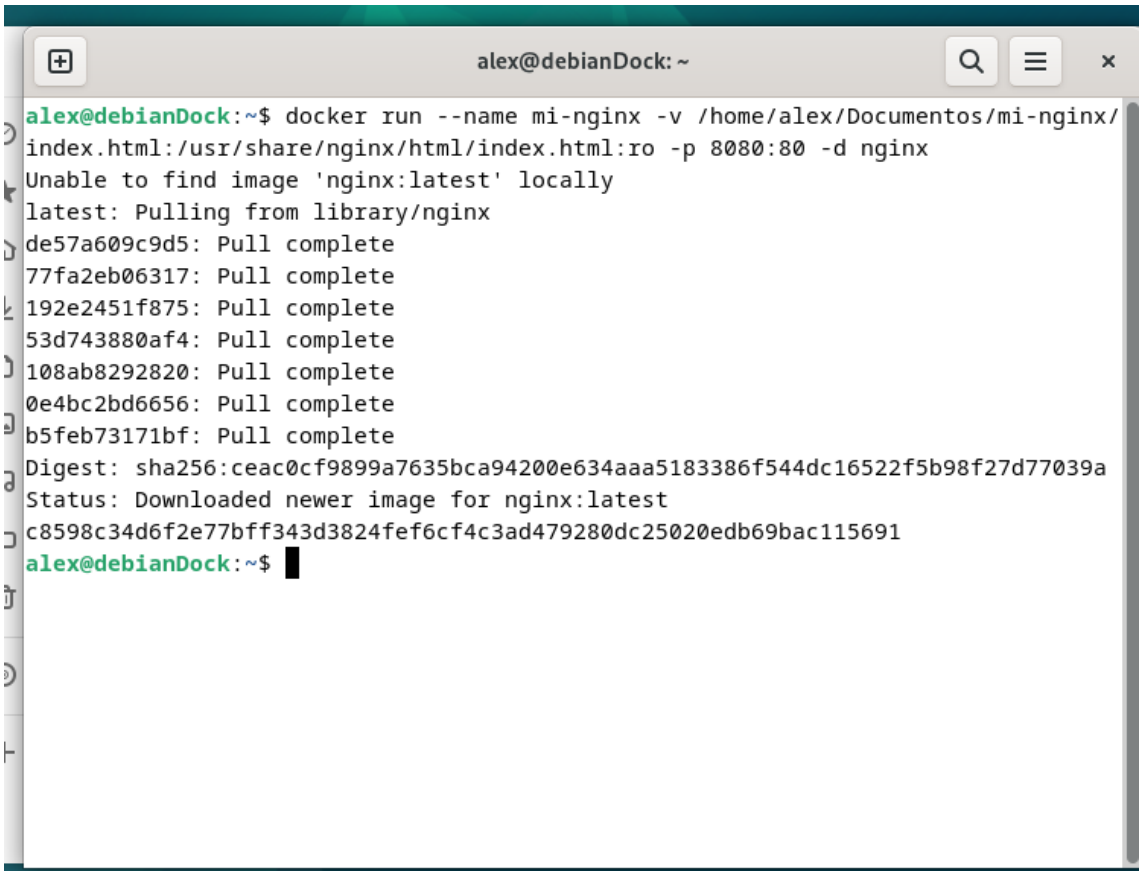

Ilustración 3 CREANDO CONTENEDOR NGINX

Una vez creo el contenedor, si nos fijamos en el comando usado para crearle, solo apunto al archivo html, y no al css. Esto hace que sólo se añada a mi contenedor ese archivo. Todo esto, por la forma en la que se organizan los directorios en Linux, por no tocar demasiadas cosas no vaya a ser que pasen cosas anómalas y por pensar que con este comando incluía toda la carpeta donde yo tenía mi página web.

De seguro existe una forma en la que yo lo puedo tener completo de la forma en la que se especifica en la actividad, pero, por esa serie de motivos, he decidido dejarlo así.

Una vez hecho y aclarado este paso, vamos a ver el resultado y a seguir con los pasos siguientes.

Ahora debo de comprobar el estado de este contenedor.

Usando el comando Docker ps puedo ver un listado de los contenedores creados que están en funcionamiento. Al usarlo la primera vez no va a mostrar ninguno y puede dar la sensación de que hemos creado mal nuestro contenedor. Sin embargo, para asegurarse debemos de usar un comando ligeramente diferente.

Acabo de darme cuenta de que no he hecho una captura exacta de ese momento, pero tengo otra que nos sirve para el mismo ejemplo.

