

TAREA 1 – EJECUTAR CONTENEDORES SIMPLES

Aprender a usar contenedores para diferentes casos de uso.

Contenedor de una sola tarea

Ejemplo: Ejecutar hostname en Alpine Linux.

Comando: `docker container run alpine hostname`

Contenedor interactivo

Ejemplo: Acceder a una terminal en un contenedor Ubuntu.

Comando: `docker container run -it --rm ubuntu bash`

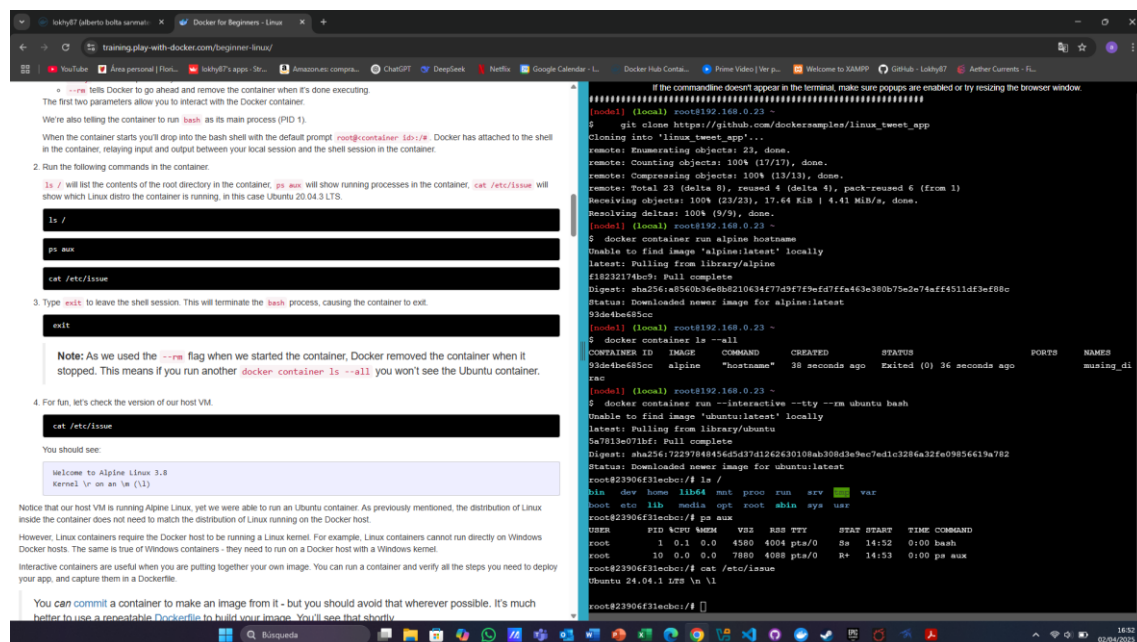
Permite ejecutar comandos dentro del contenedor (`ls`, `cat /etc/issue`).

Contenedor en segundo plano

Ejemplo: Ejecutar MySQL como servicio.

Comando: `docker container run --detach --name mydb -e MYSQL_ROOT_PASSWORD=123 mysql`

Usamos `docker container logs` y `docker exec` para interactuar con él.



The image shows a browser window displaying a Docker tutorial for Linux. The tutorial includes instructions on running containers, listing contents, and checking the host version. To the right, a terminal window shows the execution of these commands. The terminal output shows the successful execution of `docker container run alpine hostname`, which returns `root8192.168.0.23`. Subsequently, `docker container run -it --rm ubuntu bash` is executed, and the user is prompted for a password. The terminal also shows the output of `cat /etc/issue` inside the Ubuntu container, displaying the Alpine Linux version and kernel information.

TAREA 2 – CREAR Y EJECUTAR UNA APLICACIÓN PERSONALIZADA

Empaquetar una app web (NGINX) en una imagen Docker.

Construir la imagen

Se usa un Dockerfile para copiar archivos HTML y configurar NGINX.

Comando: `docker image build --tag mi-app:1.0`.

Ejecutar el contenedor

Publicar el puerto 80 del host hacia el contenedor: `docker container run -d -p 80:80 --name web-app mi-app:1.0`

Verificar la app

Acceder a `http://localhost` para ver el sitio web.

The screenshot shows a web browser displaying a Docker tutorial. The page includes a list of steps and a terminal window showing the execution of Docker commands. The steps are:

- Stop and remove the currently running container.
- Re-run the current version without a bind mount.
- Notice the website is back to the original version.
- Stop and remove the current container.

Update the image

To persist the changes you made to the `index.html` file into the image, you need to build a new version of the image.

- Build a new image and tag it as `2.0`.

Remember that you previously modified the `index.html` file on the Docker hosts local filesystem. This means that running another `docker image build` command will build a new image with the updated `index.html`.

Be sure to include the period (`.`) at the end of the command.

```
docker image build --tag $DOCKERID/linux_tweet_app:2.0 .
```

Notice how fast that built! This is because Docker only modified the portion of the image that changed vs. rebuilding the whole image.

- Let's look at the images on the system.

```
docker image ls
```

You now have both versions of the web app on your host.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker id/linux_tweet_app	2.0	401812e95312b	16 seconds ago	1009B
docker id/linux_tweet_app	1.0	bb32b5783cd3	4 minutes ago	1009B
mysql	latest	34e7708090cf3	2 weeks ago	412MB
ubuntu	latest	20896327ab2e	2 weeks ago	122MB
nginx	latest	da5939581ac8	3 weeks ago	1009B
alpine	latest	76da59c80196	3 weeks ago	3.97MB

Test the new version

The terminal output shows the following commands and their results:

```
docker rm --force linux_tweet_app
docker container run \
  --detach \
  --publish 80:80 \
  --name linux_tweet_app \
  $DOCKERID/linux_tweet_app:1.0
[+] Building 0.0s (0/0)
ERROR: invalid tag "/linux_tweet_app:1.0": invalid reference format
docker: invalid reference format.
Error response from daemon: No such container: linux_tweet_app
[+] Building 0.0s (0/0)
$ docker container run \
  --detach \
  --publish 80:80 \
  --mount type=bind,source=$(pwd),target=/usr/share/nginx/html \
  $DOCKERID/linux_tweet_app:1.0
docker: invalid reference format.
See 'docker run --help'.
[+] Building 0.0s (0/0)
$ cp index-new.html index.html
[+] Building 0.0s (0/0)
$ docker rm --force linux_tweet_app
Error response from daemon: No such container: linux_tweet_app
[+] Building 0.0s (0/0)
$ docker image build --tag $DOCKERID/linux_tweet_app:2.0 .
[+] Building 0.0s (0/0)
ERROR: invalid tag "/linux_tweet_app:2.0": invalid reference format
[+] Building 0.0s (0/0)
$ docker image ls
```

TAREA 3 – MODIFICAR UNA APP EN EJECUCIÓN Y DISTRIBUIRLA

Actualizar la app sin reconstruir la imagen y subirla a Docker Hub.

Modificación en caliente

Usar *bind mounts* para sincronizar cambios locales: `docker run -d -p 80:80 -v $(pwd):/usr/share/nginx/html --name web-app mi-app:1.0`

Cambios en `index.html` se reflejan al instante.

Crear una nueva versión

Reconstruir la imagen con los cambios: `docker image build --tag mi-app:2.0`.

Subir a Docker Hub

Compartir la imagen públicamente: `docker image push mi-docker-id/mi-app:2.0`

