

Documento Detallado: Comandos Docker

Índice

1. Introducción a Docker y Terminología
 2. Gestión de Contenedores
 3. Gestión de Imágenes
 4. Volúmenes y Persistencia de Datos
 5. Redes en Docker
 6. Dockerfile: Instrucciones Principales
 7. Variables de Entorno: ARG y ENV
 8. Docker Compose
 9. Comandos de Limpieza y Optimización
 10. Comandos Avanzados y Contextos
 11. Comandos de imágenes para: commit, pull, push y tag
 12. Práctica 1 - Dockerfile
 13. Práctica 2 - Ejercicio Completo
 14. Resumen de Comandos Útiles
-

1. Introducción a Docker y Terminología

Docker es una plataforma para desarrollar, transportar y ejecutar aplicaciones mediante contenedores, que encapsulan todo lo necesario para funcionar de forma aislada y portable. Docker usa imágenes y contenedores:

- **Imagen**: plantilla inmutable con todo el código y dependencias.
 - **Contenedor**: instancia en ejecución de una imagen.
 - **Dockerfile**: archivo que indica cómo construir una imagen personalizada.
 - **Volumen**: espacio persistente para datos fuera del contenedor.
 - **Red**: conexión virtual para intercambiar datos entre contenedores.
-

2. Gestión de Contenedores

- `docker run` : crea y ejecuta un nuevo contenedor desde una imagen.
 - Parámetros útiles:
 - `-d` : ejecuta en segundo plano (detached)
 - `-it` : modo interactivo con terminal
 - `--name` : asigna nombre
 - `-p host:contenedor` : mapea puertos

- `-v vol:/ruta` : monta volumen
- `-e VAR=valor` : define variable entorno

Ejemplo completo : `docker run -d --name pruebadocker -p 8080:80 -v /home/alumno/documentos:/var/www/html apache`

- `docker start <container>` : inicia un contenedor detenido
 - `docker start pruebadocker`
- `docker stop <container>` : detiene un contenedor
 - `docker stop pruebadocker`
- `docker rm <container>` : elimina un contenedor detenido
 - `docker rm pruebadocker`
- `docker ps -a` : lista contenedores activos (`-a` para todos)
- `docker exec -it <container> <cmd>` : ejecuta comando interactivo dentro del contenedor (ejemplo: bash)
 - `docker exec -it pruebadocker /bin/bash`
- `docker logs <container>` : muestra la salida de registros generados por el contenedor
- `docker rename <container> <nuevo_nombre>` : cambia el nombre del contenedor
 - `docker rename pruebadocker pruebadockernuevonombre`

3. Gestión de Imágenes

- `docker pull <imagen>` : descarga una imagen desde Docker Hub
 - `docker pull apache`
- `docker build -t nombreImagen .` : crea nueva imagen siguiendo instrucciones de Dockerfile actual
 - `docker build -t imagenapache_php .`
- `docker images` : lista imágenes disponibles localmente
- `docker rmi <imagen>` : borra imagen local
- `docker commit <container> <nueva_imagen>` : genera nueva imagen desde contenedor modificado
 - `docker commit pruebadocker imagen1`
- `docker push <imagen>` : sube imagen a registro remoto
 - `docker push imagen1`
- `docker search <término>` : busca imágenes en Docker Hub
- `docker login/logout` : inicia/cierra sesión registro Docker Hub

4. Volúmenes y Persistencia de Datos

- `docker volume create <nombre>` : crea un volumen persistente.
 - `docker volume create volumendatosphp`
- `docker run -v volumen:/ruta ...` : monta volumen en contenedor.
 -
- `docker volume ls` : lista volúmenes.
- `docker volume rm <nombre>` : elimina volumen.
- **Tipos de volumen:**
 - *Named volume*: gestionado por Docker
 - *Bind mount*: carpeta host mapeada dentro del contenedor
- Ejemplo:
text

`docker run -v /datos:/app/datos ubuntu`

•

5. Redes en Docker

- `docker network create <red>` : crea una red virtual personalizada.
 - `docker network create red1`
 - `docker run --network <red> ...` : conecta contenedor a una red específica.
 - `docker run -d - -name dockerprueba - -network red1 ...`
 - `docker network ls` : lista redes existentes.
 - `docker network inspect <red>` : detalles de una red.
 - `docker network inspect red1`
 - `docker network connect (nombre red) (nombre contenedor)` : Para conectar contenedor existe a una red
 - `docker network connect red1 apachephp`
 - **Utilidad:** Agrupar contenedores para comunicación privada y servicios complejos como balanceo de carga.
-

6. Dockerfile: Instrucciones Principales

Un Dockerfile define los pasos para construir una imagen a medida. Principales comandos:

- `FROM` : imagen base
- `LABEL` : metadata
- `ADD` y `COPY` : copia de archivos al contenedor
- `RUN` : ejecuta comandos para instalar dependencias
- `WORKDIR` : define directorio de trabajo
- `USER` : usuario que ejecuta instrucciones siguientes

- **EXPOSE** : documenta puertos sugeridos
 - **ENV** : define variable de ambiente
 - **ARG** : variable de construcción
 - **CMD / ENTRYPOINT** : comando principal del contenedor
- Ejemplo:

```
text
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN npm install --production
EXPOSE 3000
CMD [ "node", "index.js" ]
```

