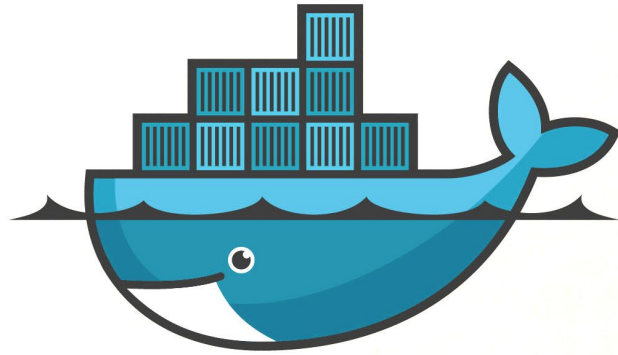


Contenedores



docker

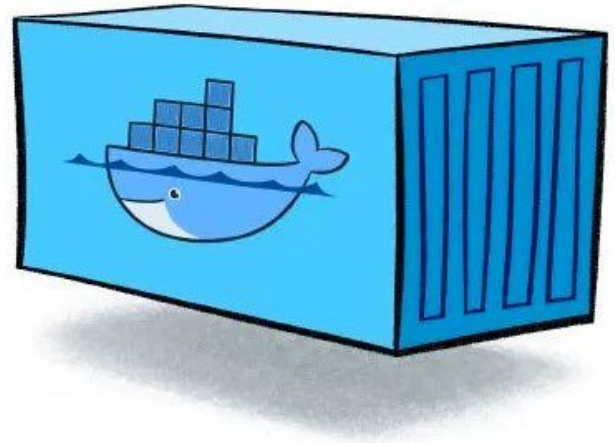
Repaso *Contenedores*

Un **contenedor** es una instancia de una imagen.

Contienen las **aplicaciones** que se ejecutan.

Pueden ser arrancados, parados y ejecutados.

Poseen un **identificador** único de 64 caracteres (con una versión corta de 12)



Comandos básicos

run	Crear y ejecutar
create	Crear
ps	Listar
start/stop/restart	Iniciar/Parar/Reiniciar
inspect	Obtener información
exec	Ejecutar comando

Comandos básicos

cp	Copiar archivos
attach	Enlazar stdin/stdout
log	Ver logs
rename	Cambiar nombre
rm	Eliminar



Comandos básicos - run

docker run *imagen:version*

Crea un contenedor a partir de la imagen y versión especificadas, y lo **ejecuta**.

```
pepe@pepe-VBox:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:4c5f3db4f8a54eb1e017c385f683a2de6e06f75be442dc32698c9bbe6c861edd
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Comandos básicos - run

```
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a49065fee71f	hello-world	"/hello"	6 days ago	Exited (0) 6 days ago		trusting_rosalind

Si se ejecuta otra vez, crea otro contenedor.

```
pepe@pepe-VBox:~$ docker run hello-world
```

Hello from Docker!
This message shows that your installation appears to be working correctly.

```
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ae860cc19af8	hello-world	"/hello"	8 seconds ago	Exited (0) 6 seconds ago		competent_bose
a49065fee71f	hello-world	"/hello"	6 days ago	Exited (0) 6 days ago		trusting_rosalind

Comandos básicos - run

Tiene muuuuuchas **opciones**

