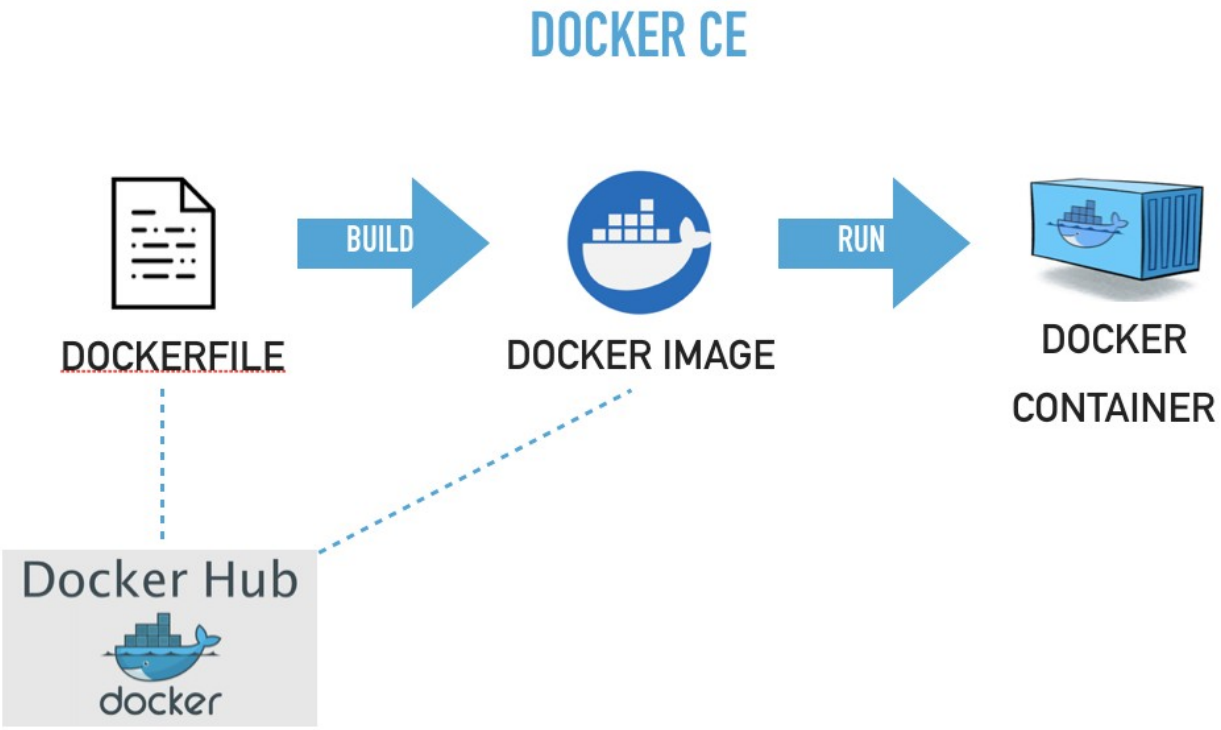
Actividad 11-07: Puesta en marcha de un contenedor docker con Dockerfile



ÍNDICE

1. Introducción

En el presente documento se muestra cómo instalar un contenedor Docker mediante una imagen Docker obtenida de un Dockerfile.

Para el ejemplo, voy a escribir mi propio Dockerfile. Por razones de sencillez, va a consistir en un script sencillito en el lenguaje de programación Ruby. El sistema operativo que se utilizará a lo largo de toda la actividad es Ubuntu 20.04 LTS.

Un Dockerfile es un fichero de texto que Docker analiza y ejecuta secuencialmente, línea a línea desde el principio. Básicamente, está diseñado para que Docker sepa cómo construir la imagen Docker deseada. El comando para ejecutar un Dockerfile es → *docker build*

2. Descripción del Contenedor

El contenedor que voy a crear y utilizar parte de la imagen Docker oficial de Ruby, un lenguaje de programación dinámico, orientado a objetos, de propósito general y código abierto, etc. En el contenedor voy a añadir un fichero con extensión *.rb* que contiene un sencillo script para imprimir por pantalla una cadena de caracteres. En vez de escribir el clásico “Hello World”, voy a utilizarlo para quejarme un poco de la vida, puesto que originariamente iba a hacer todo este tutorial con una imagen Docker de MariaDB, pero mi conocimiento no ha bastado para arreglar los extraños errores que me iban surgiendo.

