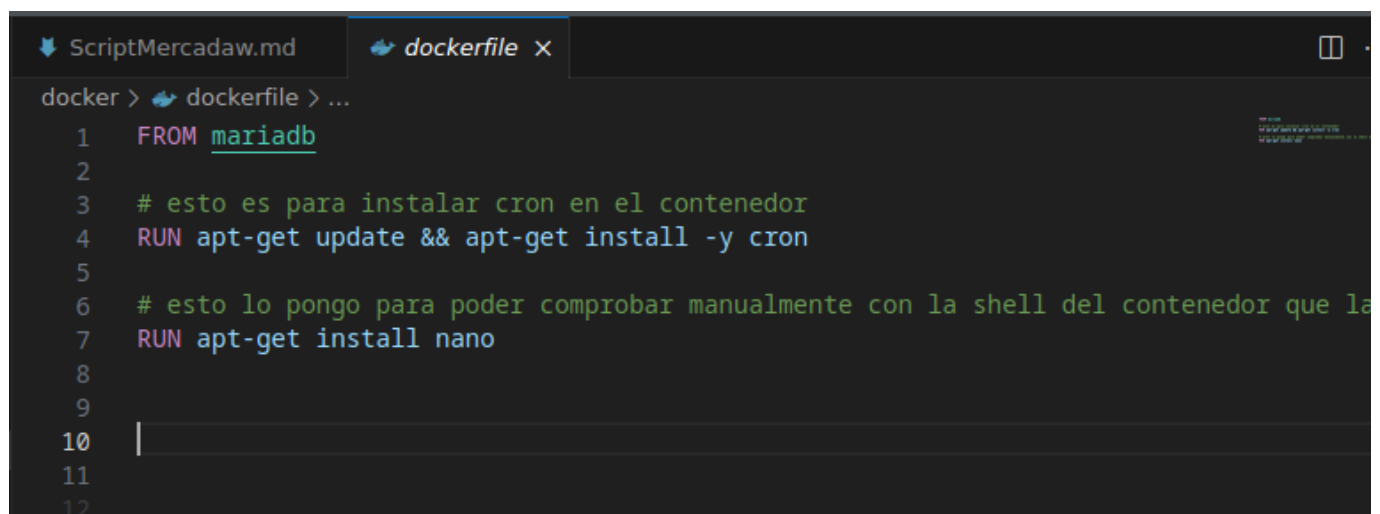


# Script proyecto Mercadaw

El script del que te vamos a hablar a continuación se encuentra en la carpeta "**docker**" en nuestro proyecto. También hemos copiado todo lo relacionado con el script en la carpeta "**sistemas**" para que puedas echar un ojo a los archivos, pero **si ejecutas el script desde esta última no funciona**, por lo que si lo quieres probar debes lanzarlo situado en **docker**.

Antes de hablarte del script cabe destacar cómo está montado el contenedor de la base de datos, puesto que además de la base de datos hemos incluido un par de cosas más para instalar en ella cuando se lance el comando **docker compose up**:



```
docker > dockerfile > ...
1 FROM mariadb
2
3 # esto es para instalar cron en el contenedor
4 RUN apt-get update && apt-get install -y cron
5
6 # esto lo pongo para poder comprobar manualmente con la shell del contenedor que la
7 RUN apt-get install nano
8
9
10 |
11
12
```

- actualizamos los paquetes del contenedor con **apt-get update**
- instalamos cron para poder programar tareas en el docker.
- instalamos nano para, posteriormente, ver si se ha realizado correctamente la tarea mandada por el script que vamos a comentar a continuación.

