

# Resolución del ejercicio inicial

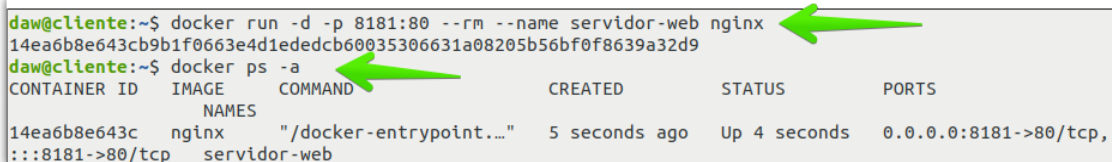
Realizado por Inés Menéndez

Crear un contenedor demonio a partir de la imagen `nginx`, el contenedor se debe llamar `servidor_web` y se debe acceder a él utilizando el puerto 8181 del ordenador donde tengas instalado docker.

## Cuestiones

1. Pantallazo donde se vea la creación del contenedor y podamos comprobar que el contenedor está funcionando.

```
docker run -d -p 8181:80 --rm --name servidor-web nginx
docker ps -a
```



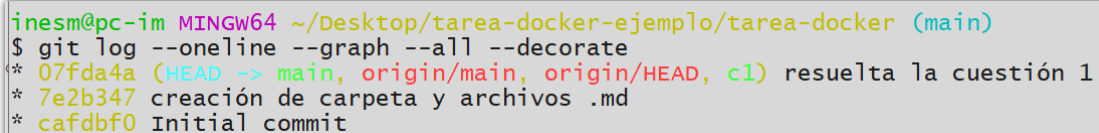
A terminal window showing the execution of Docker commands. The first command is `docker run -d -p 8181:80 --rm --name servidor-web nginx`, which creates a new container. The second command is `docker ps -a`, which lists all containers. The output shows a single container with ID `14ea6b8e643c`, image `nginx`, and status `Up 4 seconds`. Two green arrows point to the container ID and the `docker ps -a` command.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
14ea6b8e643c	nginx	"/docker-entrypoint..."	5 seconds ago	Up 4 seconds	0.0.0.0:8181->80/tcp, :::8181->80/tcp

Se muestra a modo de ejemplo cómo va avanzando el repositorio

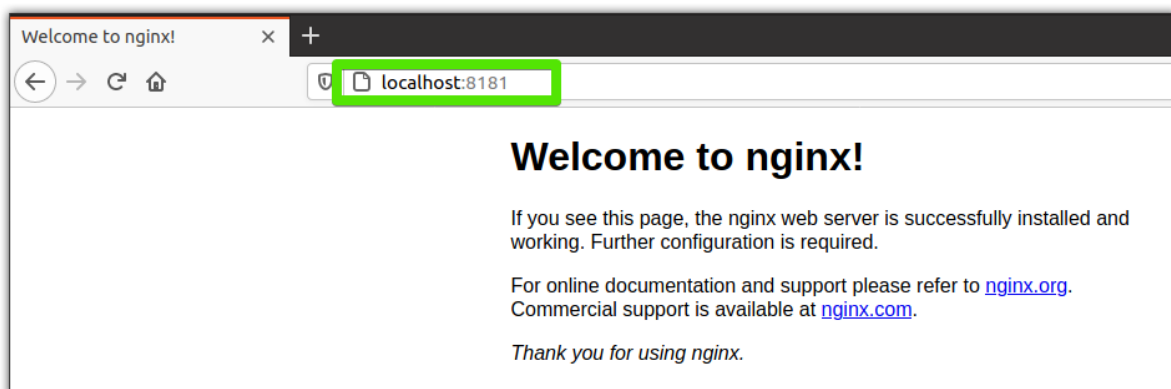
Fijarse en que se ha utilizado una rama **c1** para resolver la primera cuestión, y una vez finalizado se fusionó con 'main'

-- NO es necesario documentarlo



A terminal window showing the output of the `git log --oneline --graph --all --decorate` command. The output shows three commits: `07fda4a` (HEAD -> main, origin/main, origin/HEAD, c1) resuelta la cuestión 1, `7e2b347` creación de carpeta y archivos .md, and `cafdbf0` Initial commit. The commit `07fda4a` is highlighted with a green box.

