

Contenedores Docker. Docker-compose

1. Apuntes previos:

a) Ejemplo de estructura del fichero `docker-compose.yml`

```
version: '3.1'
services:
  app:
    container_name: guestbook
    image: iesgn/guestbook
    restart: always
    ports:
      - 8080:5000
  db:
    container_name: redis
    image: redis
    restart: always
    volumes:
      - redis:/data
volumes:
  redis:
```

En este fichero YAML se definen los datos relativos a los contenedores de la aplicación como **servicios** (**app** y **db**) y dentro de cada servicio los datos del nombre del **contenedor**, la **imagen** a descargar, la opción de **reiniciarse** si se apaga, las **variables de entorno**, los **puertos** que se exponen y los **volúmenes** de datos persistentes que se van a emplear

Este fichero debe guardarse en una carpeta de nuestro sistema y a continuación desde la consola del sistema, se debe acceder a dicha carpeta para ejecutar el comando que crea los contenedores definidos en el archivo:

```
docker-compose up -d
git: > ejemplo > ➜ docker-compose.yml
      ⌂ Run All Services
1   services:
2     ⌂ Run Service
3     app:
4       container_name: guestbook
5       image: iesgn/guestbook
6       restart: always
7       ports:
8         - 4200:5000
9     ⌂ Run Service
10    db:
11      container_name: redis
12      image: redis
13      restart: always
14      volumes:
15        - redis:/data
16      volumes:
17        redis:
```

b) Opciones habituales de docker-compose (en **negrita** las más típicas):

- **docker-compose up**: Crea los contenedores (servicios) que están descritos en el docker-compose.yml.
- **docker-compose up -d**: Crea en modo detach los contenedores (servicios) que están descritos en el
- **docker-compose stop**: Detiene los contenedores que previamente se han lanzado con docker-compose up.
- **docker-compose run**: Inicia los contenedores descritos en el docker-compose.yml que estén parados.
- **docker-compose rm**: Borra los contenedores parados del escenario. Con la opción -f elimina también los contenedores en ejecución.
- **docker-compose restart**: Reinicia los contenedores. Orden ideal para reiniciar servicios con nuevas configuraciones.
- **docker-compose down**: Para los contenedores, los borra y también borra las redes que se han creado con docker-compose up (en caso de haberse creado).
- **docker-compose down -v**: Para los contenedores y borra contenedores, redes y volúmenes.
- **docker-compose exec servicio1 /bin/bash**: Ejecuta una orden, en este caso /bin/bash en un contenedor llamado servicio1 que estaba descrito en el docker-compose.yml
- **docker-compose build**: Ejecuta, si está indicado, el proceso de construcción de una imagen que va a ser usado en el docker-compose.yml a partir de los ficheros Dockerfile que se indican.

□	●	ejemplo	-	-	-	0.51%	42.08MB / 23.08G	0.36%	791KB / 14G	■	⋮	■
□	●	redis	ce398e59665f	redis		0.23%	4.79MB / 11.54Gi	0.04%	160KB / 0B	■	⋮	■
□	●	guestbook	3b27ea0849af	iesgn/gues	4200:5000	0.28%	37.29MB / 11.54G	0.32%	631KB / 14G	■	⋮	■

c) Volúmenes con docker-compose □ Ejemplo de volumen para mariadb:

```

version: '3.1'

services:
  db:
    container_name: contenedor_mariadb
    image: mariadb
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: asdasd
    volumes:
      - mariadb_data:/var/lib/mysql
  volumes:
    mariadb_data:

```

Se definen en el bloque ultimo volumes: pero además hay que indicar que volumen usara cada contenedor como se indica que se va a crear un volumen mariadb_data apuntando a la carpeta /var/lib/mysql del contenedor

También es posible crear volúmenes en carpetas concretas dentro de nuestro sistema como se muestra:

```

volumes:
  - c:\users\avelino\mariadb_data:/var/lib/mysql

