Introducción a los contenedores

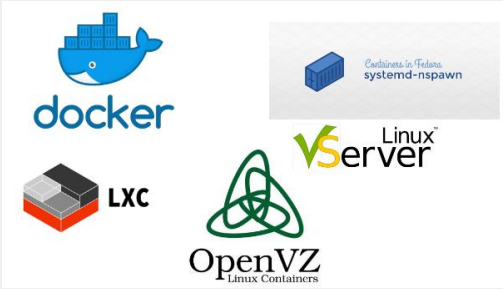
Entornos de Desarrollo

IES Benigasló

2023

docker

Presentacíon



- **Andrei Micleusanu Micleusanu**
- IES Benigasló
- Correo: a.micleusanu@edu.gva.es

Docker es la implementación de contenedores de software más popular.

- Images
- Dockerfiles
- Containers
- Registry
- Volumes

Conceptos básicos de Docker

01 **Images**
Una imagen es un archivo inerte e inmutable que es básicamente un snapshot de un contenedor. Similar al concepto de *ISO*. Versiones.

02 **Dockerfiles**
Un Dockerfile es el archivo en el que definimos las instrucciones necesarias para crear una imagen de Docker.

03 **Containers**
Un contenedor es una imagen de docker en funcionamiento.

04 **Registry**
El docker registry es un repositorio donde se suben las imágenes de docker. Docker nos ofrece el repositorio DockerHub para ser usado gratuitamente.

05 **Volumes**
El almacenamiento es efímero y cuando el contenedor se pare los datos internos desaparecen. Docker permite montar volúmenes que persisten más allá de la vida del contenedor.

Parte B – Práctica de Docker

- 01 Docker cli: Docker run
- 02 Dockerfile: ¿Qué es? - Creación Dockerfile
- 03 Docker build
- 04 Docker cli: Docker run
- 05 Docker Hub
- 06 Docker push
- 07 Docker run from Docker Hub

01 Docker cli: docker run

Docker-cli

Línea de comandos

Docker cli es el software que nos permite ejecutar
comandos con Docker.

Docker container run

Lanzar contenedores

Con docker container run podemos desplegar contenedores.
https://docs.docker.com/engine/reference/run/

```
$ docker
Usage: docker [OPTIONS] COMMAND [ARG...]
       docker [ --help | -v | --version ]

A self-sufficient runtime for containers.

Options:
      --config string      Location of client config files (default "/root/.docker")
  -D, --debug              Enable debug mode
      --help               Print usage
  -H, --host value         Daemon socket(s) to connect to (default [])
  -l, --log-level string   Set the logging level ("debug"|"info"|"warn"|"error"|"fatal") (default "info")
      --tls                Use TLS; implied by --tlsverify
      --tlscacert string   Trust certs signed only by this CA (default "/root/.docker/ca.pem")
      --tlscert string     Path to TLS certificate file (default "/root/.docker/cert.pem")
      --tlskey string      Path to TLS key file (default "/root/.docker/key.pem")
      --tlsverify          Use TLS and verify the remote
  -v, --version            Print version information and quit

Commands:
    attach    Attach to a running container
    # [...]
```

01 Docker cli: docker run

Lanzad este comando: `docker container run -p 80:80 nginx:alpine`

- - docker container run: lanza el contenedor
- - p 80:80: mapea el puerto 80 del contenedor con el el host
- - nginx:alpine: el nombre de la imagen a lanzar
- - Ir a: http://localhost

01 Docker cli: docker run

Lanzad este comando:

- docker container run: lanza el contenedor
- p 80:80: mapea el puerto 80 del contenedor con el del host
- nginx:alpine: el nombre de la imagen a lanzar
- Ir a: http://localhost

Ejercicio 1

- ¿Cómo lanzar un contenedor en segundo plano?
- ¿Cómo ver los contenedores que se están ejecutando?
- ¿Cómo ver logs de un contenedor en segundo plano?
- Lanzad tres contenedores de **nginx** en diferentes puertos.