2. Pantallazo donde se vea el acceso al servidor web utilizando un navegador web (recuerda que tienes que acceder a la ip del ordenador donde tengas instalado docker)



Vemos, a modo de ejemplo, cómo está el repositorio con las dos ramas:

c2 had recent pushes 1 minute ago
 [Compare & pull request](#)

main
 2 branches
 0 tags
 [Go to file](#)
[Add file](#)
[Code](#)

inesAlumnaDaw resuelta la cuestión 1
 07fda4a 9 minutes ago 3 commits

initial	resuelta la cuestión 1	9 minutes ago
README.md	creación de carpeta y archivos .md	18 minutes ago

README.md

## tarea-docker

Ejemplo de resolución de tarea. Realizado por Inés Menéndez

3. Pantallazo donde se vean las imágenes que tienes en tu registro local.

docker images

```
daw@cliente:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx	latest	f6987c8d6ed5	2 months ago	141MB
httpd	2.4	dabbfbfe0c57b	2 months ago	144MB
ubuntu	latest	ba6accce9d29	4 months ago	72.8MB
hello-world	latest	feb5d9fea6a5	5 months ago	13.3kB

Se muestra cómo vamos actualizando el repositorio, utilizando ramas y subiendo los cambios al remoto -- NO es necesario documentar esto en la tarea, se muestra como orientación...

```
inesm@pc-im MINGW64 ~/Desktop/tarea-docker-ejemplo/tarea-docker (c3)
$ git add .

inesm@pc-im MINGW64 ~/Desktop/tarea-docker-ejemplo/tarea-docker (c3)
$ git commit -m "resuelta cuestión 3"
[c3 b2e1b5c] resuelta cuestión 3
2 files changed, 7 insertions(+), 1 deletion(-)
create mode 100644 inicial/Ejercicio inicial.assets/image-20220309173417360.png

inesm@pc-im MINGW64 ~/Desktop/tarea-docker-ejemplo/tarea-docker (c3)
$ git push origin c3
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 27.90 KiB | 3.49 MiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'c3' on GitHub by visiting:
remote:   https://github.com/inesAlumnaDaw/tarea-docker/pull/new/c3
remote:
To https://github.com/inesAlumnaDaw/tarea-docker.git
 * [new branch]      c3 -> c3

inesm@pc-im MINGW64 ~/Desktop/tarea-docker-ejemplo/tarea-docker (c3)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

3. Pantallazo donde se vea cómo se elimina el contenedor (recuerda que antes debe estar parado el contenedor).

Para resolver esta cuestión, creamos un contenedor que expone el puerto 80 al puerto 8181 de nuestro cliente, vemos que está 'corriendo'-UP, lo paramos, vemos que se ha parado, lo borramos y comprobamos que se ha borrado

```
docker run -d -p 8181:80 --name servidor-web nginx
docker ps
docker stop servidor-web
docker ps -a
docker rm servidor-web
```

```
daw@cliente:~$ docker run -d -p 8181:80 --name servidor-web nginx
e422623547945b3d862e985294e37857e76412f7f1b9a6e359a11f4c853ccee7
daw@cliente:~$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
e42262354794   nginx     "/docker-entrypoint..." 4 seconds ago  Up 3 seconds  0.0.0.0:8181->80/tcp,
:::8181->80/tcp  servidor-web
daw@cliente:~$ docker stop servidor-web
servidor-web
daw@cliente:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
e42262354794   nginx     "/docker-entrypoint..." 16 seconds ago Exited (0) 2 seconds ago
servidor-web
daw@cliente:~$ docker rm servidor-web
servidor-web
daw@cliente:~$ docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
```