# Docker

Bernat Costa

# Docker como herramienta de trabajo

Docker nos permitirá "virtualizar" varias bbdd en un mismo equipo

Un contenedor por servicio

Podremos tener varios mysql, sqlservers, apaches... en una misma máquina funcionando en puertos distintos

# ¿Porque docker y no VirtualBox?

- Más ágil de levantar máquinas
- No replica todo el SO, el kernel se aprovecha el del anfitrión
- Contenedores más libianos que Maquinas virtuales
- Contenedores preconfigurados.

# ¿Que debo saber de docker?

- Imagenes y contenedores
- Puertos
- Variables de entorno
- Volumenes

# Imagenes y contenedores

Con docker vamos a construir contenedores a partir de una imagen.

La imagen es la base de nuestro contentenedor.

Docker nos garantiza que una imagen, se comportará igual en cualquier host.

Los contenedores son volatiles. Cuando se paran y destruyen, los datos se pierden.

Un contenedor corre un servicio ( una bbdd, un servidor web, un servidor de correo....)

Parámetros de configuración

# ¿De donde sacamos las imagenes?

Las podemos crear con un fichero Dockerfile y un comando de docker ( docker build).

A diferencia del año pasado, vamos a centrarnos en la creación de imagenes y publicarlas

Hub.docker.com

# El Fichero Dockerfile

- Es un fichero de texto que pondremos en nuestro proyecto web, en el raiz.
- Cuando lancemos el comando docker build . Estaremos ejecutando los comandos de ese fichero.
- Deberemos seleccionar una imagen de partida, con el comando FROM, luego con el comando COPY podremos añadir ficheros de nuestro ordenador a nuestra imagen, para luego poder exponer un puerto con el EXPOSE

# Exemplo de Dockerfile

- FROM ubuntu/apache2
- COPY ./index.html /var/www/html/
- EXPOSE 80
-

# Dockerfile para compilar y desplegar un proyecto con java con maven.

- FROM maven:3-openjdk-18 as builder
- RUN mkdir -p /build
- WORKDIR /build
- COPY pom.xml /build
- RUN mvn -B dependency:resolve dependency:resolve-plugins
- COPY src /build/src
- RUN mvn package

- FROM tomcat:8.5-jdk11-openjdk-slim
- COPY --from=builder /build/target/daw.war /usr/local/tomcat/webapps/
- EXPOSE 80
- CMD ["catalina.sh", "run"]

# El orden en el Dockerfile

- Docker estructura sus contenedores en capas. Cada línea del Dockerfile es una capa.
- Si cambiamos algo en el código de un fichero que se añade a nuestra imagen al principio, volverá a realizar todas las líneas siguientes del Dockerfile.
- Si cambiamos algo en el código de un fichero que se añade al final, solo rehará aquellas líneas posteriores al cambio de ese fichero.
- Conclusión, hacer los apt installs  o resolver las dependencias mvn antes de pasar nuestro código. Eso va a tardar y mejor hacerlo una vez y no cada vez que tocamos una línea de código.

# Docker push

- Con docker push, podremos publicar nuestra imagen en docker hub.
- Antes tendremos que crear el repositorio en docker hub y ponerle el mismo nombre a la imagen.
- Ejemplo
  - Mi repositorio en docker hub se llama minombre/webestatica
  - Tendré que lanzar:
    - Docker build . -t minombre/webestatica
    - Docker push
  - Con docker image ls podemos ver las imágenes en nuestro ordenador. Si no le pasamos el –t y un nombre, docker nos asigna un nombre al azar.

# ¿Cómo corremos un contenedor?

| | |
|---|---|
|  Arrancar un apache: | `docker run  httpd` |
|  En background | `docker run –d httpd` |
|  Ponerle un nombre al contenedor | `docker run -d --name apachealpine  httpd` |
|  Tags | `docker run -d --name apachealpine  httpd:alpine` |

# Que podemos hacer con un contenedor

- Ver la lista de contenedores corriendo
  - Docker ps
- Ver el log de un contenedor
  - Docker logs nombredelcontenedor ( o el hash)
- Parar/arrancar/borrar un contenedor:
  - Docker stop/start/rm

# Puertos

- Para que el contenedor sea útil, debemos exponer puertos del contenedor en la máquina anfitrión.
- ¿De que nos sirve un apache si no nos da una web?
- Con el parámetro -p "enlazamos" el puerto de nuestra máquina con el del contenedor.
  - `docker run -d --name apachealpine -p 81:80 httpd:alpine`
- IMPORTANTE: cada contenedor, tendrá sus puertos abiertos. Mirar la documentación en hub.docker.com

# Variables de entorno del contenedor

- Hay imagenes que te permiten pasarle variables de entorno, por ejemplo, el pass de root de un sgbbdd.
  - `docker run -d --name mysql -e MYSQL_ROOT_PASSWORD=root -p 3306:3306 mysql:5.7.32`
- Ahora, con un cliente de mysql, podriamos conectarnos al localhost:3306 con password root.

# Volúmenes

Los contenedores son volatiles. No son persistentes

Las bbdd son persistentes.

¿Como podemos montar una bbdd en un contenedor volatil?

Parámetro v

```
docker run -d  --name mysql -e
MYSQL_ROOT_PASSWORD=root -p 3306:3306  -v
mysql_data:/var/lib/mysql mysql:5.7.32
```

# Docker-compose

- Es una herramienta para levantar y parar contenedores de forma amigable
- Son scripts, donde pondremos toda la configuración, y así nos ahorramos de poner todos los comandos.
- Se basa en un fichero docker-compose.yml
- Se arranca/para con
  - `Docker-compose up -d`
  - Docker-compose down

# Ejemplos docker compose

Un wordpress con su bbdd

Una bbdd Microsoft SQLServer persistente.

# Docker compose con línea de backup

docker-compose.yml

```yaml
version: "3.2"
services:

  sql-server-db:
    container_name: sql-server-db
    image: mcr.microsoft.com/mssql/server:2017-latest
    ports:
      - "1533:1433"
    environment:
      SA_PASSWORD: "12345Ab##"
      ACCEPT_EULA: "Y"
    volumes:
        - ./backup:/var/opt/mssql/backup
        - ./data:/var/opt/mssql/data

```

# Ejercicio

- Crea un contenedor propio con una web tuya estática.
- Pruébalo en tu máquina
- Publícala en docker hub.