Ejercicio IV: Docker

Nicolás A. Ortega Froysa

17/11/2021

Índice

1. Hoja De Control Del Documento

Cuadro 1: Documento/Archivo

Fecha Última Modificación	17/11/2021	**Versión/Revisión**	v01r04
Fecha Creación	10/11/2021		
Fecha Finalización	17/11/2021		

Cuadro 2: Registro De Cambios

Versión/Revisión	**Página(s)**	**Descripción**
v01r01	Todas	Creación y elaboración del documento.
v01r02	3-4	Detallado de instalación y primer docker.
v01r03	4-5	Finalizar sección de «Primer Docker».
v01r04	5-7	Añadir secciones Apache y Conclusión

Cuadro 3: Autores Del Documento

Apellidos, Nombre	**Curso**
Ortega Froysa, Nicolás Andrés	1

Preparado	**Revisado**	**Aprobado**
Ortega Froysa, Nicolás Andrés		

2. Instalación de Docker

Para poder instalar Docker podemos instalarlo en una máquina de arquitectura x86–64, ARM, s390x, y ppc64le, y de sistema operativo Linux, Windows, o macOS. Para esta práctica se usará una distribución de Linux denominada ArchLinux en la arquitectura ARM (aunque realmente los comandos serán los mismos en cualquier sistema de ArchLinux). ArchLinux se puede instalar siguiendo las instrucciones de la guía de instalación de ArchLinux en su Wiki.[1]

Cuando ya se tiene ArchLinux instalado, podemos instalar Docker fácilmente desde la línea de comando con el comando `pacman -S docker`. Esto instalará el programa de docker y todas sus dependencias, mas siendo ArchLinux no se inicializa automáticamente el servicio de Docker, así que es necesario iniciarlo manualmente. Suponiendo que estamos en un sistema que use systemd, podemos hacer esto mediante el comendo `systemctl start docker`. Si queremos que se inicialice al iniciar el sistema operativo, podemos ejecutar adicionalmente el comando `systemctl enable docker`.

3. Primer Contenedor: Hello World

Al tenerlo corriendo de fondo ya podemos empezar a tratar con él. Esto se hace por medio del comando `docker`. Para usarlo es necesario ejecutar como superusuario, que se puede hacer o cambiando al usuario `root`, o añadiendo `sudo` antes de todos los comandos. También existe la opción de añadir el usuario a un grupo especial, `docker`. Esto se puede hacer por medio del comando `usermod -aG docker <usuario>`, donde `<usuario>` es el nombre del usuario actual que queremos añadir al grupo. Este comando también se tiene que ejecutar como superusuario (y por lo tanto debe de ser `root` o usar `sudo`).

Si ejecutamos el comando `docker image ls` podemos ver qué imágenes tenemos disponibles localmente en la máquina, que recién instalado no debería de haber ninguna. También podemos usar `docker ps -a` para ver qué contenedores tenemos corriendo. Esto nos servirá luego para gestionar los contenedores y las imágenes locales.

Ahora, para descargar e inicializar un contenedor lo hacemos directamente pidiendo que Docker corra la imagen que nos interesa. Para empezar usaremos la imagen `hello-world`. Para hacer esto ejecutamos el comando `docker run hello-world` y nos debe de salir un mensaje indicando que se ha ejecutado correctamente (figura 1).

[1] `https://wiki.archlinux.org/title/Installation_guide`

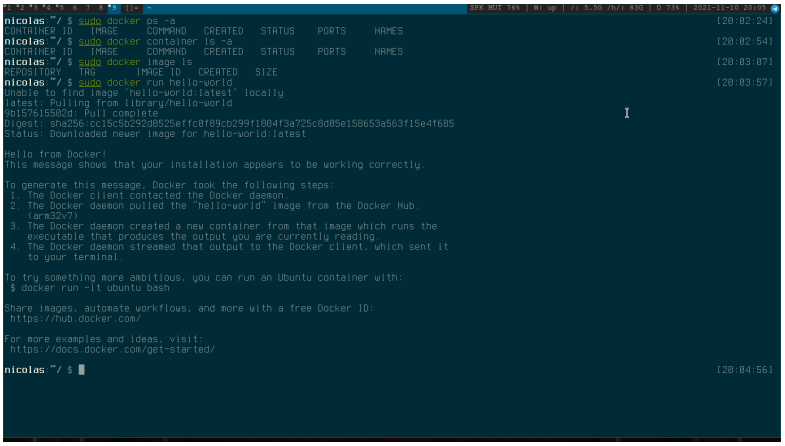

Figura 1: Ejecución del container `hello-world`.