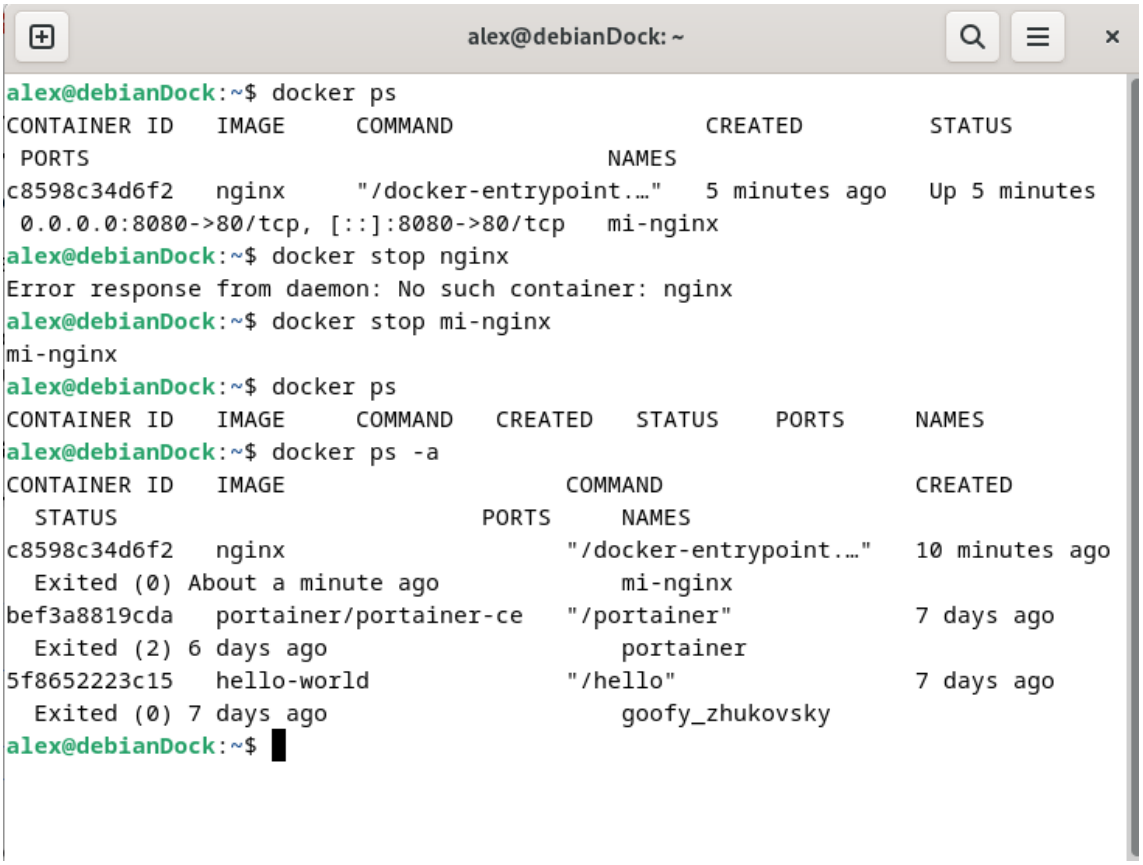

Ilustración 1 DOCKER PS -A

Como vemos en la imagen, si uso el comando Docker ps, no aparece nada. Es el ultimo comando introducido. Ahora bien, si usamos el comando **Docker ps -a**, nos aparece un listado de todos los contenedores creados, tanto aquello que están activos como los que están parados. Con esto podemos comprobar todos los que tenemos.
En este caso vemos el contenedor mi-nginx, portainer, que ya explicaré después lo que es y googy_zhukovsky, que es un contenedor de prueba que simplemente al activarlo te permite ver hello-world por pantalla.

Ahora lo que debemos hacer es levantar nuestro contenedor nginx para ver si todo funciona correctamente.

Para levantar y poner en funcionamiento un contenedor debemos de usar el siguiente comando:

Docker start “nombre-de-mi-contenedor”

En este caso, el comando completo debería ser Docker start mi-nginx.

Pero antes de ver el resultado en el navegador, a colación de lo explicado antes, vamos a listar los contenedores activos con Docker ps. Sin -a.

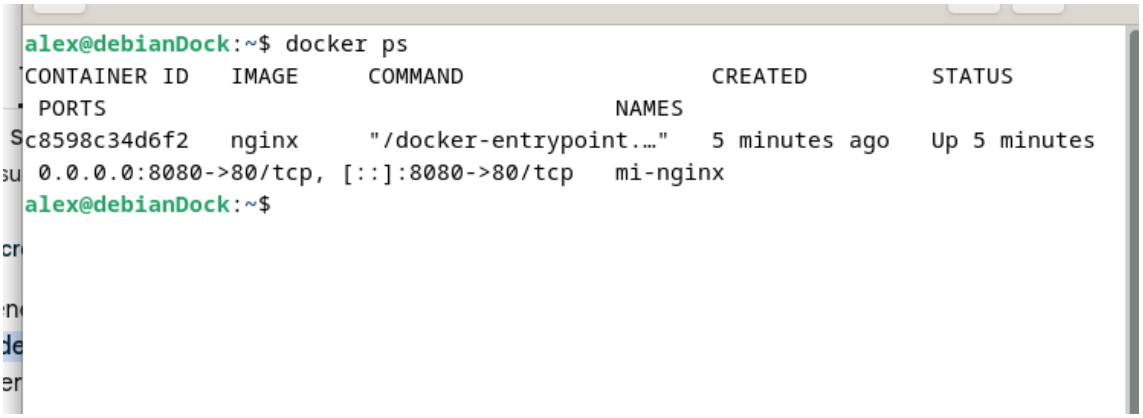

Ilustración 2 Listando contenedores activos

Como vemos en la imagen, una vez ejecutado Docker start, si solicitamos un listado simple de contenedores, vemos como aquel que nosotros requerimos está levantado. Vamos a comprobar el resultado en el navegador.