Partiendo de esto, cabe comentar que en el archivo **docker-compose.yml** tenemos un volumen que nos permite interactuar de forma "remota" con el contenedor y guardar archivos de este y en este, es decir, es una carpeta que actúa como un espejo y permite interactuar con el contenedor desde la raíz de nuestro proyecto:

```
version: '3.1'

services:
  db:
    build:
      context: .
      dockerfile: Dockerfile
    restart: always
    ports:
      - '3306:3306'
    expose:
      - '3306'
    env_file:
      - ".env"
    volumes:
      - .:/sql
    environment:
      - ALLOW_EMPTY_PASSWORD=yes
      - MARIADB_USER=${MARIADB_USER}
      - MARIADB_DATABASE=${MARIADB_DATABASE}
      - MARIADB_PASSWORD=${MARIADB_PASSWORD}
      - MARIADB_ROOT_PASSWORD=${MARIADB_ROOT_PASSWORD}
```

En este caso la carpeta espejo es **sql**, la cual se creará en el contenedor.

Partiendo de esto vamos a desglosar el script poco a poco:

```
ScriptMercadaw.md  docker-compose.yml  scriptInsercionDatos.sh x
docker > $ scriptInsercionDatos.sh
1  #!/bin/bash
2
3  # Levanto el contenedor
4  docker compose up -d
5
6  # Esperar 16 segundos para asegurarse de que el contenedor esté listo
7  sleep 15
8
9  # Abre la terminal del contenedor y pasa los datos de ficheroCargaDatos.sql a la base de de datos MercaDAW
10 docker exec -it docker-db-1 /bin/bash -c "
11   cd sql &&
12   mariadb -u root -psecret MercaDAW < ficheroCargaDatos.sql
13 "
14
15 # Crea una tarea crontab en el contenedor, la cual copia los datos de la base de datos en el fichero copiaSeguridad.sh cada 1 minuto
16 docker exec -it docker-db-1 /bin/bash -c 'echo "* * * * * cd /sql && mariadb-dump -u root -psecret MercaDAW > copiaSeguridad.sql" | crontab -'
17
18 # Inicio los servicios crontab del contenedor
19 docker exec -it docker-db-1 /bin/bash -c "service cron start"
20
21
```

1. Lo primero es que al ejecutar el script se levanta el contenedor y se activa.
2. Tras esto se "duerme" 15 segundos, por si tiene que buscar la imagen en internet y descargarla que de tiempo a que se realice el proceso completo.
3. **docker exec -it docker-db-1 /bin/bash -c** nos permite abrir la terminal del contenedor, que en este caso se llama el contenedor **docker-db-1**, y tras abrirla ejecuta en él **cd sql** para situarse en la carpeta que es un espejo y **mariadb -u root -psecret MercaDAW < ficheroCargaDatos.sql** para insertar en la base de datos **MercaDAW** los datos de **ficheroCargaDatos.sh**, con el usuario **root** y la contraseña **secret**.

4. Tras esto volvemos a llamar a la terminal del contenedor y ejecutamos `'echo " * * * * cd /sql && mariadb-dump -u root -psecret MercaDAW > copiaSeguridad.sql" | crontab -'*`, lo cual crea una "tarea" crontab que vaya a la carpeta "sql" del contenedor y ejecute el comando de copia de seguridad de base de datos tipo mariadb y lo escriba en **copiaSeguridad.sql**. El tipo de archivo lo buscamos en internet (el de la copia de seguridad donde guardar los datos). El de inserción de datos y el de copia de seguridad es parecido, con la diferencia de que uno importa los datos a la database y otro los exporta, siempre poniendo el usuario, contraseña y la base de datos con la que operar.
5. Por último, volvemos a llamar a la terminal del docker y le pedimos que inicie los servicios de cron, para que comience el proceso de copias de seguridad.

El archivo de carga de datos y el de copia de seguridad ambos están en nuestra carpeta "docker" del proyecto, pues sin estar en esa carpeta no funcionaría el script.

Por último utilicé de forma manual la shell del docker para ejecutar **crontab -e**, y así pude comprobar que la tarea indicada en el script estaba realmente "escrita". Si quieres probar que funciona situate en la raíz del proyecto y realiza los siguientes pasos en la terminal: **cd docker -> bash scriptInsercionDatos.sh**

*NOTA: CUANDO SE EJECUTA EL DOCKER TE QUEDAS SIN INTERNET. DEBE EJECUTARSE EN UN S.O. LINUX*

**Realizado por Jorge Alfonso, Eddyson, Sergio y Adrian Nuñez**