¡Allá vamos!

3. Instalación y puesta en marcha del contenedor con un Dockerfile

Mediante la consola de comandos de mi máquina Ubuntu, he generado la siguiente configuración previa:

```
alberto@maq-docker-Alberto:~$ cd Desktop/ruby_dockerfile/
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$ ls
Dockerfile  hello_world.rb
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$
```

(Un directorio en el que he generado un fichero Dockerfile y el script de Ruby; hello_world.rb).

Mi Dockerfile es muy sencillo, tiene la siguiente pinta:

```
#DockerFile creado para una actividad de clase
#-----------------------
FROM ruby:latest
MAINTAINER Alberto_FdezPalacios_Aquino
RUN /bin/sh -c 'echo "***BIENVENIDO AL DOCKERFILE DE LA ACTIVIDAD ENTREGABLE***"'
COPY . .
CMD ["ruby","hello_world.rb"]
~
```

Lo que hago es descargar la imagen más actualizada disponible de Ruby, firmar como el autor del Dockerfile, ejecutar un simple comando que lanza un mensaje de bienvenida (lo he hecho para practicar la diferencia entre RUN y CMD en el Dockerfile, que es la siguiente):

RUN → Se ejecuta cuando creas la imagen a partir del Dockerfile

CMD → Se ejecuta cuando creas una instancia de la imagen, es decir, cuando haces el docker run

La línea *COPY . .* → la utilizo para copiar todo lo que hay en el directorio actual de fuera del contenedor, dentro del contenedor (esto tiene aplicaciones prácticas que sirven para añadir ficheros, programas, o incluso proyectos enteros dentro de un contenedor Docker al momento de ejecutar el comando *docker run*). De esta manera es como estoy pasando el fichero con el script de ruby, que se encuentra en el directorio actual (PWD).

La última línea, *CMD [“ruby”, “hello_world.rb”]* → Se encarga de ejecutar el script de Ruby, siguiendo una sintaxis que manda los strings que he escrito entre [] a la terminal.

El fichero *hello_world.rb* imprime una retahíla de palabras que utilizo para quejarme de todo:

```
Acompáñame a presenciar esta triste historia:\n\n Me he rendido. Llevo todo el
día (mañana y tarde) intentando hacer funcionar mi contenedor de MariaDB, pero
```

```
me ha resultado imposible. Creo que me he acercado bastante, pero me he dado cuenta de que no sólo fallaba en Docker, sino también en mi propia máquina virtual de Ubuntu. Es un error que intentaré preguntar en clase...\n La cuestión es que he acabado haciendo este sencillo script en Ruby para paliar mi desesperación desmedida y hacer funcionar mi Dockerfile.\n\n Al menos, he aprendido: \n - A escribir mi propio Dockerfile. \n - A moverme con soltura con los comandos Docker. \n - A hacer funcionar los contenedores de Docker que no están fastidiosamente estropeados... \n\n\n ¡Y ENCIMA ESTE ME HA FUNCIONADO A LA PRIMERA!
```

Comencemos con la parte que de verdad interesa del presente documento. Para ejecutar el Dockerfile y obtener la imagen, tengo que aplicar el comando *docker build*.

Pequeño recordatorio: puedo utilizar el comando *docker* sin añadir *sudo* delante porque pertenezco al grupo docker, lo que me permite utilizar Docker sin elevar mis permisos:

```
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$ docker build -t ruby_dockerfile .
Sending build context to Docker daemon  3.584kB
Step 1/5 : FROM ruby:latest
latest: Pulling from library/ruby
6aefca2dc61d: Pull complete
```

(Podemos observar como se van cumplimentando los pasos que hemos especificado en nuestro Dockerfile).

```
Step 2/5 : MAINTAINER Alberto_FdezPalacios_Aquino
 ---> Running in 08581d6b878f
Removing intermediate container 08581d6b878f
 ---> 3cdfd46d752c
Step 3/5 : RUN /bin/sh -c 'echo "***BIENVENIDO AL DOCKERFILE DE LA ACTIVIDAD ENTREGABLE***"'
 ---> Running in 751f5725cbf1
***BIENVENIDO AL DOCKERFILE DE LA ACTIVIDAD ENTREGABLE***
Removing intermediate container 751f5725cbf1
 ---> 234313f25491
Step 4/5 : COPY . .
 ---> fbb73ff40529
Step 5/5 : CMD ["ruby","hello_world.rb"]
 ---> Running in 46e5a11a8ffc
Removing intermediate container 46e5a11a8ffc
 ---> c19c3d668c89
Successfully built c19c3d668c89
Successfully tagged ruby_dockerfile:latest
```

(Aquí podemos ver que el proceso ha culminado con éxito).