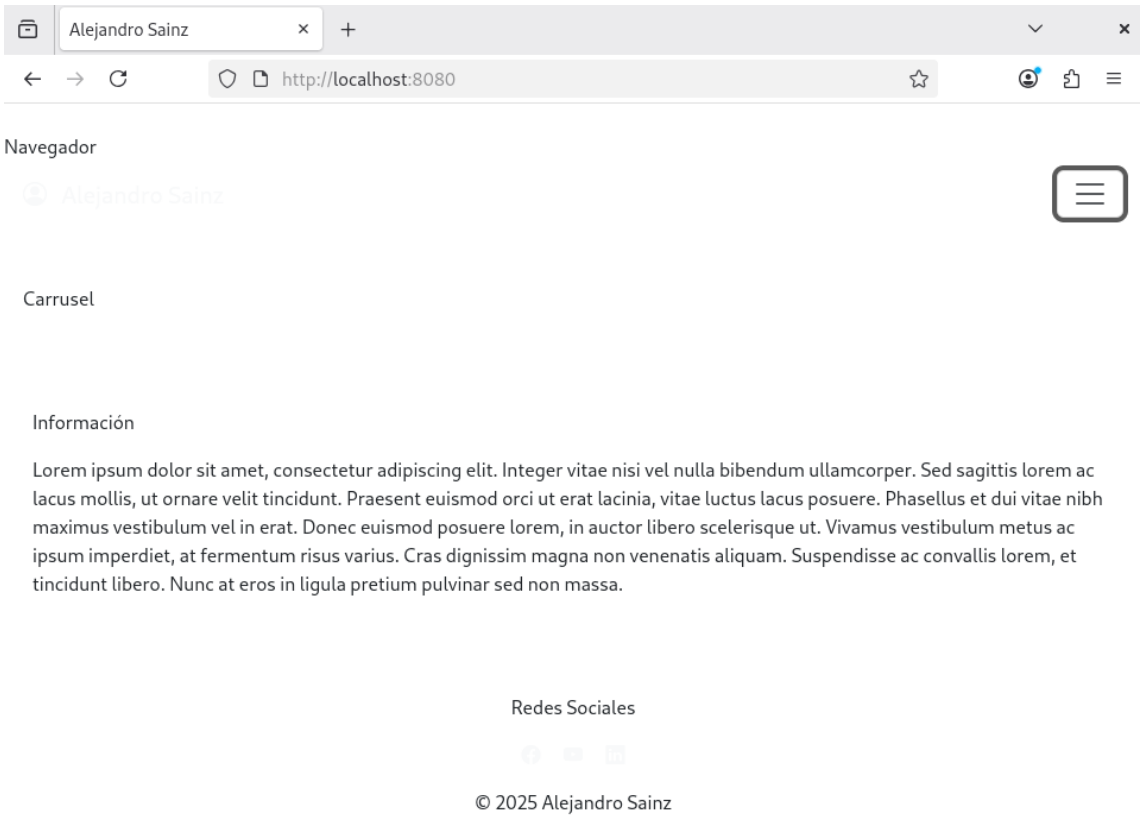

Ilustración 3 Nginx arriba

Si accedemos a http://localhost, a veces tendremos que indicar el puerto, pero no siempre, vemos lo que sería mi página web. Como ya he indicado antes, sin css, por aquel error que cometí al crear el contenedor o al no buscar la forma de copiar el css en la carpeta destino.

Salvando esto último, el funcionamiento es correcto.

A partir de aquí debemos de realizar una serie de operaciones. En este caso, parar, reiniciar, ver los logs y borrar un contenedore en cuestión.

Vamos a ello paso a paso.

Arrancar el contenedor ya lo he explicado en el punto anterior. No tiene mucho misterio.

Además, en el caso de Linux (no tengo claro que en Windows sea igual) la terminal no te da más información de que se ha iniciado un contenedor que mostrar el nombre del contenedor en la línea de comandos.

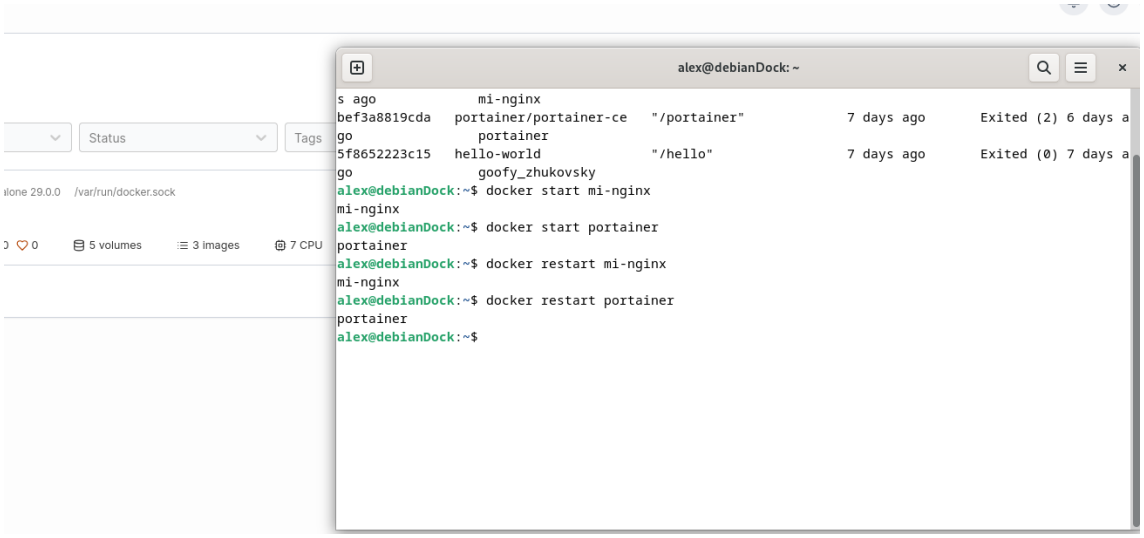

Ilustración 4 Diferentes operaciones

Vamos a cubrir dos de las operaciones con esta imagen. Después de ejecutar un Docker ps -a, como vemos en la imagen anterior, se nos lista los diferentes contenedores disponibles. Como vemos en la imagen, levanto dos contenedores con Docker start, nginx y portainer.