```
pepe@pepe-VBox:~$ docker run --help

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

Options:
  --add-host list          Add a custom host-to-IP mapping (host:ip)
  -a, --attach list        Attach to STDIN, STDOUT or STDERR
  --blkio-weight uint16    Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
  --blkio-weight-device list Block IO weight (relative device weight) (default [])
  --cap-add list           Add Linux capabilities
  --cap-drop list          Drop Linux capabilities
  --cgroup-parent string   Optional parent cgroup for the container
  --cgroupns string        Cgroup namespace to use (host|private)
                           'host': Run the container in the Docker host's cgroup namespace
                           'private': Run the container in its own private cgroup namespace
                           '': Use the cgroup namespace as configured by the
                              default-cgroupns-mode option on the daemon (default)
  --cidfile string         Write the container ID to the file
  --cpu-period int         Limit CPU CFS (Completely Fair Scheduler) period
  --cpu-quota int          Limit CPU CFS (Completely Fair Scheduler) quota
  --cpu-rt-period int      Limit CPU real-time period in microseconds
  --cpu-rt-runtime int     Limit CPU real-time runtime in microseconds
  -c, --cpu-shares int     CPU shares (relative weight)
  --cpus decimal           Number of CPUs
  --cpuset-cpus string     CPUs in which to allow execution (0-3, 0,1)
  --cpuset-mems string     MEMs in which to allow execution (0-3, 0,1)
  -d, --detach             Run container in background and print container ID
  --detach-keys string     Override the key sequence for detaching a container
```

Comandos básicos - run

Lanzar contenedor y ejecutar un **comando**:

```
pepe@pepe-VBox:~$ docker run ubuntu echo 'hola'  
hola
```

Lanzar contenedor y abrir un **terminal** interactivo:

```
pepe@pepe-VBox:~$ docker run -it ubuntu /bin/bash  
root@8e73db266b01:/# exit  
exit  
pepe@pepe-VBox:~$ █
```

Lanzar contenedor con un **nombre** concreto:

```
pepe@pepe-VBox:~$ docker run --name ubuntu01 ubuntu echo 'hola'  
hola  
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d9ed63ac696e	ubuntu	"echo hola"	8 seconds ago	Exited (0) 7 seconds ago		ubuntu01

Comandos básicos - run


Lanzar contenedor y **borrarlo** cuando se pare:

```
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
pepe@pepe-VBox:~$ docker run --rm --name ubuntu01 ubuntu echo 'hola'
hola
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES
```

Lanzar contenedor en **segundo plano**:

```
pepe@pepe-VBox:~$ docker run nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is
/docker-entrypoint.sh: Looking for shell scripts
/docker-entrypoint.sh: Launching /docker-entrypo
10-listen-on-ipv6-by-default.sh: info: Getting t
10-listen-on-ipv6-by-default.sh: info: Enabled l
/docker-entrypoint.sh: Launching /docker-entrypo
/docker-entrypoint.sh: Launching /docker-entrypo
```

Primer plano



```
pepe@pepe-VBox:~$ docker run -d nginx
31eac0bb3371d81827dbf06974b839b098240661d5f4f5e8c1c54b928905e6a0
pepe@pepe-VBox:~$
```

Segundo plano

Comandos básicos - run

Lanzar contenedor y **mapear puertos** con el anfitrión:

```
pepe@pepe-VBox:~$ docker run -d -p 3000:80 nginx
0b533713d5dedac3c8563493fcb781d7d560a3da2ad4a7c33f02220edf847f78
pepe@pepe-VBox:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0b533713d5de	nginx	"/docker-entrypoint...."	58 seconds ago	Up 57 seconds	0.0.0.0:3000->80/tcp, :::3000->80/tcp	quirky_elgamal



El mapeo de puertos **no se puede cambiar** una vez creado el contenedor.

Si no se especifica el puerto del anfitrión, lo elige Docker.

Comandos básicos - run

Lanzar contenedor y **mapear automáticamente los puertos expuestos** con el anfitrión:

```
→ docker run -d -P nginx  
7e8b670e8e8a10a8c1773f085e9277403b2395b0a5d6b2d7f5feb862ec59df8a
```

```
→ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
7e8b670e8e8a	nginx	"/docker-entrypoint..."	5 seconds ago	Up 4 seconds	0.0.0.0:49154->80/tcp, :::49154->80/tcp

Los puertos expuestos son configurados a la hora de hacer la imagen.

