Contenedores

Los contenedores son instancias de las imágenes que hemos creado o hemos descargado que se ejecutan de forma aislada.

Listado

La orden para ver el listado de contenedores del sistema es `docker container ls` o la forma abreviada `docker ps`. Si lo ejecutamos nos dará un listado vacío porque no hay ningún contenedor activo.

Probemos con el parámetro `--all` o `-a`.

```
$ docker container ls -a
CONTAINER ID  IMAGE        COMMAND     CREATED         STATUS      PORTS
NAMES
4bd76e08b07f  wordpress    "docker-…"  11 minutes ago  Exited (0)
peaceful_murdock
69a3c34c224d  hello-world  "/hello"    18 minutes ago  Exited (0)
blissful_goldwasser
```

Estos contenedores están parados y se pueden volver a ejecutar, con el mismo estado que tuviera el sistema de archivos cuando se detuvieron.

Ejecutar comandos dentro de un contenedor

Ya hemos usado `docker run` para crear e iniciar un contenedor. También podemos usar este comando para ejecutar programas que estén dentro del contenedor. Por ejemplo:

```
docker run --name ubuntu_bash --rm -i -t ubuntu bash
```

Info

Las primeras versiones de *Docker* eran más limitadas, respecto a la creación de objetos. Así que salió con comandos como `docker start`, `docker stop`, etc. relacionados con los contenedores. Cuando surgieron más objetos no había consistencia entre los comandos de otros objetos (como `docker volumes ls`) y los de los contenedores.

Así que se ha creado una jerarquía nueva de subcomandos bajo el comando `container` que son equivalentes y se mantienen por compatibilidad:

Antiguo	Nuevo
`docker run`	`docker container run`
`docker start`	`docker container start`
`docker stop`	`docker container stop`
`docker rm`	`docker container rm`
`docker inspect`	`docker container inspect`
`docker exec`	`docker container exec`

No hay más diferencia entre ellos que el nombre.

Pero esta forma de ejecutar cosas, crea un nuevo contenedor. Si queremos ejecutar un comando en un contenedor que ya esté iniciado, debemos usar `docker container exec`.

Ejecuta lo siguiente en otro terminal (no cierres el anterior).

```
docker exec -w /tmp ubuntu_bash touch my_file.sh
```

El parámetro `-w` indica el directorio de trabajo, después indicamos el contenedor donde queremos ejecutar el comando (`ubuntu_bash`) y por último el comando a ejecutar (`touch my_file.sh`).

Si en el primer terminal ejecutamos un listado del directorio tmp:

```
# ls /tmp
my_file.sh
```

Vemos como podemos modificar un contenedor ya iniciado con `docker container exec`.

Pulsa `Control+C` en el primer terminal para cerrar y borrar el contenedor.

Iniciar un contenedor

Con `docker container start` podemos iniciar un contenedor parado:

```
$ docker container start peaceful_murdock
peaceful_murdock
$ docker ps
CONTAINER ID  IMAGE      COMMAND    CREATED         STATUS  PORTS
NAMES
4bd76e08b07f  wordpress  "docker…"  14 minutes ago  Up      0.0.0.0:8080-
>80/tcp  peaceful_murdock
```

Veremos que la web de instalación de *WordPress* está de nuevo disponible. Solo que ahora el contenedor se ejecuta en segundo plano y no lo podemos detener como antes.

Detener un contenedor

Con `docker container stop` podemos detener un contenedor iniciado, indicando su id o su nombre

```
$ docker container stop 4bd76e08b07f
4bd76e08b07f
```

Tip

Podemos hacer referencia a los contenedores por su ID o por su nombre.

Borrar un contenedor

Un contenedor detenido ocupa espacio. Si hemos dejado de necesitar un contenedor podemos borrarlo con `docker container rm`. Igualmente hay que indicar id o nombre.

```
$ docker container rm 4bd76e08b07f
4bd76e08b07f
```

Danger

Hay que tener cuidado al borrar contenedores. Cuando un contenedor se borra se elimina cualquier información que contenga y no esté almacenada en algún lugar externo al propio contenedor.