Como ya he indicado antes, la única notificación que recibimos es que, en el prompt de la terminal, se nos indica el nombre del contenedor levantado. No voy a incluir más capturas de pruebas en el navegador, es lo mismo que ya he mostrado antes.

Después, prueba a ejecutar el siguiente comando:

Docker restart “nombre-de-mi-contenedor”

Este comando reinicia el contenedor indicado. Sin dramas, y, además, sin tardar mucho.

De nuevo, si mostrase una captura del navegador, sería lo mismo. Con una pequeña diferencia en mi caso. La web se mostraría igual, no tiene más misterio. Pero Portainer, si tuviésemos una sesión iniciada, nos mostraría la página de login. Más adelante mostraré capturas de Portainer.

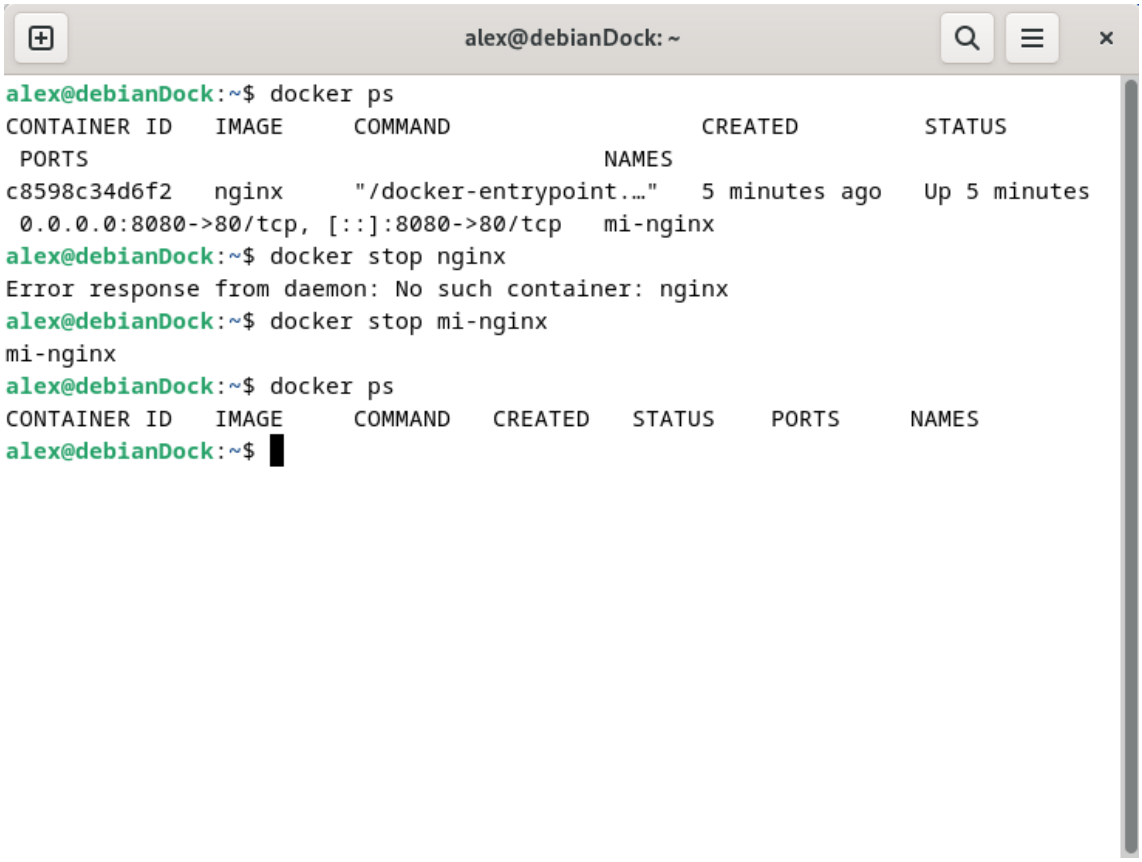

Ilustración 5 Parando un contenedor

La siguiente operación tampoco es muy complicada. Se entiende por si sola. Vamos, que no es neurocirugía.

Para parar un contenedor usamos el siguiente comando:

Docker stop “nombre-de-mi-contenedor”

Como vemos en la imagen, si somos capaces de escribir correctamente el nombre que le hemos dado al contenedor y no el nombre de la imagen, el contenedor se para. Si hiciésemos una llamada a la dirección IP de nginx nos devolvería una página no encontrada muy vistosa.

Ahora, que este ha sido otro de mis errores, voy a borrar un contenedor. Suerte que tenía dos y pude mostrar después los logs de ese contenedor.

Para borrar un contenedor, lo primero que debemos de hacer es pararlo, con Docker stop. Una vez detenido, procedemos a borrarlo.