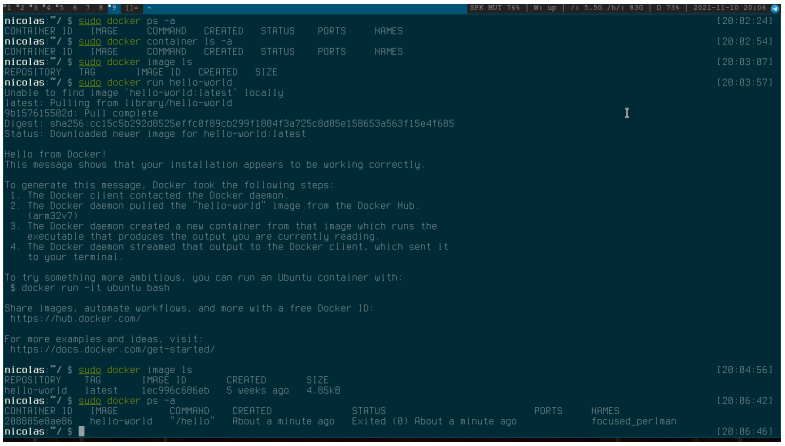

Figura 2: Lista de contenedores e imágenes.

Ahora, si volvemos a ejecutar el comando `docker ps -a`, nos mostrará que existe un nuevo contenedor en ejecución. Además, al ejecutar `docker image ls` nos muestra que se ha descargado además la imagen `hello-world`. Si miramos de nuevo la lista de contenedores corriendo, notaremos que tiene un atributo del *nombre* (`NAMES`). Aquí aparecerá el nombre que tiene el contenedor que podemos usar para referirnos a él, además del `CONTAINER ID`. El nombre del contenedor viene generado por una palabra y un apellido aleatorio, en nuestro caso `focused_perlman` (figura 2). Éstos luego los podemos usar para manejar los contenedores. E.g. podemos borrar un contenedor con el comando `docker rm <name>` ó `docker rm <id>`. Este nombre lo podemos asignar en la creación del contenedor (el subcomando `run`) añadiendo la opción `--name=<name>`. En cuanto a la ID del contenedor, no es necesa-

rio especificar la ID entera (que puede ser larga y fácil de confundir) sino simplemente con poner los primeros caracteres de la cadena que sean únicas basta.

4. Contenedor de Servidor Apache

Los contenedores se suelen usar no para correr aplicaciones simples que te dan un resultado, como una calculadora, sino para proveer servicios en entornos controlados. Esto es muy útil para los servidores, que tienen que proveer servicios de una forma estable.

Para poner a prueba esto, veremos cómo instalar un servicio de Apache (servidor web) en un contenedor. Para el ejercicio se ha propuesto el uso de la imagen `bitnami/apache`, mas este imagen no está disponible para la arquitectura de mi ordenador (ARMv7), así que tuve que buscar una alternativa, y encontré la imagen oficial de Apache: `httpd`.[2]

Así que para crear este contenedor usamos el comando de antes: `docker run -d -p 8888:80 httpd`. Veamos lo que hace cada elemento de este comando:

- `run`: como vimos antes, corre el contenedor, descargando y creándolo si no se ha creado ya.
- `-d`: un *detach* que permite correr el contenedor de fondo mientras la línea de comando queda libre para correr otros comandos.
- `-p 8888:80`: publicar el puerto 80 del contenedor al puerto 8888 del anfitrión. También se puede usar `-P` que asignará un puerto aleatorio.
- `httpd`: el nombre de la imagen que queremos usar en nuestro contenedor.

