instrucciones Dockerfile

volumes

- `docker run -v $PWD:/var/www/html/ <imagen>` : mapea $PWD al directorio indicado dentro del contenedor
- `docker exec -it <contenedor_id> bash` : acceder al contenedor

los volúmenes se usan para persistencia, ya que un contenedor cuando se para, todo lo que hubiese se pierde.

label

- `label key=value`: etiquetas a las imágenes (para posterior filtrado)

arg

- `arg key=value`: variables de uso interno

env

- `env key=value`: variables de entorno en el contenedor (para nuestra app)

add

- `add <origen> <destino>`: copia un fichero o una URL en el destino
  - es un copy on steroids
  - también descomprime un tgz o similar directamente
  - CREA UNA LAYER (como RUN y COPY)

expose

- `expose 80/tcp`: informativo (de cara a quien trabaje con esa imagen y a docker para trabajar con el `-P` (mapeo automático de puertos))

workdir

- `workdir /app`: cd al directorio

imágenes alpine

- **nginx:latest** VS **nginx:1.15-alpine**
  - las versiones **alpine** están basadas en la imagen **alpine**, muy ligera

user

- USER <user||uid>: cambiar de usuario (buena práctica usar UID)
- RUN useradd -u 1001 -m -g root dani
  - en **openshift** siempre ha de estar el usuario añadido al grupo de **root**

```
#!/bin/bash fix-permissions.sh
for arg in "$@"
do
    find "$arg" -exec chgrp 0 {}\;
    find "$arg" -exec chmod g+rw {}\;
    find "$arg" -type d -exec chmod g+x {}\;
done
```

si extendemos de una imagen que ha cambiado el USER y nosotros hemos de realizar cambios, hemos de hacer el cambio a USER root, hacemos lo que necesitemos y lo volvemos a dejar con el usuario que toque.

context

- docker build -t <imagen> . : el **.** es el contexto donde buscar todos los ficheros a los que hacemos referencia, incluido **Dockerfile** al demonio docker
  - para evitar el problema del **Dockerfile**, podemos pasarle el parámetro -f <ubicación_Dockerfile>
  - para hacerlo más límpio, usar el **.** siempre y todo referenciado desde ahí

bbdd

bien para desarrollo, no para producción

es un cuello de botella, ya que para mantener la persistencia hay que montar un volumen por NFS (red), lo que hace que sea lento

los DBA ullen de las máquinas virtuales, quieren servidores físicos, así que de docker...

volumes

- VOLUME /myvol: informamos a Docker que ahí habrá un volumen (implica persistencia)
  - funcionamiento parecido a EXPOSE y -P
  - si no especificamos nosotros con -v se gestiona «internamente»
    - docker volume ls
    - docker volume inspect <hash_volumen>
    - docker inspect <contenedor_id>

  - FROM ... RUN mkdir /app RUN echo "..." /app/hello VOLUME /app

```
RUN echo "..." > /app/hello2
```

entrypoint

- es lo primero que se ejecuta (antes que CMD)
- ENTRYPOINT [«echo»]
  - a este comando le llega por parámetro lo que viene desde CMD
- ''ENTRYPOINT [«/entrypoint.sh»]
  - ejecución en tiempo de ejecución
  - ```
    #! /bin/bash
    exec "$@"
    ```
    ```