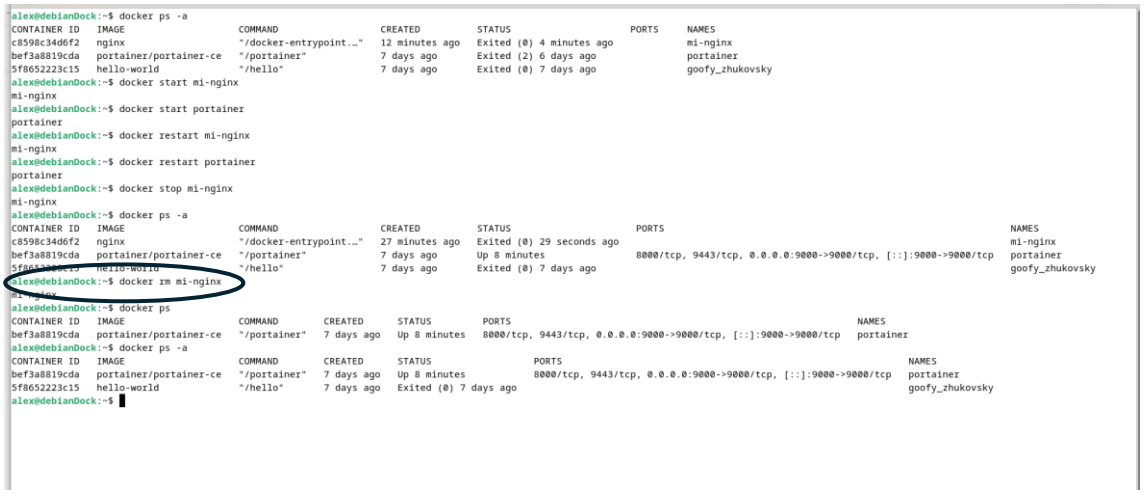

Ilustración 6 Tendría que borrar la terminal antes de hacer nuevas capturas

Ya se que no se ve demasiado bien, más que nada por el tamaño. Pero nos vale.

Una vez parado un container debemos ejecutar el siguiente comando:

Docker rm “nombre-de-mi-contenedor”

Con esto nos borrar el contenedor indicado. Para que se viese bien, por eso el tamaño de la captura, ejecuto Docker ps y Docker ps -a. Con esto podemos comprobar que el contenedor mi-nginx ya no está entre nosotros.

Ahora bien, debemos de mostrar los logs de un contenedor. Y yo, que no me di cuenta, había borrado ya mi contenedor de nginx. Menos mal, que todavía me quedaba el de portainer.

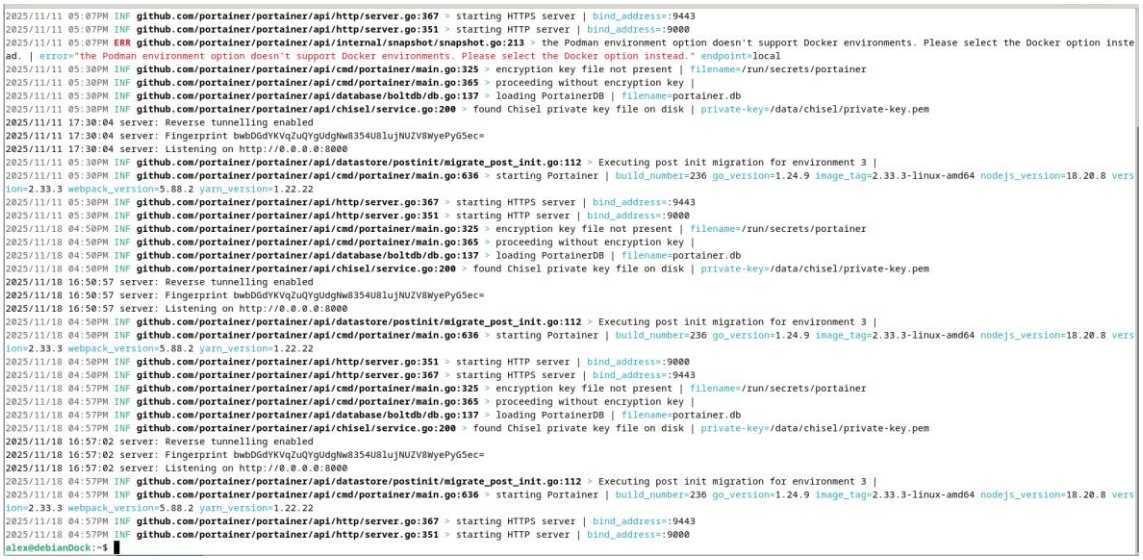

Ilustración 7 Logs de un contenedor.

Este es otro comando muy sencillo. La verdad es que la parte básica de Docker es muy sencilla e intuitiva.

El resultado de la imagen anterior se obtiene con el siguiente comando:

Docker logs “nombre-de-mi-contenedor”

Si queremos los logs en tiempo real:

Docker logs -f “nombre-de-mi-contenedor”

También existe una versión que es para ver las últimas líneas que no probé. Seria con:

Docker logs –tail “N” “nombre-de-mi-contenedor”

En este caso sustituiríamos N por el número de líneas que queramos ver.

Y hasta aquí todo lo relacionado con lo que se pedía en esta parte de la práctica.

Ahora bien, dije que iba a hablar de Portainer. No muy en profundidad, pero un poco. Portainer es una de las opciones que tenemos en Linux para tener un GUI de Docker. La instalé un poco por curiosidad. Así tenía la opción de ver gráficamente lo que hacía. El problema es que me faltó algo de tiempo para terminar de entender algunas de sus opciones y así poder manejar los contenedores de forma visual y no por terminal. Voy a listar simplemente algunas de las cosas que he hecho y, al final, daré una pequeña explicación.

