

Docker - Desarrollo web con Docker

Docker Network y Composer de contenedores (YAML)

Docker Network

Docker establece una red aislada del host, solo para contenedores.

Para conectar contenedores en la red de Docker no se requiere asignar puertos; basta indicar los IDs o nombres de los contenedores.

Para ver las redes que crea Docker:

Docker network ls



Docker Network

Crear una nueva red de Docker:

Docker network create [nombre_red]

Al arrancar (run) contenedores se debe incluir ***--net [nombre_red]*** para añadirlo a la red de Docker creada.

Ejemplo:

Docker network create mongo-network

Nota: Recomendación para próximos ejemplos:

docker pull mongo-express:0.49

docker pull mongo:4.2.1




Variables de entorno de imagenes - run -e

Algunas imágenes pueden incluir variables de entorno configurables. Por ejemplo, para sobrescribir usuario y contraseña por defecto en una base de datos.

Estas variables se indicarán en su documentación y se pueden gestionar con el flag **-e** al arrancar el contenedor con la imagen (run).

Ejemplo para MongoDB:

```
docker run -p 27017:27017 -d  
-e MONGO_INITDB_ROOT_USERNAME=admin  
-e MONGO_INITDB_ROOT_PASSWORD=password  
-e MONGO_INITDB_DATABASE=user-account  
--name mongo --net mongo-network  
mongo
```




Docker Network

Para añadir contenedores a la red, al arrancar con run les indicamos la misma red.

Ejemplo: añadir Mongo Express para poder gestionar la base de datos MongoDB de la red Docker (PhpMyAdmin para MongoDB):

```
docker run -d -p 8081:8081  
-e ME_CONFIG_MONGODB_ADMINUSERNAME=admin  
-e ME_CONFIG_MONGODB_ADMINPASSWORD=password  
-e ME_CONFIG_MONGODB_SERVER=mongo  
--name mongo-express  
--net mongo-network  
mongo-express
```

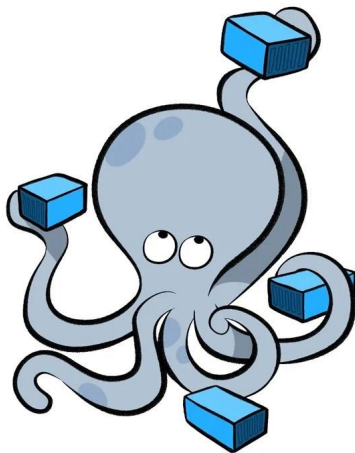
Para acceder a Mongo Express en localhost:8081 usar admin / pass.
Acceder y crear una colección (collection) llamada "users" (tabla).



Docker Compose

Como hemos visto, arrancar los contenedores y posteriormente gestionarlos vía comandos puede resultar una tarea tediosa.

Docker Compose nos ayuda a facilitar esta tarea definiendo instrucciones sobre **cómo deben arrancar múltiples contenedores y trabajar** entre ellos.



Docker Compose vs Run

RUN command para mongoDB:

```
docker run -d  
--name mongo  
-p 27017:27017  
-e MONGO_INITDB_ROOT_USERNAME=admin  
-e MONGO_INITDB_ROOT_PASSWORD=password  
--net mongo-network  
mongo
```

mongo-docker-compose.yaml:

```
version: '3'  
services:  
  mongo:  
    image: mongo  
    ports:  
      - 27017:27017  
    environment:  
      - MONGO_(...)_USERNAME=admin  
      - MONGO_(...)_PASSWORD=password
```

Docker Compose vs Run

RUN command para mongo-express:

```
docker run -d  
--name mongo-express  
-p 8081:8081  
-e ME_CONFIG_MONGODB_ADMINUSERNAME=admin  
-e ME_CONFIG_MONGODB_ADMINPASSWORD=password  
-e ME_CONFIG_MONGODB_SERVER=mongo  
--net mongo-network  
mongo-express
```

mongo-docker-compose.yaml:

```
version: '3'  
services:  
  mongo:  
    image: mongo  
    (...)  
  
  mongo-express:  
    image: mongo-express  
    ports:  
      -8080:8081  
    environment:  
      - ME_(...)_ADMINUSERNAME=admin  
      - ME_(...)_ADMINPASSWORD=password  
      - ME_(...)_SERVER=mongo
```

Al estar en un mismo .yaml o .yml, composer los añade a la misma Docker Network

Docker Compose - mongo-docker-compose.yaml

```
version: '3'
services:
  mongo:
    image: mongo
    ports:
      - 27017:27017
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=password
  mongo-express:
    image: mongo-express
    ports:
      - 8080:8081
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
      - ME_CONFIG_MONGODB_ADMINPASSWORD=password
      - ME_CONFIG_MONGODB_SERVER=mongo
```

Docker Compose - Estructura

Aspectos clave:

- Fichero con extensión `.yaml` o `.yml`.
- ¡La indentación se debe respetar!
- Comienza por “version” siempre; indica la versión de composer a usar.
- Dentro de services se indica el nombre de los contenedores a arrancar.
- Dentro de cada contenedor se define la imagen a usar y las propiedades del contenedor (puertos, variables de entorno, etc).



Docker Compose - Arrancar contenedores

Mediante el comando docker-compose se puede arrancar los contenedores indicados en un fichero .yaml o .yml, indicando que realice un “up”.


docker-compose -f [fichero.yml] up

Ejemplo:

docker-compose -f mongo-docker-compose.yaml up

Al revisar los logs resultantes se puede ver el nombre de la red Docker que se ha creado.

Si accedemos a localhost:8081 veremos que no existe la BD user-accounts y la colección users creada previamente.



Docker Compose - Detener contenedores

Con docker-compose, indicando el fichero .yaml o .yml con los contenedores a detener, pero esta vez indicando “down”.

```
docker-compose -f [fichero.yml] down
```

Ejemplo:

```
docker-compose -f mongo-docker-compose.yaml down
```

Al detener y volver a arrancar el mismo fichero .yaml, docker compose reactiva los contenedores y su red asociada.



Docker Volumes

Mediante Docker Volumes se puede lograr tener datos persistentes en contenedores.

Esto se logra asociando un fichero virtual del contenedor a un fichero real del Host.

Esto implica guardar datos de contenedores de forma local.



Docker Volumes - Tipos

Formas de gestionar Docker volumes:

- **Host volume:** `docker run -v [directorio_fisico:directorio_virtual]` (referencia directa de carpetas)
- **Anonymous volume:** `docker run -v [directorio_virtual]` (Docker se encarga de guardarlo en el Host)
- **Named volume:** `docker run -v [nombre_carpeta:directorio_virtual]` (Permite referenciar el volumen por el nombre dado, sin necesidad de saber la ruta)

Normalmente se almacenan en `.\docker\volumes` del Host.



Docker Volumes - Docker Compose

version: '3'

services:

mongo:

image: mongo

ports:

- 27017:27017

environment:

- MONGO_INITDB_ROOT_USERNAME=admin

- MONGO_INITDB_ROOT_PASSWORD=password

volumes:

- db-data:/data/db # define un named volume y la ruta virtual a guardar (¡ver docs!)

(...)

al final del fichero se debe listar las unidades declaradas

volumes:

db-data:

driver: local # Almacenamiento local en el Host