NOTA: el modificador *-t* que he añadido en el comando *docker build* es simplemente para añadirle una etiqueta a la imagen que voy a generar (un nombre, básicamente). Por otro lado, el punto del final (.) sirve para que Docker busque automáticamente el Dockerfile dentro del directorio en el que me encuentro. Por eso lo he llamado así, sin extensión, y sin tener más de un Dockerfile dentro del mismo directorio.

Si yo ahora intento ejecutar el script, veremos que da error:

```
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$ ruby hello_world.rb
bash: /usr/bin/ruby: No such file or directory
```

Esto sucede porque de momento SÓLO he creado la imagen Docker, pero no he instanciado todavía dicha imagen, ni mucho menos estoy dentro del contenedor como para poder usar Ruby, por lo que lo que acabo de hacer es intentar utilizarlo en una máquina que no lo tiene instalado (esto no es demostración de mucho, puesto que habría funcionado si tengo Ruby instalado en la Ubuntu, pero servía para ejemplificar).

Para que funcione DENTRO del contenedor Docker, lo hago de la siguiente manera:

```
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$ docker run -it --name ruby_prueba1 ruby_dockerfile
Acompáñame a presenciar esta triste historia:

 Me he rendido. Llevo todo el día (mañana y tarde) intentando hacer funcionar mi contenedor de MariaDB, p
 acercado bastante, pero me he dado cuenta de que no sólo fallaba en Docker, sino también en mi propia má
ntaré preguntar en clase...
 La cuestión es que he acabado haciendo este sencillo script en Ruby para paliar mi desesperación desmedi

 Al menos, he aprendido:
 - A escribir mi propio Dockerfile.
 - A moverme con soltura con los comandos Docker.
 - A hacer funcionar los contenedores de Docker que no están fastidiosamente estropeados...


 ¡Y ENCIMA ESTE ME HA FUNCIONADO A LA PRIMERA!
```

Podemos comprobar que efectivamente el contenedor está corriendo en nuestra máquina Ubuntu:

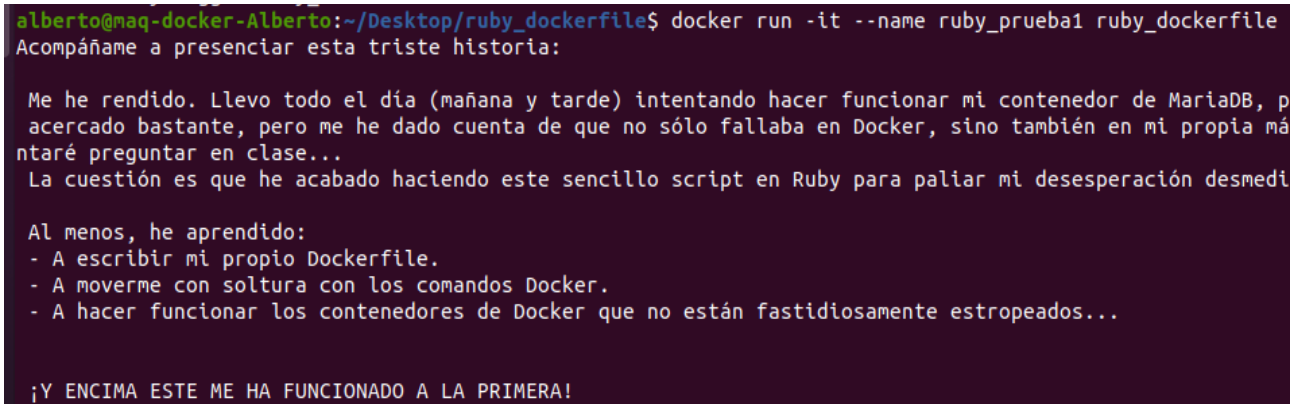
```
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$ docker ps
CONTAINER ID   IMAGE            COMMAND       CREATED          STATUS          PORTS     NAMES
acd1478fbcb1   ruby_dockerfile  "/bin/bash"   39 minutes ago   Up 39 minutes             ruby_prueba1
```