Ilustración 8 Instalando Portainer

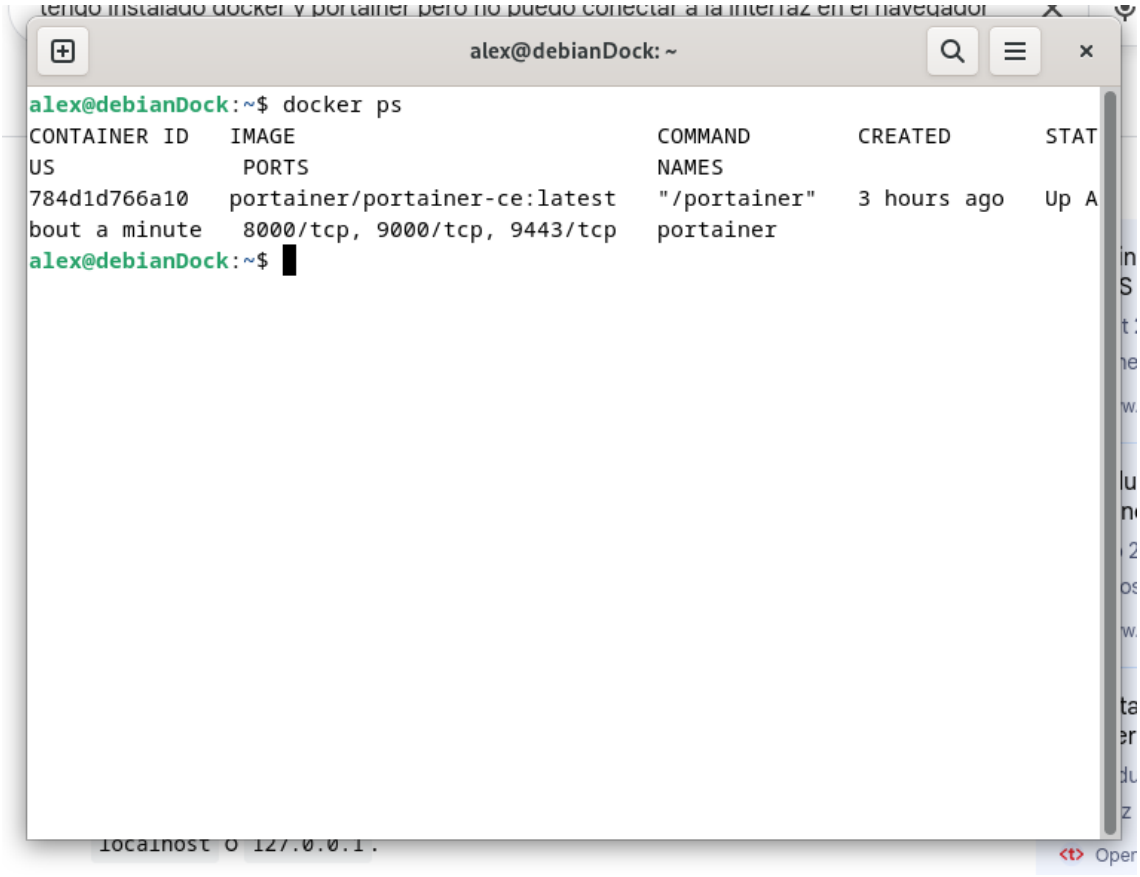

Ilustración 9 Comprobando que Portainer está levantado

```
alex@debianDock:~$ docker port portainer
8000/tcp -> 0.0.0.0:8000
8000/tcp -> [::]:8000
9000/tcp -> 0.0.0.0:9000
9000/tcp -> [::]:9000
9443/tcp -> 0.0.0.0:9443
9443/tcp -> [::]:9443
alex@debianDock:~$
```