Comandos básicos - run

Lanzar contenedor y definir **variables de entorno**:

```
pepe@pepe-VBox:~$ docker run -it -e VAR='hola' bash
bash-5.1# echo $VAR
hola
bash-5.1# exit
exit
pepe@pepe-VBox:~$
```

Las variables de entorno se utilizan **para configurar** aspectos de contenedor.

Algunas son **opcionales** y otras **obligatorias**.

Hay que **consultar la documentación** de cada imagen.

Comandos básicos - run

Lanzar contenedor y **enlazarlo a otro contendor**:

```
➔ docker run --link mysql:db -d wordpress  
45541b0f4f32c6e7ada08b9d95040471914f5d1ca158b2023c0407510ce9f221
```

La opción **--link** permite acceder a otro contenedor por nombre del contenedor o por alias, si se especifica (nombre:alias).

Lo que se hace internamente es modificar el archivo **/etc/hosts** del contenedor. Es lo que se conoce como **resolución de nombres estática**.

También **comparte** las variables de entorno del contenedor enlazado.

Esta opción se considera **obsoleta**, en favor de usar redes privadas de docker.



Comandos básicos - create

docker create *imagen:version*

Crea un contenedor a partir de la imagen y versión especificadas.

A diferencia de run, **no lo ejecuta**.

Devuelve el **identificador** del contenedor.

```
pepe@pepe-VBox:~$ docker create hello-world  
286c9463200c48242922e11816580d37570a883592ebea5bb0d0f88280a91966
```

```
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
<u>286c9463200c</u>	hello-world	"/hello"	2 minutes ago	Created		vibrant_kepler
<u>4cde466fed02</u>	hello-world	"/hello"	6 minutes ago	Exited (0) 5 minutes ago		zealous_shockley

Comandos básicos - create

Admite las mismas **opciones** que run.

```
pepe@pepe-VBox:~$ docker create --help
```

```
Usage: docker create [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Create a new container

Options:

--add-host list	Add a custom host-to-IP mapping (host:ip)
-a, --attach list	Attach to STDIN, STDOUT or STDERR
--blkio-weight uint16	Block IO (relative weight), between 10 and 1000, or 0 to disable (default 0)
--blkio-weight-device list	Block IO weight (relative device weight) (default [])
--cap-add list	Add Linux capabilities
--cap-drop list	Drop Linux capabilities
--cgroup-parent string	Optional parent cgroup for the container
--cgroupns string	Cgroup namespace to use (host private)
	'host': Run the container in the Docker host's cgroup namespace
	'private': Run the container in its own private cgroup namespace
	':': Use the cgroup namespace as configured by the default-cgroupns-mode option on the daemon (default)
--cidfile string	Write the container ID to the file
--cpu-period int	Limit CPU CFS (Completely Fair Scheduler) period
--cpu-quota int	Limit CPU CFS (Completely Fair Scheduler) quota
--cpu-rt-period int	Limit CPU real-time period in microseconds
--cpu-rt-runtime int	Limit CPU real-time runtime in microseconds
-C, --cpu-shares int	CPU shares (relative weight)
--cpus decimal	Number of CPUs
--cpuset-cpus string	CPUs in which to allow execution (0-3, 0,1)
--cpuset-mems string	MEMs in which to allow execution (0-3, 0,1)
--device list	Add a host device to the container
--device-cgroup-rule list	Add a rule to the cgroup allowed devices list



Comandos básicos - ps

docker ps
docker ps -a

Lista los contenedores en ejecución.

El parámetro **-a** lista también los contenedores parados.

```
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
286c9463200c	hello-world	"/hello"	2 minutes ago	Created		vibrant_kepler
4cde466fed02	hello-world	"/hello"	6 minutes ago	Exited (0) 5 minutes ago		zealous_shockley



Comandos básicos - start/stop/restart

```
docker start id/name  
docker stop id/name  
docker restart id/name
```

Inicia/para/reinicia un contenedor.

