Titorial Básico de Docker

Docker executa servizos nun entorno aillado dentro do host a partir dunha imaxe, que é unha simple plantilla inmutable. As operacións básicas para xestionar as imaxes e despregar docker son as seguintes:

Xestión de Contedores Docker

Creación de Docker: Crea un Docker a partir dunha imaxe

docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

Exemplos:

docker create –name exemplo-debian debian:trixie
docker create --name webtest -p 8080:80 nginx
docker create --name cliente1 --network rede_proba alpine tail -f /dev/null # O start executa tail, e o CMD
docker create --name app_local -e TZ=Europe/Madrid -v /home/franro/app:/usr/share/nginx/html nginx

Arranque do Docker: Arranca o Docker creado

docker start NAME

Exemplo:

docker start exemplo-debian
docker start exemplo-debian > /dev/null 2>&1

Execución dun comando nun Docker: Executa un comando no interior do docker.

docker exec NAME COMMAND

Exemplos:

docker exec webtest echo “hola mundo”
docker exec -it webtest /bin/bash #-it permite interactuar co teclado e pantalla co docker

Creacion, Arranque e Execución: descarga da imaxe, creación, arrancada e execución.

docker run NAME COMMAND

Exemplos:

docker run debian:trixie echo "hola mundo!"
docker run -it debian:trixie /bin/bash && sleep infinity

Parada do Docker: Para o Docker creado

docker stop NAME

Exemplo: docker stop exemplo-debian

Lista de Docker: Amosa os dockers do sistema

docker ps [-a] *# -a lista todos, incluso os parados e finalizados*

Exemplo: docker ps -a

Borrado de Docker: Elimina un Docker

docker rm NAME

Limpeza de Docker: Elimina todos os Docker parados

docker container prune

Logs dun Docker: Elimina un Docker

docker logs NAME

Información dun Docker: Obten información dun Docker

docker inspect NAME

Xestión de Imaxes

Descarga de imaxes: Descarga a imaxe de dockerhub e a almacena localmente

docker image pull / docker pull [OPTIONS] NAME[:TAG|@DIGEST]

Exemplo: docker pull nginx:1.27@sha256:9e742cc4f67d0d93e4a58c8c69cb21b629b65b34dc7c9fae2c10c13d0f07b8b8

Listado das imaxes locais: Lista as imaxes dispoñibles no sistema

docker image list [--digests]

Exemplo: docker image list

Borrado de imaxe local: Elimina unha imaxe do sistema

docker image rm NAME

Exemplo: docker image rm nginx

Limpeza de imaxes locais: Elimina as imaxes do sistema que non teñen uso

docker image prune

Exemplo: docker image prune

Subida de imaxes: Sube unha imaxe a dockerhub

docker image push [OPTIONS] NAME[:TAG]

Exemplos:

```
[ docker login ]
docker tag myapp:latest <username>/myapp:latest          # tag latest
docker image push <username>/myapp:latest
[docker logout]
```

Etiquetado de imaxes: Etiqueta unha imaxe para categorizala

docker image tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

Exemplos:

```
docker tag myapp:latest franro/myapp:stable
docker tag myapp:latest franro/myapp:dev
```

Información dunha imaxe: Facilita información dunha imaxe

docker image inspect [OPTIONS] IMAGE [IMAGE...]

Historial dunha imaxe: Facilita información sobre o que se fixo sobre unha imaxe

docker history [OPTIONS] IMAGE

Backup dunha imaxe: Fai un backup dunha imaxe en formato .tar

docker image save [OPTIONS] IMAGE [IMAGE...]

Exemplos:

```
docker image save -o nginx.tar nginx:latest
docker image save -o web_stack.tar  nginx:latest  mysql:8.0  redis:7
docker image save nginx:latest | gzip > nginx.tar.gz
```

Recuperación do Backup: Recupera un backup feito con *docker image save*

docker image load [OPTIONS]

Exemplos:

```
docker image load -i nginx.tar
cat nginx.tar | docker image load
gunzip -c nginx.tar.gz | docker image load
```

Backup parcial dunha imaxe: Fai un backup dunha imaxe en formato .tar sen as capas, historial nin instrucións de Dockerfile, so o sistema de ficheiros final

docker export [OPTIONS] CONTAINER

Exemplos:

docker export meu_contedor > rootfs.tar
docker export -o rootfs.tar.gz meu_contedor
docker export meu_contedor | gzip > nginx.tar.gz

Recuperación dun Backup parcial dunha imaxe: Recupera un backup feito con docker export

docker image import [OPTIONS] file|URL|- [REPOSITORY[:TAG]]

Exemplos:

docker import rootfs.tar meu_rootfs:1.0
docker import https://exemplo.com/rootfs.tar.gz meu_rootfs:latest
cat rootfs.tar | docker import – meu_rootfs:dev
gunzip -c rootfs.tar.gz | docker import – meu_rootfs:gz

docker import --change "CMD ['/bin/bash']" --change "ENV DEBUG=true" rootfs.tar meu_rootfs:debug

Xestión de Volumes

Un **volume** é un almacén de datos externo á capa de lectura da imaxe. Está pensado para persistir datos máis alá da vida dun contedor, compartir datos entre contedores e evitar que os datos queden empacados nas capas da imaxe.

- Os datos de volumes non forman parte da imaxe (non incrementan o tamaño da mesma).
- Por defecto os named volumes gardanse no host en: `/var/lib/docker/volumes/<nome>/_data` (ruta do daemon).
- Podes usar volumes nomeados, bind mounts (montaxes do host) ou tmpfs.