Esto nos creará el nuevo contenedor y lo pondrá en funcionamiento. Si hemos usado la opción `-p 8888:80` entonces deberíamos poder acceder al servidor por medio de nuestro navegador haciendo una petición a `localhost:8888`. Si por el otro lado hemos usado `-P` nos dirá el puerto y reemplazamos el `8888` de la dirección anterior por el número real del puerto (figura 3).

Aunque ahora hemos podido crear un servicio de web con nuestro contenedor, la pregunta sería cómo cambiar el contenido. Lo más fácil sería simplemente cambiar el archivo que hay dentro del contenedor en sí. Esto lo hacemos escribiendo nuestro propio archivo HTML, y después copiándolo

[2] `https://hub.docker.com/_/httpd`

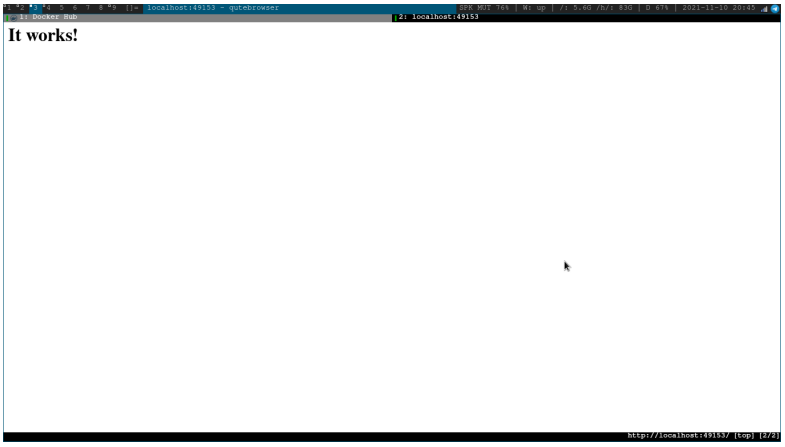

Figura 3: Acceso al servicio apache del contenedor.

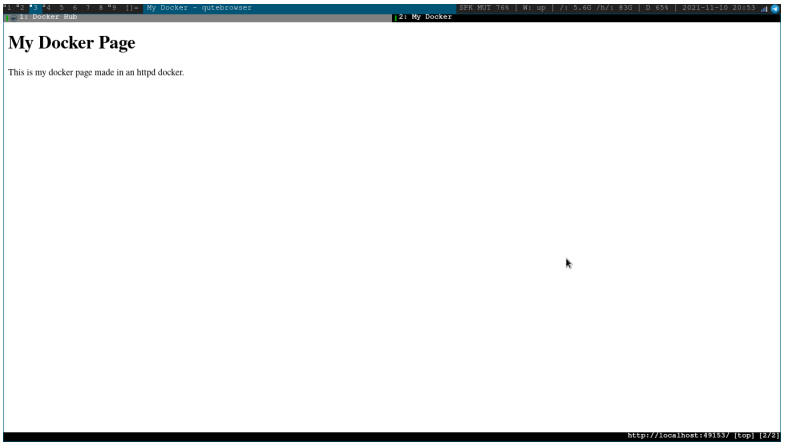

Figura 4: Nueva página en el contenedor.

a la dirección local dentro del contenedor. Esto se hace por medio del comando `docker cp index.html <name>:/usr/local/apache2/htdocs/`. Si ahora volvemos a hacer una nueva petición a la página, y nos sale la nueva página que hemos creado (figura 4).

5. Conclusión

Realmente hay más capacidades que tiene el Docker, especialmente para facilitar el copiar archivos del sistema anfitrión al sistema huésped. Aún así, se puede ver que tiene gran potencial. Los docker permiten crear servicios dentro de un entorno controlado, lo cual asegura más estabilidad – algo vital para

cualquier tipo de sistema en producción. También parece ser una herramienta con mucha potencial en cuanto a programación para dispositivos empotrados o lo también lo que se denomina *cross-compilation*, gracias a tener un entorno limpio y sin binarios innecesarios.

6. Derechos de Autor y Licencia