04 Docker Compose

Docker Compose es una herramienta que facilita la gestión de aplicaciones que requieren múltiples contenedores en Docker. Permite definir todos los contenedores, redes y volúmenes necesarios en un solo archivo de configuración (usualmente en formato YAML). Con Docker Compose, puedes levantar y gestionar toda la infraestructura de la aplicación con un solo comando, simplificando el proceso de desarrollo y asegurando que los contenedores funcionen juntos de manera coordinada. Es especialmente útil para entornos de desarrollo y pruebas.

A continuación vamos a definir el conjunto de pasos a realizar y una vez realizado vamos a ver como simplificarlo.

Sin docker compose

Docker Hub images

Postgres

pgAdmin

1. Crear un volumen para almacenar la información de la base de datos

```
PS C:\Users\Carballeira\Documents\Docker> docker volume create postgres-db
postgres-db
PS C:\Users\Carballeira\Documents\Docker> docker volume ls
DRIVER    VOLUME NAME
local     postgres-db
```

2. Montar la imagen de postgres

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> -d `
>> --name postgres-db `
>> -e POSTGRES_PASSWORD=123456 `
>> -v postgres-db:/var/lib/postgresql/data `
>> postgres:15.1
Unable to find image 'postgres:15.1' locally
15.1: Pulling from library/postgres
536a8b356450: Download complete
7f0a68628bce: Download complete
bb263680fed1: Download complete
48111fe612c1: Download complete
3ce7f8df2b36: Download complete
87f7b375a7d2: Download complete
75a54e59e691: Download complete
7cca76b73db0: Download complete
```

```
dc1f0e9024d8: Download complete
f30287ef02b9: Download complete
07b5cb2894c7: Download complete
32b11818cae3: Download complete
d9daaa1dc184: Download complete
Digest: sha256:02547253a07e6edd0c070caba1d2a019b7dc7df98b948dc9a909e1808eb77024
Status: Downloaded newer image for postgres:15.1
48e22a64dc018234b5d12f67d6a975530325d0d616d6f4381257cdc02597f558

PS C:\Users\Carballeira\Documents\Docker> docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS
PORTS      NAMES
48e22a64dc01   postgres:15.1   "docker-entrypoint.s…"   8 seconds ago   Up 7
seconds   5432/tcp   postgres-db
```

3. Montar la imagen de pgAdmin

```
PS C:\Users\Carballeira\Documents\Docker> docker container run `
>> --name pgAdmin `
>> -e PGADMIN_DEFAULT_PASSWORD=123456 `
>> -e PGADMIN_DEFAULT_EMAIL=superman@google.com `
>> -dp 8080:80 `
>> dpage/pgadmin4:6.17
Unable to find image 'dpage/pgadmin4:6.17' locally
6.17: Pulling from dpage/pgadmin4
ca7dd9ec2225: Download complete
96528202a796: Download complete
86881266b9c2: Download complete
3376692967d7: Download complete
239c7c08c598: Download complete
352eac98d4d0: Download complete
b613bbab0161: Download complete
ad91e12622a9: Download complete
496dff0917f8: Download complete
1669cc6c32bc: Download complete
2ec496e62b87: Download complete
37eb4951c967: Download complete
8e80feca838b: Download complete
3b35cc859e08: Download complete
Digest: sha256:503f7328901d772c46c819a2e8dc50e0a8755a6cb3d81b2d80c76b9aee35e8e0
Status: Downloaded newer image for dpage/pgadmin4:6.17
2b12dc137dfd3418f12a7831ac3128e0ad95927724724a817bd21157d228bd5e

PS C:\Users\Carballeira\Documents\Docker> docker container ls
CONTAINER ID   IMAGE                COMMAND                  CREATED
STATUS          PORTS                                NAMES
8a3ca2ea6b1a   dpage/pgadmin4:6.17   "/entrypoint.sh"         11 minutes ago   Up
11 minutes   443/tcp, 0.0.0.0:10000->80/tcp   pgAdmin
e16c5da33dc8   postgres:15.1         "docker-entrypoint.s…"   12 minutes ago   Up
12 minutes   5432/tcp
```

4. Ingresar a la web con las credenciales de superman

http://localhost:10000/

5. Intentar crear la conexión a la base de datos

1. Click en Servers
2. Click en Register > Server
3. Colocar el nombre de: "SuperHeroesDB" (el nombre no importa)
4. Ir a la pestaña de connection
5. Colocar el hostname "postgres-db" (el mismo nombre que le dimos al contenedor)
6. Username es "postgres" y el password: 123456
7. Probar la conexión

6. Ohhh no!, no vemos la base de datos, se nos olvidó la red

7. Crear la red