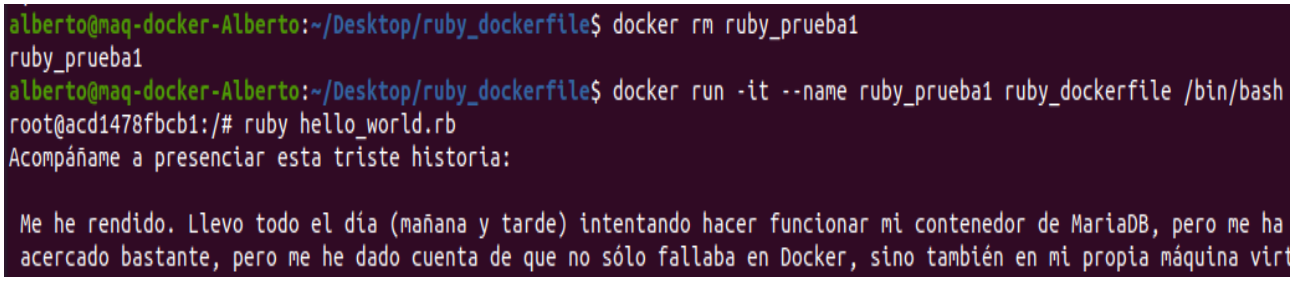
Otra opción es hacer el *docker run* con los modificadores adecuados para que me abra una consola y, en dicha consola, ejecutar el script (pues estaríamos dentro del contenedor, con Ruby instalado).

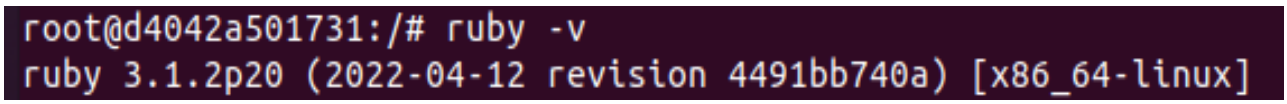
```
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$ docker rm ruby_prueba1
ruby_prueba1
alberto@maq-docker-Alberto:~/Desktop/ruby_dockerfile$ docker run -it --name ruby_prueba1 ruby_dockerfile /bin/bash
root@acd1478fbcb1:/# ruby hello_world.rb
Acompáñame a presenciar esta triste historia:

 Me he rendido. Llevo todo el día (mañana y tarde) intentando hacer funcionar mi contenedor de MariaDB, pero me ha
 acercado bastante, pero me he dado cuenta de que no sólo fallaba en Docker, sino también en mi propia máquina virt
```

4. Verificación del comportamiento del Contenedor

Las anteriores capturas ya sirven de por sí para verificar el comportamiento del contenedor, pero, por criterios de evaluación de esta actividad, las pego aquí de nuevo:







5. Conclusiones

Ha sido duro, pero hemos aprendido más cosas de Docker hoy, ¡genial!

Llegados a este punto, deberías ser capaz de:

- Escribir un Dockerfile sencillo.
- Ejecutar tu Dockerfile (o cualquier otro que hayas encontrado en Internet).
- Instanciar la imagen Docker obtenida mediante el Dockerfile.
- Añadir ficheros y directorios al contenedor que crees mediante la imagen Dockerfile.
- Utilizar apropiadamente el contenedor que tanto esfuerzo te ha llevado poner a funcionar.ç

¡Enhorabuena!

6. Referencias/Bibliografía:

- https://nickjanetakis.com/blog/differences-between-a-dockerfile-docker-image-and-docker-container

- https://docs.docker.com/engine/reference/builder/

- https://geekflare.com/dockerfile-tutorial/

- https://stackoverflow.com/questions/56322341/how-does-the-entrypoint-command-works-here

- https://docs.docker.com/engine/reference/commandline/exec/

- https://pilasguru.gitbooks.io/docker-guia-para-el-usuario/content/chapter03/04crear-dockerfile.html

- https://hub.docker.com/_/ruby

- https://learn.co/lessons/hello-world-ruby#:~:text=Writing%20Code&text=rb%20that%20you%20created%2C%20you,surround%20your%20text%20with%20%22%22%20.&text=puts%20%22Hello%20World!%22