Prácticas Docker

1. Docker Exec

- Arrancamos un contenedor basado en la imagen NGINX en modo background

```
docker run -d --name nginx1 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
2a72cbf407d6: Pull complete
eccc107d7abd: Pull complete
76aa3935d77c: Pull complete
Digest: sha256:f6e250eaa36af608af9ed1e4751f063f0ca0f5310b1a5d3ad9583047256f37f6
Status: Downloaded newer image for nginx:latest
a0796441c01d02aabb7cf9fff556129c1391589d7f961ff13979d830f7c88527
```

- Comprobamos que está funcionando

```
docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
a0796441c01d        nginx               "nginx -g 'daemon of…"   56 seconds ago      Up 55 seconds       80/tcp              nginx1
```

- Lanzamos algún comando contra el contenedor con “EXEC”. Visualizar los ficheros de /

```
docker exec nginx1 ls /
bin
boot
dev
etc
home
lib
```

```
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

- Podemos lanzar varios comandos al mismo tiempo. Por ejemplo la fecha y nombre del sistema

```
docker exec nginx1 date ;  uname -n
Wed Mar 21 09:21:13 UTC 2018
localhost.localdomain
```

- Podemos lanzar un script a través de la Shell correspondiente pasando el script como argumento.

```
docker exec nginx1 /bin/sh -c  'for i in *; do echo
"ficheroi -->" $i ; done'
ficheroi --> bin
ficheroi --> boot
ficheroi --> dev
ficheroi --> etc
ficheroi --> home
ficheroi --> lib
ficheroi --> lib64
ficheroi --> media
ficheroi --> mnt
ficheroi --> opt
ficheroi --> proc
```

```
ficheroi --> root
ficheroi --> run
ficheroi --> sbin
ficheroi --> srv
ficheroi --> sys
ficheroi --> tmp
ficheroi --> usr
ficheroi --> var
```

- Por supuesto, podemos entrar en una Shell si es necesario, indicando que es interactivo

```
docker exec -it nginx1 bash
root@a0796441c01d:/# ls
bin   dev  home  lib64     mnt  proc  run   srv  tmp  var
boot  etc  lib   media     opt  root  sbin  sys  usr
```

- Salimos y paramos el contenedor

```
docker stop nginx1
nginx1
```

- Comprobamos que existe

```
docker ps -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                      PORTS
NAMES
a0796441c01d        nginx               "nginx -g 'daemon
of…"   44 minutes ago      Exited (0) 27 seconds ago
nginx1
28b5e0c7b73c        fedora              "bash"
45 minutes ago      Exited (0) 45 minutes ago
fedora1
8db922cac006        fedora:26           "bash"
4 days ago          Exited (0) 4 days ago
vibrant_spence
```

- Borramos el contenedor. Comprobamos que no está

```
docker rm nginx1
nginx1
```

```
docker ps -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                          PORTS
NAMES
28b5e0c7b73c        fedora              "bash"
About an hour ago   Exited (0) About an hour ago
fedora1
8db922cac006        fedora:26           "bash"
4 days ago          Exited (0) 4 days ago
vibrant_spence
```

- Vamos ahora a probar una imagen que tenga contenedores y que no podemos por tanto borrar directamente.
- En mi caso compruebo que existe la imagen fedora

```
docker images
REPOSITORY          TAG                 IMAGE ID
CREATED             SIZE
nginx               latest              73acd1f0cfad
7 days ago          109MB
fedora              26                  ce241ce855c8
13 days ago         232MB
fedora              latest              9110ae7f579f
13 days ago         235MB
busybox             latest              f6e427c148a7
2 weeks ago         1.15MB
hello-world         latest              f2a91732366c
4 months ago        1.85kB
```

- Intentamos borrarla. Si tenemos algún contenedor que la use no se dejará.

```
docker rmi fedora
Error response from daemon: conflict: unable to remove
repository reference "fedora" (must force) - container
28b5e0c7b73c is using its referenced image 9110ae7f579f
```

- Comprobamos que hay algún contenedor con esa imagen.
- En mi caso, tengo dos de fedora y uno de fedora:26

```
docker ps -a | grep fedora
```

```
28b5e0c7b73c        fedora              "bash"
3 hours ago         Exited (0) 3 hours ago
fedora1
8db922cac006        fedora:26           "bash"
4 days ago          Exited (0) 4 days ago
vibrant_spence
c284d12f3db6        fedora              "bash"
5 days ago          Exited (0) 4 days ago
jolly_clarke
```

- Si borro la imagen con la opción “force”:

```
docker rmi -f fedora
Untagged: fedora:latest
Untagged:
fedora@sha256:ec588fc80b05e19d3006bf2e8aa325f0a2e2ff1f609b
7afb39176ca8e3e13467
Deleted:
sha256:9110ae7f579f35ee0c3938696f23fe0f5fbe641738ea52eb83c
2df7e9995fa17
```

- Los contenedores de fedora aparecen con un id de imagen “fantasma”

```
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS                   PORTS
NAMES
28b5e0c7b73c        9110ae7f579f        "bash"
3 hours ago         Exited (0) 3 hours ago
fedora1
8db922cac006        fedora:26           "bash"
4 days ago          Exited (0) 4 days ago
vibrant_spence
c284d12f3db6        9110ae7f579f        "bash"
5 days ago          Exited (0) 5 days ago
jolly_clarke
```

- Podíamos borrarlo de la siguiente manera, pasando esa “id”

```
docker rm `docker ps -a | grep 9110ae7f579f | cut -c 1-12`
```

- En Docker no hay ninguna opción para borrar todo con un solo comando. Por tanto tenemos que acudir al Sistema Operativo para conseguirlo.

- Por ejemplo en Linux pondríamos algo como esto. Este comando primero devuelve los ids de todos los contenedores “$(Docker ps -qa)” y el resultado se lo pasa al comando “Docker rm”

```
docker rm $(docker ps -qa)
```