```
PS C:\Users\Carballeira\Documents\Docker> docker network create postgres-net
9aab1d8c097170fdbd4dffc68169de22e246cd5c0e7990dd07755d9a2dfba889
```

8. Asignar ambos contenedores a la red

```
PS C:\Users\Carballeira\Documents\Docker> docker network ls
NETWORK ID     NAME           DRIVER    SCOPE
e30be88d86cf   bridge         bridge    local
0de5fed7fe42   host           host      local
8d2078019d88   none           null      local
9aab1d8c0971   postgres-net   bridge    local
```

9. Conectar ambos contenedores

```
PS C:\Users\Carballeira\Documents\Docker> docker network connect postgres-net 8a3
PS C:\Users\Carballeira\Documents\Docker> docker network connect postgres-net e16
```

10. Intentar el paso 4. de nuevo.

Si logra establecer la conexión, todo está correcto, proceder a crear una base de datos, schemas, tablas, insertar registros, lo que sea.

Con docker compose

A continuación se adjunta el archivo docker-compose.yml.

```
services:
  db:
    container_name: postgres_database
    image: postgres:15.1
    volumes:
      - postgres-db:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=123456

  pgAdmin:
    depends_on:
      - db
    image: dpage/pgadmin4:6.17
    ports:
      - "10000:80"
    environment:
      - PGADMIN_DEFAULT_PASSWORD=123456
      - PGADMIN_DEFAULT_EMAIL=superman@google.com

volumes:
  postgres-db:
    external: true
```

Para lanzar el servicio tenemos el siguiente modo.

```
PS C:\Users\Carballeira\Documents\Docker\Recursos> docker compose up
time="2024-12-17T18:47:50+01:00" level=warning
msg="C:\\Users\\Carballeira\\Documents\\Docker\\Recursos\\docker-compose.yml: the
attribute `version` is obsolete, it will be ignored, please remove it to avoid
potential confusion"
[+] Running 3/3
 ✔ Network recursos_default      Created
0.0s
 ✔ Container postgres_database   Created
0.1s
 ✔ Container recursos-pgAdmin-1  Created
0.1s
Attaching to postgres_database, pgAdmin-1
postgres_database  |
postgres_database  | PostgreSQL Database directory appears to contain a database;
Skipping initialization
postgres_database  |
postgres_database  | 2024-12-17 17:47:51.602 UTC [1] LOG:  starting PostgreSQL
15.1 (Debian 15.1-1.pgdg110+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian
10.2.1-6) 10.2.1 20210110, 64-bit
postgres_database  | 2024-12-17 17:47:51.603 UTC [1] LOG:  listening on IPv4
address "0.0.0.0", port 5432
postgres_database  | 2024-12-17 17:47:51.603 UTC [1] LOG:  listening on IPv6
```

```
address "::", port 5432
postgres_database  | 2024-12-17 17:47:51.607 UTC [1] LOG:  listening on Unix
socket "/var/run/postgresql/.s.PGSQL.5432"
postgres_database  | 2024-12-17 17:47:51.614 UTC [29] LOG:  database system was
interrupted; last known up at 2024-12-17 16:19:24 UTC
postgres_database  | 2024-12-17 17:47:51.763 UTC [29] LOG:  database system was
not properly shut down; automatic recovery in progress
postgres_database  | 2024-12-17 17:47:51.768 UTC [29] LOG:  redo starts at
0/15541C8
postgres_database  | 2024-12-17 17:47:51.768 UTC [29] LOG:  invalid record length
at 0/15542B0: wanted 24, got 0
postgres_database  | 2024-12-17 17:47:51.768 UTC [29] LOG:  redo done at 0/1554278
system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s
postgres_database  | 2024-12-17 17:47:51.772 UTC [27] LOG:  checkpoint starting:
end-of-recovery immediate wait
postgres_database  | 2024-12-17 17:47:51.787 UTC [27] LOG:  checkpoint complete:
wrote 3 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.004 s,
sync=0.002 s, total=0.017 s; sync files=2, longest=0.001 s, average=0.001 s;
distance=0 kB, estimate=0 kB
postgres_database  | 2024-12-17 17:47:51.791 UTC [1] LOG:  database system is
ready to accept connections
pgAdmin-1          | NOTE: Configuring authentication for SERVER mode.
pgAdmin-1          |
pgAdmin-1          | pgAdmin 4 - Application Initialisation
pgAdmin-1          | ======================================
pgAdmin-1          |
pgAdmin-1          | [2024-12-17 17:48:05 +0000] [1] [INFO] Starting gunicorn
20.1.0
pgAdmin-1          | [2024-12-17 17:48:05 +0000] [1] [INFO] Listening at:
http://[::]:80 (1)
pgAdmin-1          | [2024-12-17 17:48:05 +0000] [1] [INFO] Using worker: gthread
pgAdmin-1          | [2024-12-17 17:48:05 +0000] [90] [INFO] Booting worker with
pid: 90
```

NOTA: Si queremos dejar de ejecutar el compose tendremos que hacer `ctrl + C` y ejecutar `docker compose down` para eliminar los servicios creados.

