Resumen de comandos DOCKER

COMANDOS

ATTACH

Conecta la E/S std, y la consola de error a la consola del host, para así poder operar diréctamente con un contenedor que ya está corriendo.

Para parar el contenedor una vez que queramos salir de su consola, pulsar `Ctrl+V`, si queremos que el contenedor siga corriendo, pulsar `Ctrl+P` + `Ctrl+Q`.

Uso: `docker attach [OPTIONS] CONTAINER`

BUILD

Construye un contenedor en base a un fichero dockerfile existente en un directorio especificado. Todo lo que se ponga en ese directorio pasará a ser transferido al contenedor: *CUIDADO*.

Este comando **NO LANZA** el contenedor, sólo lo construye. Para lanzarlo, utiliza RUN.

Uso: `docker build [OPTIONS] PATH | URL | -`

- `--tag , -t`: Name and optionally a tag in the 'name:tag' format

CONTAINER EXEC

Ejecuta un comando en un contenedor que ya está corriendo

Uso: `docker exec [OPTIONS] CONTAINER COMMAND [ARG...]`

- `--detach , -d`: Detached mode: run command in the background
- `--detach-keys`: Override the key sequence for detaching a container
- `--env , -e`: Set environment variables
- `--interactive , -i`: Keep STDIN open even if not attached
- `--privileged`: Give extended privileges to the command
- `--tty , -t`: Allocate a pseudo-TTY
- `--user , -u`: Username or UID (format: <name|uid>[:<group|gid>])
- `--workdir , -w`: Working directory inside the container

CONTAINER LOGS

Busca los logs de un contenedor

Uso: `docker container logs [OPTIONS] CONTAINER`

CONTAINER LS o CONTAINER PS.

Lista los contenedores en ejecución. Si se quieren tambien listar los que están parados, hay que añadir la opción `-a`.

CONTAINER PRUNE

Quita los contenedores parados.

CONTAINER STOP

Para uno o mas contenedores en ejecución. Los contenedores no se borran y siguen a la espera de volver a ser ejecutados. Se pueden eliminar con el comando **PRUNE**.

Uso: `docker container stop [OPTIONS] CONTAINER [CONTAINER...]`

IMAGE LS

Lista las imágenes existentes . Si se quieren tambien listar todas, hay que añadir la opción `-a`.

NETWORK CONNECT

Conecta un contenedor a una red anteriormente creada.

Uso: `docker network connect [OPTIONS] NETWORK CONTAINER`

- `--ip`: IPv4 address (e.g., 172.30.100.104)

NETWORK CREATE

Crea una red en docker para que los contenedores puedan conectarse.

Hay que tener en cuenta que al conectarse, cada contenedor tiene un nombre dentro de la red, igual al de su TAG, y por lo tanto, se puede utilizar este nombre como si operara dentro de la red un DNS que los conociera (comandos ping ...).

Uso: `docker network create [OPTIONS] NETWORK`

NETWORK DISCONNECT

Desconecta a un contenedor de una red.

Uso: `docker network disconnect [OPTIONS] NETWORK CONTAINER`

- `--force , -f`: Fuerza la desconexión.

NETWORK INSPECT

Muestra información detallada de una red.

Uso: `docker network inspect [OPTIONS] NETWORK [NETWORK...]`

NETWORK LS

Lista las redes existentes.

RM

Elimina uno o mas contenedores

Uso: `docker rm [OPTIONS] CONTAINER [CONTAINER...]`

- `-f`: Fuerza la parada del contenedor y su posterior eliminación.
- `-v`: Elimina los volúmenes asociados con el contenedor.

RUN

Ejecuta un comando en un contenedor nuevo.

Uso: `docker run [OPTIONS] IMAGE [COMMAND] [ARG del comando]`

- `--interactive`: indica que se quiere una sesión interactiva.
- `--tty`: reserva un pseudo-tty.
- `--rm`: indica a Docker que elimine el contenedor cuando su ejecución haya finalizado.
- `--detach` ejecuta el contenedor en segundo plano.
- `--mount src=vol,dst=path_dst`: monta el volumen `vol` en el directorio destino, dentro del contenedor, cuyo directorio está indicado por `path_dst`.
- `--name` nombra el contenedor como **mydb**.
- `--restart=on-failure:n`: permite especificar el número de veces que Docker intentaría reiniciar un contenedor en caso de que termine con un código de error, antes de detenerlo.
- `--restart=always`: reinicia un contenedor cuando termina con un código de error. No hay un número máximo de reinicios.

SWARM INIT

Sirve para inicializar el orquestador de contenedores.

Uso:: `docker swarm init [OPTIONS]`

- `--advertise-addr`: Advertised address (format: <ip|interface>[:port]). Sirve para que los servicios estén a la escucha de una dirección IP, cuando la máquina tiene mas de una.

VOLUME CREATE

Crea un volumnen.

Uso: `docker volume create [OPTIONS] [VOLUME]`

- `--label`: Set metadata for a volume
- `--name`: Specify volume name

VOLUME LS

Lista los volúmenes existentes.

VOLUME PRUNE

Elimina los volúmenes de datos no utilizados por ningún contenedor. Esto no elimina los ficheros del host.

FICHERO DOCKERFILE

FORMATO

```
# Comentarios
INSTRUCCIÓN argumentos
```

Instrucciones

- `FROM imagen`: inicializa la imagen indicada para instrucciones posteriores.
- `WORKDIR directorio`: establece el directorio por defecto donde se ejecutarán todos los comandos (RUN, CMD, ENTRYPOINT, COPY and ADD) dentro del contenedor.
- `COPY fuente dir_destino`: copia los contenidos fuente del host al directorio de destino del contenedor. Si no se pone nada, se llevarán al directorio de trabajo del contenedor.
- `RUN comando`: ejecuta cualquier comando en la imagen. El resultado de la ejecución es permanente en la imagen y puede ser usado por otros comandos posteriores.
- `EXPOSE puerto/[protocolo]`: Informa a docker que el contenedor escuchará en el puerto especificado cuando esté corriendo. En si, la instrucción no hace nada, pues no abre el puerto, y sólo se hace para propósitos informativos.
- `ENV nombre_var valor`: Crea dentro del contenedor una variable de entorno con el nombre y el valor especificado.
- `CMD [comando, paramentro1, parametro2, ...]` Ejecuta un comando cuando el contenedor se lanza. Tiene mas formas sintácticas, parecidas a la pura línea de comandos de RUN.

FICHERO DOCKER-COMPOSE.YML

ESTRUCTURA DE EJEMPLO

```
# versión de la sintaxis.
version: "3"
services:

  # Nombre del servicio. Puede habet tantos bloques de estos como se quiera.
  web:
    # imagen que se va a utilizar
    image: nombre_de_imagen
    deploy:
      replicas: n_replicas
      resources:
        limits:
          cpus: "0.1"
          memory: 50M
      restart_policy:
        condition: on-failure
    ports:
      - "4000:80"
    networks:
 # redes a las que se va a conectar este servicio replicado.
      - webnet

# declaración de redes creadas para este proyecto.
networks:
  webnet:
```

SOLUCIÓN AL EJERCICIO PARA POSITIVO

- Hacer un pull de la imagen httpd
- desplegar un swarm