Ejercicio 1

- **¿Cómo lanzar un contenedor en segundo plano**?
  - docker container run -d ...
  - docker run -d ...
- **- ¿Como ver los contenedores que se están ejecutando?**
  - docker container ls
  - docker ps
- **- ¿Como ver logs de un contenedor en segundo plano?**
  - docker container logs “id del contenedor”
  - docker logs “id del contenedor”
- **- Lanzad tres contenedores de nginx en diferentes puertos**
  - docker container run -d -p 80:80 nginx:alpine
  - docker container run -d -p 81:80 nginx:alpine
  - docker container run -d -p 82:80 nginx:alpine

02 Dockerfile: Qué es y la creación

Un Dockerfile es un fichero que nos permite crear imágenes de docker a nuestro gusto, basadas en otras imágenes.

https://docs.docker.com/engine/reference/builder/

Dockerfile conceptos:

- FROM: la primera instrucción de un Dockerfile indica cual es la imagen base que utilizamos para nuestro Dockerfile.
- ADD: añade ficheros al contenedor durante su construcción.
- RUN: ejecuta comandos dentro del contenedor.

```
FROM debian:stretch-slim

ENV NGINX_VERSION 1.13.6-1~stretch

RUN apt-get update

EXPOSE 80 443

CMD ["nginx", "-g", "daemon off;"]
```

02 Dockerfile: Creando nuestro Dockerfile

- Creemos un contenedor de nginx:alpine pero desplegando nuestra web. Necesitamos crear un archivo que se llame “Dockerfile”.

```
FROM nginx:alpine

ADD index.html /usr/share/nginx/html/
```

- Necesitamos crear (en el mismo directorio) el archivo index.html. Este archivo puede contener cualquier código html.

Docker cli: docker image build

Para crear imágenes a partir del dockerfile

Para poder construir nuestras imágenes, necesitamos ejecutar el comando "docker image build".

La sintaxis es la siguiente:

 # docker image build [-t nombre-de-la-imagen] [path contexto del dockerfile]

Ejemplo 1:

```
docker image build -t andrei-micle-amm:latest .
```

Ejemplo 2.

```
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx:alpine
 ---> 315798907716
Step 2/2 : ADD index.html /usr/share/nginx/html/
 ---> da6a99d9ef58
Successfully built da6a99d9ef58
Successfully tagged alvaro-racero-uma:latest
```

02 Dockerfile: Ejercicio Dockerfile – 2

Instrucciones:

- Queremos lanzar un contenedor con una aplicación propia.
  Esta aplicación no es más que un script en **sh** con un echo
  (ver captura abajo).
- El contenedor tomará de imagen base alpine:latest, copiará nuestra aplicación a /bin/nombredelaaplicación

y luego declarará el comando a ejecutar en el contenedor.
(Instrucción CMD de Dockerfile).

```
#!/bin/sh
echo "hola andrei micleusanu"
```

04 Docker cli : docker run – nociones avanzadas

- Desde docker run siempre podemos sobreescribir el comando especificado en el Dockerfile.

```
alvaroracero@MLGPC159:~/uma/echo$ docker run alvaro-racero-echo echo "Test para UMA"
Test para UMA
```

- También podemos decir que el contenedor se ejecute de forma interactiva y conectado a nuestro terminal con las opciones “-ti”.

```
alvaroracero@MLGPC159:~/uma/echo$ docker run -it alvaro-racero-echo /bin/sh
/ #
```

02 Dockerfile: Ejercicio Dockerfile – 3

Instrucciones:

- Queremos lanzar un contenedor con con el comando “curl” instalado. Este contenedor estará basado en Ubuntu. Después, queremos poder ejecutar este contenedor para lanzar comandos “curl” a distintas webs.

- Pista “CMD” no nos vale para esto.
  FROM xxxxx
  ENV http_proxy http://proxy.laboratorios.ac.uma.es:3128
  ENV https_proxy http://proxy.laboratorios.ac.uma.es:3128
  XXX

04 Docker Hub

Instrucciones:

- Docker Hub
  Docker Hub es un docker registry público donde subir nuestras imágenes
- Docker hub nos permite registrarnos gratuitamente y subir nuestros contenedores de forma pública.
- Exploración de repositorios oficiales.
- Crear una cuenta en https://hub.docker.com/

Step 1 Registro en Docker Hub y creación de nuestro primer repositorio PÚBLICO.

Step 2 Uso del comando docker login para loguear nuestra pc en ese repositorio.

Step 3 Cambiar el tag de nuestro contenedor para ajustarlo al repositorio recién creado y subir el contenedor a Docker Hub con “docker push”.

04 Docker Hub

Instrucciones:

- Docker Hub
  Docker Hub es un docker registry público donde subir nuestras imágenes
- Docker hub nos permite registrarnos gratuitamente y subir nuestros contenedores de forma pública.
- Exploración de repositorios oficiales.
- Crear una cuenta en https://hub.docker.com/

Step 1 Registro en Docker Hub y creación de nuestro primer repositorio PÚBLICO.

Step 2 Uso del comando docker login para loguear nuestra pc en ese repositorio.

Step 3 Cambiar el tag de nuestro contenedor para ajustarlo al repositorio recién creado y subir el contenedor a Docker Hub con “docker push”.