```
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ae860cc19af8	hello-world	"/hello"	18 minutes ago	Exited (0) 18 minutes ago		competent_bose
a49065fee71f	hello-world	"/hello"	6 days ago	Exited (0) 6 days ago		trusting_rosalind

```
pepe@pepe-VBox:~$ docker start trusting_rosalind  
trusting_rosalind
```

```
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ae860cc19af8	hello-world	"/hello"	19 minutes ago	Exited (0) 19 minutes ago		competent_bose
a49065fee71f	hello-world	"/hello"	6 days ago	Exited (0) 4 seconds ago		trusting_rosalind

Comandos básicos - start/stop/restart

La opción **-a** muestra la salida del contenedor.

```
pepe@pepe-VBox:~$ docker start -a trusting_rosalind
```

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

```
To generate this message, Docker took the following steps:
```

1. The Docker client contacted the Docker daemon.

Comandos básicos - start/stop/restart

La opción **-i** enlaza la entrada estándar.

Se usa para contenedores interactivos que esperan entrada de datos.

El contenedor debe haberse creado con la opción **-it** antes.

```
pepe@pepe-VBox:~$ docker create -it bash
9efc2839480fcea64fca8fdab371236f17658b9a0521384b19709a65775a95df
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
9efc2839480f   bash          "docker-entrypoint.s..." 4 seconds ago  Created                                gifted_kare
ae860cc19af8   hello-world   "/hello"                 38 minutes ago Exited (0) 38 minutes ago             competent_bose
a49065fee71f   hello-world   "/hello"                 6 days ago    Exited (0) 15 minutes ago             trusting_rosalind
pepe@pepe-VBox:~$ docker start -i 9efc2839480f
bash-5.1# ls
bin    dev    etc    home  lib    media mnt    opt    proc  root  run    sbin  srv    sys    tmp    usr    var
bash-5.1# exit
exit
pepe@pepe-VBox:~$
```



Comandos básicos - inspect

docker inspect *id/name*

Proporciona mucha **información** sobre el contenedor.

```
pepe@pepe-VBox:~$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9efc2839480f	bash	"docker-entrypoint.s..."	7 minutes ago	Exited (0) 7 minutes ago		<u>gifted_kare</u>
ae860cc19af8	hello-world	"/hello"	46 minutes ago	Exited (0) 46 minutes ago		competent_bose
a49065fee71f	hello-world	"/hello"	6 days ago	Exited (0) 23 minutes ago		trusting_rosalind

Comandos básicos - inspect

```
pepe@pepe-VBox:~$ docker inspect gifted_kare
[
  {
    "Id": "9efc2839480fcea64fca8fdab371236f17658b9a0521384b19709a65775a95df",
    "Created": "2022-03-28T18:10:30.465728548Z",
    "Path": "docker-entrypoint.sh",
    "Args": [
      "bash"
    ],
    "State": {
      "Status": "exited",
      "Running": false,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 0,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2022-03-28T18:10:44.266357581Z",
      "FinishedAt": "2022-03-28T18:10:48.943354892Z"
    },
    "Image": "sha256:91d5f5779560f72b02d2bf2d7eca9969faabf6a26bbea96d61c4af5c7d6ea76a",
    "ResolvConfPath": "/var/lib/docker/containers/9efc2839480fcea64fca8fdab371236f17658b9a0521384b19709a65775a95df/resolv.co",
    "HostnamePath": "/var/lib/docker/containers/9efc2839480fcea64fca8fdab371236f17658b9a0521384b19709a65775a95df/hostname",
    "HostsPath": "/var/lib/docker/containers/9efc2839480fcea64fca8fdab371236f17658b9a0521384b19709a65775a95df/hosts",
    "LogPath": "/var/lib/docker/containers/9efc2839480fcea64fca8fdab371236f17658b9a0521384b19709a65775a95df/9efc2839480fcea64f17658b9a0521384b19709a65775a95df-json.log",
  }
]
```



Comandos básicos - exec

```
docker exec [opciones] id/name COMANDO [argumentos]
```

Ejecuta un **comando** en un contenedor.