```
PS C:\Users\gilbl\OneDrive\Escritorio\docker> docker compose up -d
```

NOTA-2: Tambien podriamos crear a la misma altura que el archivo.yml una carpeta donde se guarda la información del contenedor con la base de datos.

```
services:
  db:
    container_name: postgres_database
    image: postgres:15.1
    volumes:
      # - postgres-db:/var/lib/postgresql/data
      - ./postgres:/var/lib/postgresql/data
```

```
    environment:
      - POSTGRES_PASSWORD=123456

  pgAdmin:
    depends_on:
      - db
    image: dpage/pgadmin4:6.17
    ports:
      - "10000:80"
    volumes:
      - ./pgAdmin:/var/lib/pgadmin
    environment:
      - PGADMIN_DEFAULT_PASSWORD=123456
      - PGADMIN_DEFAULT_EMAIL=superman@google.com
# volumes:
# postgres-db:
# external: true
```

NOTA-3: Una mejora interesante sería crear un archivo `.env` con las variables de entorno del proyecto.

```
POSTGRES_PASSWORD=123456
PGADMIN_PASSWORD=123456
PGADMIN_EMAIL=superman@google.com
```

```
services:
  db:
    container_name: postgres_database
    image: postgres:15.1
    volumes:
      - postgres-db:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}

  pgAdmin:
    depends_on:
      - db
    image: dpage/pgadmin4:6.17
    ports:
      - "10000:80"
    environment:
      - PGADMIN_DEFAULT_PASSWORD=${PGADMIN_PASSWORD}
      - PGADMIN_DEFAULT_EMAIL=${PGADMIN_EMAIL}

volumes:
  postgres-db:
    external: true
```

Nota-4: Tambien podemos lanzar el docker-compose pasandole las variables en tiempo de ejecucion como se muestra a continuación.

```
services:
  db:
    image: mysql:5.7
    volumes:
      - "./.data/dbwordpress:/var/lib/mysql"
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD:
      MYSQL_DATABASE:
      MYSQL_USER: wordpress
      MYSQL_PASSWORD:
```

```
MYSQL_ROOT_PASSWORD=secret_password MYSQL_DATABASE=rootdesdezero
MYSQL_PASSWORD=000000 docker-compose up -d
```

NOTA-5: Podemos ver la información del docker compose con `docker compose logs` en caso de que hagamos un arranque desatendido.

Otros ejemplos comentados de Docker Compose

Sin red externa definida.

```
version: '3.8' # Versión del esquema de Docker Compose. Asegúrate de que sea
compatible con tu versión de Docker.
services: # Define los servicios que se ejecutarán como contenedores.
  app: # Nombre del servicio (puedes cambiarlo según tus necesidades).
    image: myapp:latest # La imagen de Docker que se usará para este servicio.
    build: # Opcional: Permite construir la imagen desde un Dockerfile.
      context: . # Directorio donde se encuentra el Dockerfile (normalmente el
directorio actual).
      dockerfile: Dockerfile # Nombre del archivo Dockerfile (opcional si es el
predeterminado "Dockerfile").
    ports:
      - "8080:80" # Mapea el puerto 80 del contenedor al puerto 8080 en el host.
    environment: # Variables de entorno para el contenedor.
      - APP_ENV=production # Configuración del entorno de la aplicación.
      - DEBUG=false # Activa o desactiva el modo de depuración.
    volumes:
      - ./app:/usr/src/app # Mapea un directorio local al directorio dentro del
contenedor.
    depends_on:
      - db # Define que este servicio depende del servicio "db".
    restart: unless-stopped # Política de reinicio: reinicia el contenedor excepto
si es detenido manualmente.
```