Ilustración 10 Listado de puertos de Portainer

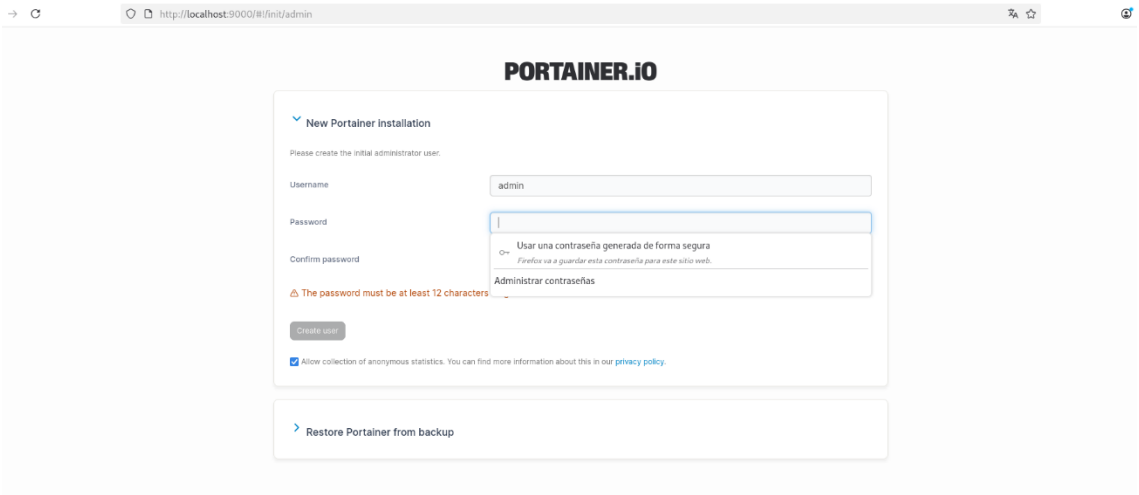

Ilustración 11 Login Portainer

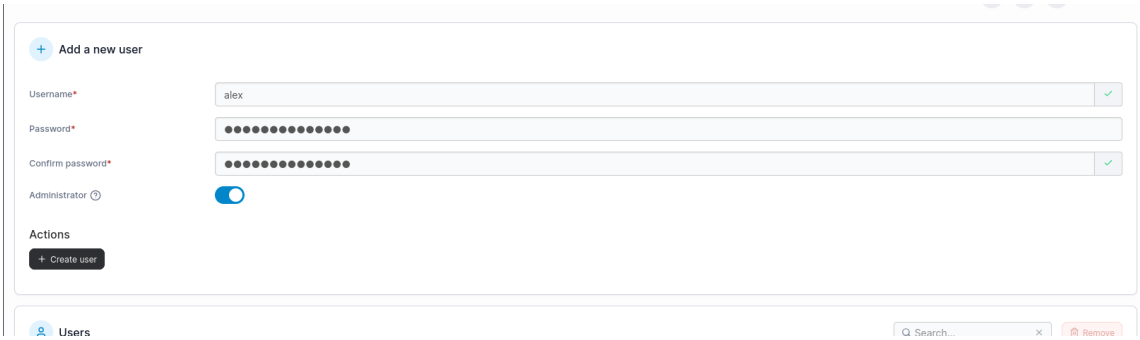

Ilustración 12 Creando usuario

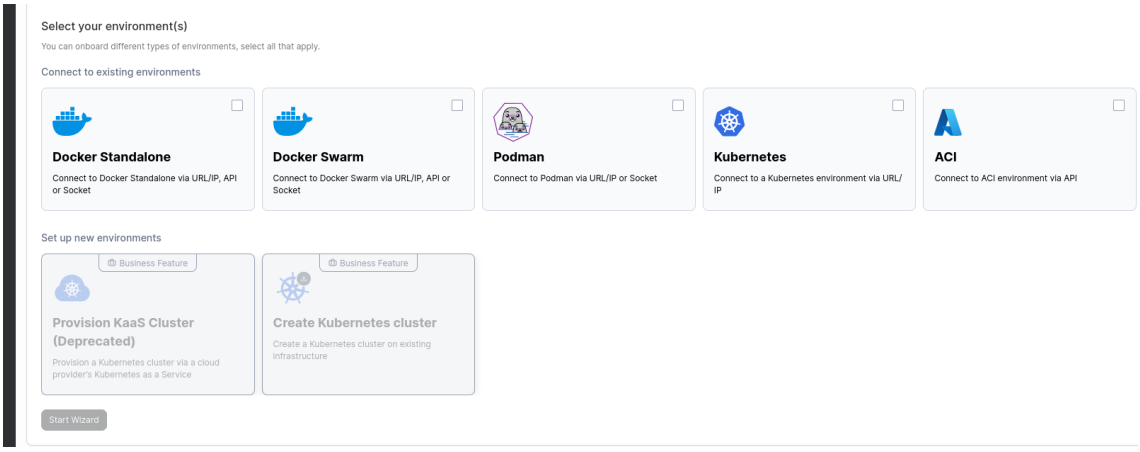

Ilustración 13 Añadiendo Enviroment

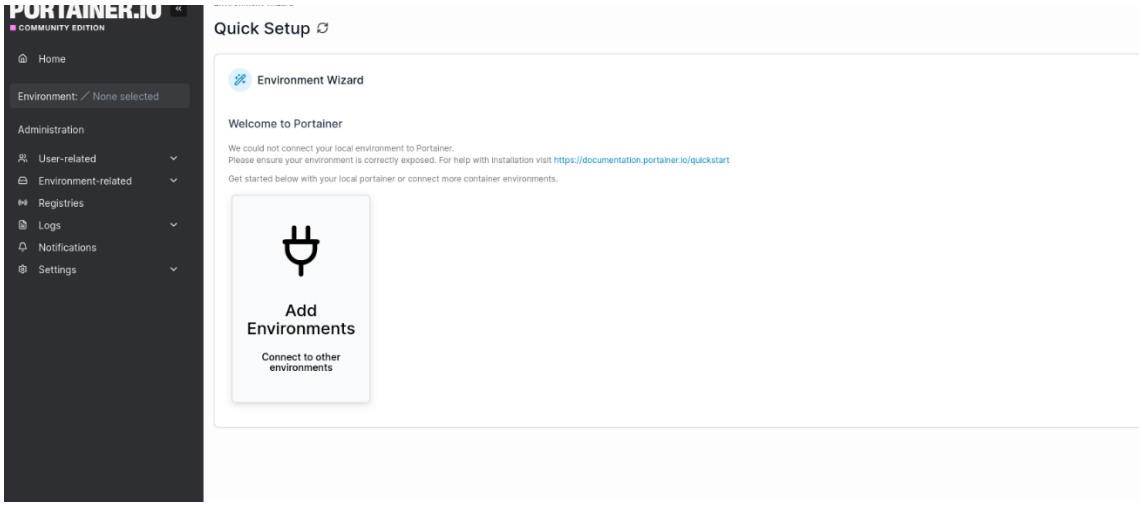

Ilustración 14 Portainer DashBoard

Hasta aquí voy a mostrar. Porque de aquí en adelante no supe desenvolverme.

Una vez hecho todo esto, si seleccionas un enviroment, que yo en mi caso llamé local, te sale un pequeño resumen de los contenedores alojados en local, que es el enviroment que yo seleccione. Mediante iconos te indica, total de contenedore, contenedores levantados, contenedores parados y otra serie de parámetros que yo todavía no entiendo. El problema, que como no he conseguido configurarlo completamente, cada vez que desde esta interfaz gráfica intentaba acceder a la información de los contenedores, el enviroment se paraba, deteniendo a su vez la muestra en tiempo real de la información de contenedores. Pero bueno, fue una buena práctica para conocer algo más.