El contenedor debe estar **en ejecución**.

Si el contenedor no está en ejecución, se usa `docker run` para ejecutar comandos.

Comandos básicos - exec

Vamos a usar la imagen de nginx como **ejemplo**, ya que al crear un contenedor se ejecuta hasta que lo paremos.

Nginx es un servidor web.

```
pepe@pepe-VBox:~$ docker run -d -p 80:80 nginx  
62c38067607ff0ad07c7e48657dc687dff4594ea417809f627ea143417f8b060
```

La opción -d de run hace que el contenedor se ejecute en segundo plano y no bloquee el terminal.

La opción -p de run mapea puertos entre el contenedor y el host.

```
pepe@pepe-VBox:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
62c38067607f	nginx	"/docker-entrypoint..."	35 seconds ago	Up 34 seconds	0.0.0.0:80->80/tcp, :::80->80/tcp	pensive_babbage

Comandos básicos - exec

Podemos ver que el servidor funciona accediendo a: `http://localhost`



Comandos básicos - exec

Ahora podemos usar exec para crear un archivo:

```
pepe@pepe-VBox:~$ docker exec -d pensive_babbage touch /tmp/mi_archivo.txt  
pepe@pepe-VBox:~$
```

También podemos abrir un terminal con las opciones **-it**:

```
pepe@pepe-VBox:~$ docker exec -it pensive_babbage bash  
root@62c38067607f:/# ls /tmp/  
mi_archivo.txt  
root@62c38067607f:/# exit  
exit  
pepe@pepe-VBox:~$
```

Se ve el archivo que acabamos de crear

Comandos básicos - exec

También es útil poder abrir un terminal con variables de entorno definidas:

```
pepe@pepe-VBox:~$ docker exec -it -e MY_VAR=5 pensive_babbage bash
root@62c38067607f:/# echo $MY_VAR
5
root@62c38067607f:/# exit
exit
pepe@pepe-VBox:~$
```



Comandos básicos - cp

```
docker cp origen destino
```

Copia archivos entre un contenedor y el anfitrión.

No se puede copiar archivos directamente entre contenedores

Ejemplo:

1. Creamos un contenedor
2. Creamos un archivo en el anfitrión y lo copiamos al contenedor
3. Accedemos al archivo desde el contenedor

Comandos básicos - cp

1

```
pepe@pepe-VBox:~$ docker run -it ubuntu /bin/bash
root@3c01aee1ad10:/# pwd
/
root@3c01aee1ad10:/# ls
bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin srv sys tmp usr var
```

2

```
pepe@pepe-VBox:~$ ls
Descargas Documentos Escritorio Imágenes Música Plantillas Público Vídeos
pepe@pepe-VBox:~$ echo 'hola' > archivo_01.txt
pepe@pepe-VBox:~$ cat archivo_01.txt
hola
pepe@pepe-VBox:~$ ls
archivo_01.txt Descargas Documentos Escritorio Imágenes Música Plantillas Público Vídeos
pepe@pepe-VBox:~$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS              PORTS          NAMES
3c01aee1ad10   ubuntu   "/bin/bash"             About a minute ago   Up About a minute     
pepe@pepe-VBox:~$ docker cp archivo_01.txt funny_wiles:/
pepe@pepe-VBox:~$
```

3

```
root@3c01aee1ad10:/# ls
archivo_01.txt bin boot dev etc home lib lib32 lib64 libx32 media mnt opt proc root run sbin
root@3c01aee1ad10:/# cat archivo_01.txt
hola
```



Comandos básicos - attach

```
docker attach id/name
```

Enlaza stdin y stdout con un contenedor en ejecución.

Normalmente los contenedores se ejecutan en segundo plano (opción -d), y con attach pueden volver a “primer plano”.

Comandos básicos - attach

Ejemplo:

```
pepe@pepe-VBox:~$ docker run -d --name=imprime_fecha ubuntu sh -c "while true; do $(echo date); sleep 1; done"
648decdf8cef93edcbd64bcb0c70cc8133eb4b57a88d82cb39470f50114a5c13
```

```
pepe@pepe-VBox:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
648decdf8cef	ubuntu	"sh -c 'while true; ...'"	14 seconds ago	Up 13 seconds		imprime_fecha

```
pepe@pepe-VBox:~$ docker attach imprime_fecha
Tue Mar 29 14:50:37 UTC 2022
Tue Mar 29 14:50:38 UTC 2022
Tue Mar 29 14:50:39 UTC 2022
Tue Mar 29 14:50:40 UTC 2022
Tue Mar 29 14:50:41 UTC 2022
Tue Mar 29 14:50:42 UTC 2022
^Cpepe@pepe-VBox:~$
```



Comandos básicos - logs

```
docker logs id/name
```

Muestra el **stdout/stderr** de un contenedor en ejecución.

Opciones interesantes:

- **-f**: sigue mostrando los logs.
- **-n num**: muestra los últimos “num” mensajes.

Comandos básicos - logs

Ejemplos:

```
pepe@pepe-VBox:~$ docker run -d --name=imprime_fecha ubuntu sh -c "while true; do $(echo date); sleep 5; done"  
4011ccf86bc6e173f1d288e2b551bb22fdfa9d868f27e3fb3ae2f45ed5323893
```

```
pepe@pepe-VBox:~$ docker logs -f imprime_fecha  
Tue Mar 29 15:10:48 UTC 2022  
Tue Mar 29 15:10:53 UTC 2022  
Tue Mar 29 15:10:58 UTC 2022  
^C
```

```
pepe@pepe-VBox:~$ docker logs -n 2 imprime_fecha  
Tue Mar 29 15:11:03 UTC 2022  
Tue Mar 29 15:11:08 UTC 2022
```

```
pepe@pepe-VBox:~$ docker logs -f -n 2 imprime_fecha  
Tue Mar 29 15:11:13 UTC 2022  
Tue Mar 29 15:11:18 UTC 2022  
Tue Mar 29 15:11:23 UTC 2022  
^C  
pepe@pepe-VBox:~$
```




Comandos básicos - rename

docker rename *id/name new_name*

Cambia el nombre de un contenedor

```
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
4011ccf86bc6   ubuntu   "sh -c 'while true; ..." 8 minutes ago  Exited (137) 30 seconds ago           imprime_fecha
3c01aee1ad10   ubuntu   "/bin/bash"              43 minutes ago Exited (0) 29 minutes ago           funny_wiles
pepe@pepe-VBox:~$ docker rename 3c01aee1ad10 funny_meli
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
4011ccf86bc6   ubuntu   "sh -c 'while true; ..." 8 minutes ago  Exited (137) 54 seconds ago           imprime_fecha
3c01aee1ad10   ubuntu   "/bin/bash"              43 minutes ago Exited (0) 29 minutes ago           funny_meli
pepe@pepe-VBox:~$ docker rename imprime_fecha print_date
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
4011ccf86bc6   ubuntu   "sh -c 'while true; ..." 9 minutes ago  Exited (137) 2 minutes ago           print_date
3c01aee1ad10   ubuntu   "/bin/bash"              44 minutes ago Exited (0) 30 minutes ago           funny_meli
```