```
  db: # Servicio para la base de datos.
    image: postgres:15 # Imagen de PostgreSQL, versión 15.
    environment:
      - POSTGRES_USER=admin # Usuario de PostgreSQL.
      - POSTGRES_PASSWORD=secret # Contraseña para PostgreSQL.
      - POSTGRES_DB=mydatabase # Nombre de la base de datos a crear.
    ports:
      - "5432:5432" # Expone el puerto de PostgreSQL en el host.
    volumes:
      - db_data:/var/lib/postgresql/data # Almacena los datos de la base de datos
en un volumen persistente.

  redis: # Servicio opcional para cache.
    image: redis:7 # Imagen de Redis, versión 7.
    ports:
      - "6379:6379" # Expone el puerto 6379 para Redis.

volumes: # Define volúmenes persistentes compartidos entre los servicios.
  db_data: # Volumen para los datos de la base de datos.
    driver: local # Usa el driver predeterminado de Docker para volúmenes locales.
```

Con red externa definida.

```
version: '3.8' # Versión del esquema de Docker Compose.
services: # Define los servicios que se ejecutarán como contenedores.
  app: # Servicio de la aplicación.
    image: myapp:latest # Imagen de Docker para el servicio "app".
    build: # Opcional: Permite construir la imagen desde un Dockerfile.
      context: . # Directorio donde se encuentra el Dockerfile.
      dockerfile: Dockerfile # Nombre del archivo Dockerfile (si es diferente, se
debe especificar).
    ports:
      - "8080:80" # Mapea el puerto 80 del contenedor al puerto 8080 en el host.
    environment: # Variables de entorno para configurar el contenedor.
      - APP_ENV=production # Configura el entorno de la aplicación.
      - DEBUG=false # Habilita o deshabilita el modo de depuración.
    volumes:
      - ./app:/usr/src/app # Mapea un directorio local al contenedor para
persistencia o desarrollo.
    depends_on:
      - db # Indica que este servicio depende de "db".
    networks:
      - custom-network # Conecta este servicio a la red personalizada "custom-
network".

  db: # Servicio de base de datos.
    image: postgres:15 # Imagen de PostgreSQL, versión 15.
    environment:
      - POSTGRES_USER=admin # Usuario de PostgreSQL.
      - POSTGRES_PASSWORD=secret # Contraseña de PostgreSQL.
      - POSTGRES_DB=mydatabase # Nombre de la base de datos inicial.
```

```
    ports:
      - "5432:5432" # Expone el puerto 5432 del contenedor al host.
    volumes:
      - db_data:/var/lib/postgresql/data # Volumen persistente para los datos de
la base de datos.
    networks:
      - custom-network # Conecta este servicio a la red personalizada "custom-
network".

  redis: # Servicio de cache opcional.
    image: redis:7 # Imagen de Redis, versión 7.
    ports:
      - "6379:6379" # Expone el puerto 6379 del contenedor al host.
    networks:
      - custom-network # Conecta este servicio a la red personalizada "custom-
network".

volumes: # Define volúmenes persistentes compartidos entre los servicios.
  db_data: # Volumen para almacenar los datos de PostgreSQL.
    driver: local # Usa el controlador predeterminado para volúmenes locales.

networks: # Define redes personalizadas para los servicios.
  custom-network: # Nombre de la red personalizada.
    driver: bridge # Usa el controlador "bridge" para redes (predeterminado para
redes personalizadas).
```

Propuesta de actividad

Realiza los siguientes pasos:

1. Crea un archivo con extensión .yml que a través de docker compose genere los servicios que se concretan a continuación.
2. Crea los servicios phpmyadmin y una base de datos mysql.
3. La base de datos tendrá que tener un volumen llamado "volumen3".
4. Los datos a indicar en las variables de entorno son de libre elección.
5. Hacer un informe con los pasos explicados en markdown y una imagen de la conexión.