7. Variables de Entorno: ARG y ENV

- **ARG nombre** : define variables sólo para la etapa de build.
 - **ENV nombre valor** : variable persistente en el contenedor,
 - Pueden usarse para configuración, credenciales o cualquier parámetro dinámico.
-

8. Docker Compose

Herramienta para definir y lanzar aplicaciones de varios servicios (multi-contenedor) con archivos YAML.datacamp+1

- Fichero típico: **docker-compose.yml**
- Ejemplo básico:

```
text
version: '3.9'
services:
  db:
    image: mysql:5.7
    environment:
      - MYSQL_ROOT_PASSWORD=clave
  web:
    image: nginx:latest
    ports:
      - "8080:80"
```

- Comandos más usados:
 - `docker-compose up` : inicia los servicios definidos
 - `docker-compose down` : detiene y elimina servicios
 - `docker-compose ps` : lista servicios activos
-

9. Comandos de Limpieza y Optimización

- `docker system prune` : limpia contenedores, imágenes, caché y redes no utilizados,
 - `docker image prune [-a]` : elimina imágenes no usadas (-a para todas).
 - `docker container prune` : borra contenedores detenidos
 - `docker volume prune` : elimina volúmenes no usados.
 - `docker rm $(docker ps -a -q)` : elimina todos los contenedores parados.
 - `docker rmi $(docker images -q)` : borra todas las imágenes locales.
-

10. Comandos Avanzados y Contextos

- `docker context` : gestiona contextos para cambiar entre distintos entornos Docker (máquinas, servidores, configuraciones).
 - `ls` : lista contextos
 - `inspect` : muestra detalles
 - `create` : crea nuevo contexto
 - `use` : activa contexto
 - `docker system info / docker info` : muestra detalles a bajo nivel sobre configuración Docker.
-

11. Comando de imágenes subir o bajar a dockerHub

- `docker comit` : como en git para gestionar versiones de la imagen ([más info página 42 y 43](#)).
 - `-a` :" Autor"
 - `-m` : "Comentario"
 - Luego va el id o nombre del contenedor

- Luego va usuario_dockerhub/nombre_imagen:[version_imagen]
"version_imagen se refiere al **TAG**"
 - **docker tag** : creamos un nuevo tag a una imagen (una misma imagen puede tener varios tag)
 - **docker save** : Copia de seguridad de imágenes(docker save nombre_imagen > copiaSeguridad O docker save -o copiaSeguriadad.tar nombre_imagen)
 - **docker login** : Iniciar sesión en docker hub
 - **docker push** : Despues el nombre de la imagen
 - **docker pull** : Igual que push pero hace falta iniciar sesión
-

12. Práctica 1 - Dockerfile

```
nano Dockerfile
# Usar una imagen base oficial y ligera de nginx
FROM nginx:alpine

# Copia los archivos del sitio al directorio público de nginx
COPY index.html /usr/share/nginx/html/index.html
COPY about.html /usr/share/nginx/html/about.html

# Expone el puerto 80
EXPOSE 80

cd actividad1/
568 docker build -t ejercicio1_v1 .
569 docker images
570 docker run --rm -d -p 8080:80 ejercicio1_v1
571 docker ps
574 docker login
575 docker tag ejercicio1_v1 elpolsv/ejercicio1_v1
576 docker push elpolsv/ejercicio1_v1
```

12. Práctica 2 - Ejercicio Completo

- docker network create red1
 - docker run -d -p 8080:80 --name apachephp --network red1 -v C:/Users/pablo/Documents/libreria/libreria:/var/www/html php:7.4-apache
 - docker exec -it apachephp /bin/bash
 - Dentro de este contenedor ejecutar :
 - docker-php-ext-install mysqli
 - docker-php-ext-install pdo
 - docker-php-ext-install pdo_mysql
 - apt-get update
 - apt-get install -y sendmail libpng-dev libzip-dev zlib1g-dev libonig-dev
 - rm -rf /var/lib/apt/lists/*
 - docker-php-ext-install zip
 - docker-php-ext-install mbstring
 - docker-php-ext-install zip
 - docker-php-ext-install gd
 - a2enmod rewrite
 - service apache2 restart
 - docker start apachephp
 - docker run -d --name mysql1 --network red1 -v misdatos:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=bbdd -e MYSQL_USER=pablo -e MYSQL_PASSWORD=bbdd -e MYSQL_DATABASE=mibbdd mysql:5.6
 - docker run -d --name phpmyadmin1 --network red1 -e PMA_HOST=mysql1 -p 8081:80 phpmyadmin
-

13. Resumen de Comandos Útiles

Comando	Descripción
docker run	Crea y ejecuta nuevo contenedor
docker start/stop/restart	Control del ciclo de vida de contenedores
docker exec	Ejecuta comando dentro de contenedor
docker ps	Lista contenedores en marcha
docker images	Lista imágenes locales
docker pull/build/push	Gestión de imágenes (descargar, construir, subir)
docker rm/rmi	Elimina contenedores o imágenes

docker volume/network create	Crea volúmenes y redes
docker-compose up/down	Gestiona aplicaciones multicontenedores
docker system prune	Limpieza general del sistema
docker logs	Visualiza logs del contenedor

14. Good Luck