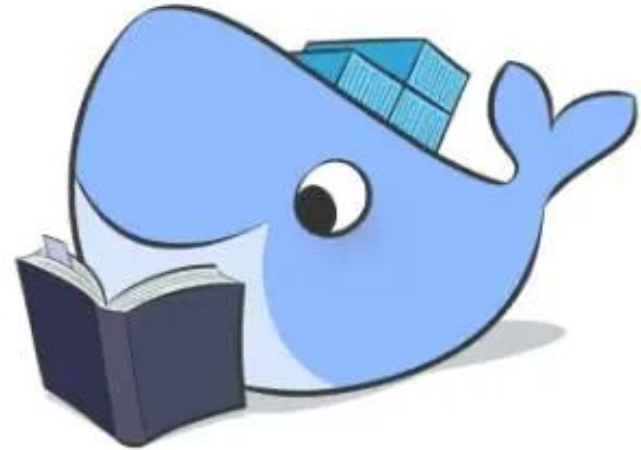
Comandos básicos - rm

docker rm *id/name*

Borra un contenedor que no está en ejecución.

```
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
4011ccf86bc6   ubuntu   "sh -c 'while true; ..." 9 minutes ago  Exited (137) 2 minutes ago           print_date
3c01aee1ad10   ubuntu   "/bin/bash"              44 minutes ago Exited (0) 30 minutes ago           funny_meli
pepe@pepe-VBox:~$ docker rm print_date
print_date
pepe@pepe-VBox:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS              PORTS          NAMES
3c01aee1ad10   ubuntu   "/bin/bash"              47 minutes ago Exited (0) 33 minutes ago           funny_meli
pepe@pepe-VBox:~$
```

Ejercicios



Ejercicio 1



Ejercicio 1a

Crea y ejecuta un contenedor a partir de la imagen de nginx con las siguientes características:

- Nombre: webserver.
- Mapeo de puertos: el 80 del contenedor al 8080 del anfitrión.
- Que se ejecute en segundo plano.

Accede a la web desde Firefox para que se vea.

Ejercicio 1b

Ahora abre un terminal interactivo con el contenedor y cambia la página web que se ve por defecto, que está en “/usr/share/nginx/html” por una que muestre un título y una imagen.

La imagen tendrás que copiarla desde el anfitrión al contenedor y meterla en la carpeta correspondiente para que se vea en la web.

Ejercicio 2



Ejercicio 2

Utilizando las imágenes oficiales de Docker de Wordpress y MySQL monta una página web con Wordpress funcional.

Después añade un contenedor con phpmyadmin que permita la administración de la base de datos.



Ejercicio 2

Paso 1: Ejecutar un contenedor con MySQL

Accede a la documentación de la imagen oficial de MySQL en [Docker Hub](#) para saber como funciona, y ejecuta un contenedor a partir de ella llamado **database**.

Ten en cuenta que necesitarás utilizar al menos las siguientes variables de entorno:

- MYSQL_ROOT_PASSWORD
- MYSQL_DATABASE

También tendrás que mapear el puerto que usa MySQL con el anfitrión para poder acceder luego desde el contenedor de Wordpress.



WORDPRESS

Ejercicio 2

Paso 2: Ejecutar un contenedor con Wordpress

Accede a la documentación de la imagen oficial de Wordpress en [Docker Hub](#) para saber como funciona, y ejecuta un contenedor a partir de ella llamado **web**.

No hace falta usar ninguna variable de entorno, ya que tienen una interfaz web de configuración; pero sí que tendrás que mapear el puerto HTTP con el anfitrión para poder acceder desde este a la web.



WORDPRESS

Ejercicio 2

Paso 2: Ejecutar un contenedor con Wordpress

Desde el contenedor de Wordpress se podrá acceder a la base de datos de 2 formas:

1. Usando la opción:

--link <nombre_contenedor_mysql>:db

2. A través de la dirección siguiente (siempre que se haya mapeado el puerto de MySQL correctamente):

ip_interfaz_docker0:puerto

Ejercicio 2



Paso 3: Ejecutar un contenedor con PhpMyAdmin

Accede a la documentación de la imagen oficial de PhpMyAdmin en [Docker Hub](#) para saber como funciona, y ejecuta un contenedor a partir de ella llamado **dbadmin**.

Hay que usar la opción:

--link <nombre_contenedor_mysql>:db