Os volumes poden ser de varios tipos:

1. ***Named volume*** (driver local por defecto) — xestionado por Docker, ex.: -v datos:/var/lib/mysql.
2. ***Bind mount*** — montaxe un directorio do host: /ruta/host:/ruta/container.
3. ***tmpfs*** — sistema de ficheiros en memoria (RAM), con --tmpfs ou --mount type=tmpfs.
4. ***Volume driver / plugins*** — NFS, Gluster, Longhorn, rexray, etc. Substitúen ou amplían o local driver.
5. ***Anonymous volum***e — Se crean automaticamente por VOLUME no Dockerfile ou -v /path sen nome; pódense acumular se non se limpan.

Creación dun Volume: Crea un volume

docker volume create [OPTIONS] [VOLUME]

Exemplo:

```
docker volume create datos_db
docker volume create --label entorno=proba volume_proba
docker volume create \
  --driver local \
  --opt type=none \
  --opt device=/mnt/datos_nginx \
  --opt o=bind \
  volume_nginx
docker volume create \
  --driver local \
  --opt type=nfs \
  --opt o=addr=192.168.1.100,rw \
  --opt device=:/export/backups \
  volume_nfs
```

Uso do Volume: Uso dos volumes nos docker

Exemplos:

```
docker run -d --name web  -v volume_nfs:/usr/share/nginx/html   nginx
docker run -d --mount type=volume,source=myvol,target=/data,readonly  busybox
docker run --rm -v pgdata:/data -v "$(pwd)":/backup  busybox sh -c "cd /data && tar czf /backup/pgdata.tar.gz ."
docker run --rm -v pgdata:/data -v "$(pwd)":/backup  busybox sh -c "cd /data && tar xzf /backup/pgdata.tar.gz --strip 1"
```

Información dun Volume: Obten información dun volume

docker volume inspect [OPTIONS] VOLUME [VOLUME...]

Lista de Volumes: Obten a lista de volumes

docker volume ls [OPTIONS]

Borrado de Volumes: Elimina un volume

docker volume rm [OPTIONS] VOLUME [VOLUME...]

Limpeza de Volumes: Elimina os volumes que non estan a ser utilizados

docker volume prune [OPTIONS]

Xestión de Redes

Docker permite que os contedores **se comuniquen entre si e co exterior** a través dun sistema de redes virtuais.
Cando se instala docker se verán estas dúas redes:

1. ***bridge*** (driver: bridge) — Os contedores se conectan a un bridge sobre o que se fai NAT
2. ***host (driver: host)***— Os contedores usan a pila de rede do host estando expostos a rede local e compartindo a IP do host e a súa interface
3. ***none*** – Sen conexión de rede

Ademais de bridge e host, é posible usar outros drivers diferentes como ***macvlan***, ***overlay*** (vxlan) ou outro tipo de plugins.

Listar as Redes: Amosa as redes existentes

docker network ls [OPTIONS]

Creación dunha Rede: Crea un volume

docker network create [OPTIONS] NETWORK

Exemplos:

docker network create rede_proba
docker network create --driver bridge rede_web
docker network create --driver bridge --subnet 172.28.0.0/16 --gateway 172.28.0.1 rede_app

Conexión de Docker a rede: Uso dos volumes nos docker

Exemplos:

docker run -d --name web1 --network mybridge nginx
docker run -d --name device --network macvlan_net --ip 192.168.1.50 myapp
docker run -d --name app --network mynet --network-alias api myimage

Conexión en vivo dun docker: Conectar un docker existente a unha rede

docker network connect [OPTIONS] NETWORK CONTAINER

Exemplo: docker network connect mybridge some_running_container

Desconexión en vivo dun docker: Desconectar un docker existente dunha rede

docker network disconnect [OPTIONS] NETWORK CONTAINER

Exemplo: docker network disconnect mybridge some_running_container

Eliminar redes: Elimina unha rede

docker network rm NETWORK [NETWORK...]

Información dunha rede: Obten información das redes

docker network inspect [OPTIONS] NETWORK [NETWORK...]

Creación dun Servizo de Xeito Manual

1. Descargamos e lanzamos un docker base cos parámetros axeitados. Neste caso imos crear un servizo NGINX e creamos volumes para que a configuración persista a actualización do servizo

docker run -it -d --name debian-nginx -v nginx_etc-nginx:/etc/nginx -v nginx_var-www:/var/www debian:trixie sleep infinity

2. Entramos no contedor actualizamos o sistema e instalamos o software

docker exec -it debian-nginx /bin/bash
root@95335e3191ec:/# apt update
root@95335e3191ec:/# apt install iproute2 vim nginx ssh inetutils-ping
root@95335e3191ec:/# exit

3. Paramos o contedor

docker stop debian-nginx

4. Creamos unha nova imaxe a partir dos datos do contedor

docker commit debian-nginx web-server
docker image list

5. Creamos e iniciamos o contedor a partir da imaxe, mapeando os volumes e o porto http

docker create --name mywebserver-01 -p 80:80 -v nginx_etc-nginx:/etc/nginx -v nginx_var-www:/var/www web-server nginx -g "daemon off;"
docker start mywebserver-01

E posible facelo máis directo:

docker run --name nginx-template -v nt_etc:/etc/nginx -v nt_var:/var/www -p 80:80 -it debian:trixie /bin/bash && sleep infinity

> **root@95335e3191ec:/# apt update**
> **root@95335e3191ec:/# apt install iproute2 vim nginx ssh inetutils-ping**
> **root@95335e3191ec:/# exit**

docker commit debian-nginx web-server
docker image list
docker create --name mywebserver-01 -p 80:80 -v nginx_etc-nginx:/etc/nginx -v nginx_var-www:/var/www web-server nginx -g "daemon off;"
docker start mywebserver-01

Dockerfiles

Docker-Compose