DOCKER-COMPOSE

José Domingo Muñoz

IES Gonzalo Nazareno

Febrero 2021

Escenarios multicontenedores

- Necesitamos varios servicios para que la aplicación funciones: Partiendo del principio de que cada contenedor ejecuta un sólo proceso, sii necesitamos que la aplicación use varios servicios (web, base de datos, proxy inverso, ...) cada uno de ellos se implementará en un contenedor.
- Si tenemos construida nuestra aplicación con microservicios, además cada microservicios se podrá implementar en un contenedor independiente.
- Cuando trabajamos con escenarios donde necesitamos correr varios contenedores podemos utilizar docker-compose para gestionarlos.

Ventajas de usar docker-compose

- Hacer todo de manera **declarativa** para que no tenga que repetir todo el proceso cada vez que construyo el escenario.
- Poner en funcionamiento todos los contenedores que necesita mi aplicación de una sola vez y debidamente configurados.
- Garantizar que los contenedores **se arrancan en el orden adecuado**. Por ejemplo: Mi aplicación no podrá funcionar debidamente hasta que no esté el servidor de bases de datos funcionando en marcha.
- Asegurarnos de que hay **comunicación** entre los contenedores que pertenecen a la aplicación.

DOCKER-COMPOSE

Con apt:

```
apt install docker-compose
```

O con pip:

```
python3 -m venv docker-compose
source docker-compose/bin/activate
(docker-compose) ~# pip install docker-compose
```

El fichero docker-compose.yml

```
version: '3.1'
services:
  wordpress:
    container_name: servidor_wp
    image: wordpress
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: user_wp
      WORDPRESS_DB_PASSWORD: asdasd
      WORDPRESS_DB_NAME: bd_wp
    ports:
      - 80:80
    volumes:
      - /opt/wordpress:/var/www/html/wp-content
  db:
    container_name: servidor_mysql
    image: mariadb
    restart: always
    environment:
      MYSQL_DATABASE: bd_wp
      MYSQL_USER: user_wp
      MYSQL_PASSWORD: asdasd
      MYSQL_ROOT_PASSWORD: asdasd
    volumes:
      - /opt/mysql_wp:/var/lib/mysql
```

Cuando creamos un escenario con `docker-compose` se crea una **nueva red definida por el usuario docker** donde se conectan los contenedores, por lo tanto, se pueden tenemos resolución por dns que resuelve tanto el nombre del contenedor (por ejemplo, `servidor_mysql`) como el alias (por ejemplo, db).

Para crear el escenario:

```
# docker-compose up -d
```

Para listar los contenedores:

```
# docker-compose ps
```

Para parar los contenedores:

```
# docker-compose stop
```

Para borrar los contenedores:

```
# docker-compose rm
```

El comando docker-compose

- `docker-compose up`
- `docker-compose up -d`
- `docker-compose stop`
- `docker-compose run`
- `docker-compose rm`ejecución.
- `docker-compose pause`
- `docker-compose unpause`
- `docker-compose restart`
- `docker-compose down`
- `docker-compose down -v`
- `docker-compose logs servicio1`
- `docker-compose exec servicio1 /bin/bash`
- `docker-compose builddocker-compose.yml`
- `docker-compose top`

Ejercicios

1. Instala docker-compose en tu ordenador. Copia el fichero `docker-compose.yml` de la documentación de la imagen oficial de wordpress.
2. Modifica el `docker-compose.yml` para que use el puerto 8001.
3. Modifica el `docker-compose.yml`, para que la base de datos se llame db_wordpress.
4. Modifica el `docker-compose.yml` para usar bind mount en vez de volúmenes.
5. Levanta el escenario con `docker-compose`.
6. Muestra los contenedores con `docker-compose`.
7. Accede a la aplicación y comprueba que funciona.
8. Comprueba el almacenamiento que has definido y que se ha creado una nueva red de tipo bridge.
9. Borra el escenario con `docker-compose`.