El problema es que, a veces porque me caliente, a veces por pura temeridad, me meto en jardines en los que no se moverme muy bien. Pero al final, es algo más que conozco.

Una cosa a destacar de esta interfaz, es que no encontré la forma de crear o eliminar contenedores desde ella. No se si no da la opción, o que quizás me faltase algún elemento más de configuración. Me habría gustado llegar a ese punto, porque hasta donde yo leí, se podía hacer. Otra cosa, es que yo no supe encontrar la forma.

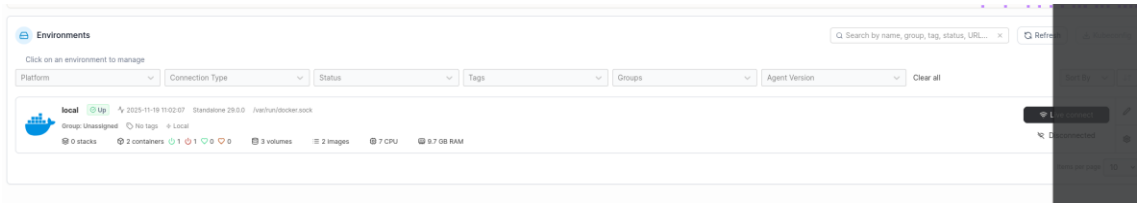

Ilustración 15 Información en Portainer

Así es como se ve la información en Portainer, en este caso, en mi ámbito local.

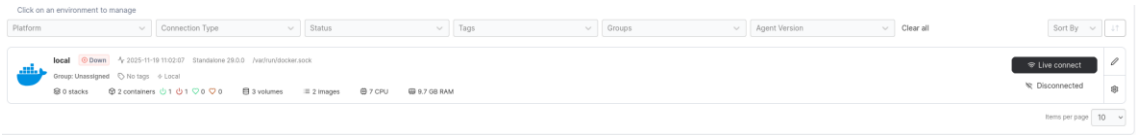

Ilustración 16 Abajo

Pero como bien he dicho antes, en cuanto toco alguno de los iconos inferiores que se refieren a los contenedores, el enviroment se viene abajo. Estaría bien mirarlo más adelante.

3 PREGUNTAS ADICIONALES

¿Qué es Nginx?

Ya hablé sobre el en otro trabajo. Es un servidor web, similar a Apache. Pero tiene ciertas diferencias. Es muy capaz y óptimo con altas cargas de tráfico, pero no es tan dinámico ni configurable o extensible como Apache. Cada vez que se le quiere añadir alguna funcionalidad debemos de compilarlo entero.

Su funcionamiento, hablando de nuestro nivel de trabajo, es muy similar a Apache. En Apache, para desplegar una web, debemos de alojar los ficheros en la carpeta htdocs. En Nginx, si no recuerdo mal, era en http. Una vez hecho esto, con hacer una llamada a la IP correspondiente, en nuestro caso localhost, se nos muestra la página alojada.

Diseccionando comandos.

docker run --name mi-nginx -v ${PWD}\index.html:/usr/share/nginx/html/index.html:ro

-p 8080:80 -d nginx

Que hace cada parte de este comando.

Docker es la instrucción principal, para indicar que programa la ejecuta.

Run indica que se debe de crear o ejecutar el contenedor y la imagen que le vamos a indicar a continuación.

--name mi-nginx Aquí estamos indicando el nombre que le vamos a dar a nuestro contenedor.

-v Volumen. Quiere decir que vamos a crear un volumen y que ese volumen lo vamos a copiar o ejecutar dentro de nuestro contenedor.

${PWD}/index.html: Aquí tenemos dos partes. ${PWD} quiere decir que esa parte es igual al valor obtenido tras ejecutar el comando PWD, es decir, donde estoy. Con esto sabemos que para ejecutar este comando debemos de estar situados en el directorio deseado para que el valor de PWD sea igual a la dirección donde está alojado el archivo index.html.

:/usr/share/nginx/html/index.html : Aquí es donde se aloja nuestro contenedor y donde vamos a conectar el archivo html. Es decir, se va a crear una copia, o simplemente se referencia que aquí se va a incluir el archivo index.html que indicamos anteriormente.

:ro : Read only. El archivo sólo se va a poder leer, pero no modificar. Por seguridad, para que no se produzca ninguna modificación inesperada.

-p 8080:80 : Con esta parte se exponen los puertos que va a usar nuestro contenedor. Podríamos indicarle más si quisiésemos. Cada vez que pongamos uno tendríamos que añadir –p “puerto:puerto”.

-d : Defered. Quiere decir que el contenedor se va a ejecutar en segundo plano.

Nginx : la imagen de Docker que se va a descargar y a instalar/desplegar en nuestro contenedor.

¿He tenido que instalar Nginx en mi ordenador?

No. La imagen de Nginx y sus dependencias o todo aquello que necesite para funcionar se instala o se despliega en el contenedor. Con esto lo que se consigue es que Nginx esté aislado no pudiendo afectar a mi equipo, ni lo que pase en mi equipo afectarle a él. No toca nada del registro de mi ordenador al instalarse, ni modifica nada en mi sistema de archivos. Y, una vez lo elimino, no queda rastro de él, ya que es como eliminar una carpeta que yo he creado.