```

2. Se pide ahora desplegar la aplicación Guestbook a partir del fichero `docker-compose.yml` que se muestra en el apartado 1. a). Este fichero lo guardaremos en una carpeta de nuestro usuario de Windows 10 y una vez en la consola accedemos a la carpeta para desplegar la aplicación con el comando `docker compose up -d`. Verificar el acceso a la aplicación con <http://localhost:8080>. Comprobar también desde la vista de los volúmenes de Docker Desktop que el almacenamiento persistente para la base de datos está creado



Guestbook

SUBMIT

3. Crear ahora un fichero `docker-compose.yml` para el despliegue de la aplicación de Temperaturas que se realizó en la práctica anterior, con los siguientes datos:

- a) versión 3.1
 - b) los servicios serán frontend y backend
 - c) para el servicio frontent:
 - a. el contenedor se llamará temperaturas-frontend
 - b. la imagen para usar es avelinopef/temperaturas_front
 - c. se debe reiniciar automáticamente
 - d. El mapeo de puertos es 80:3000
 - e. Este servicio depende de backend (usaremos depends-on)
 - d) Para el servicio backend:
 - a. el contenedor se llamará temperaturas-backend
 - b. la imagen para usar es avelinopef/temperaturas_back
 - c. se debe reiniciar automáticamente

```
1  services:
2    frontend:
3      container_name: temperaturas-frontend
4      image: avelinopef/temperaturas_front
5      restart: always
6      ports:
7        - 81:3000
8      depends_on:
9        - backend
10     backend:
11       container_name: temperaturas-backend
12       image: avelinopef/temperaturas_back
13       restart: always
```

Una vez creado, como en el caso anterior debemos guardarla en una carpeta particular y desplegarlo con `docker-compose up -d`

Verificar desde un navegador que se accede a la aplicación con <http://localhost>

□	●	act7-ejer1	-	-	-	0.89%	76.45MB / 23.08K	0.65%	319.4KB / 0B	□	⋮	刪
□	●	temperaturas e4ffb976df37	avelinopef/	avelinopef/	e4ffb976df37	0.43%	36.04MB / 11.54K	0.31%	16.4KB / 0B	□	⋮	刪
□	●	temperaturas 6a11133b5075	avelinopef/	avelinopef/	6a11133b5075	0.46%	40.41MB / 11.54K	0.34%	303KB / 0B	□	⋮	刪

Temperaturas

roduce el municipio (indica el nombre completo o el inicio) de España que
quieres buscar.

BLISCAP

4. Se pide ahora desplegar la aplicación de Wordpress con docker-compose a partir del fichero wp_mariadb.yml que se entrega.

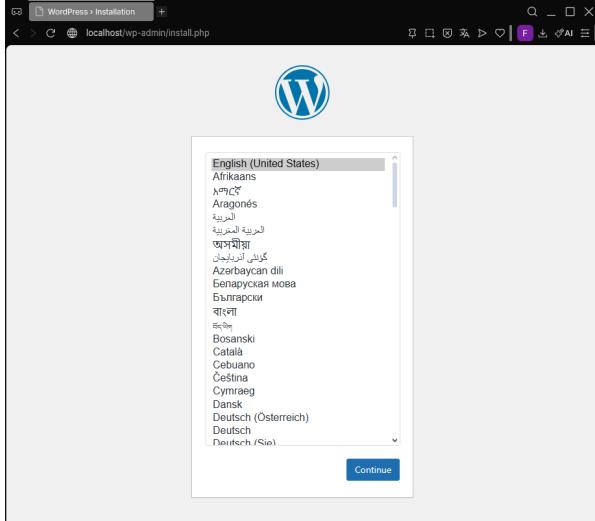
Observa los valores de las variables de entorno y ajusta los valores de modo que:

- El nombre del contenedor de wordpress será contenedor_wp
- El nombre del contenedor de mariadb será contenedor_mariadb
- El usuario para wp en mariadb será admin
- La contraseña del usuario anterior será naranco23
- La base de datos de wp será bbdd_wp
- El volumen de datos para wordpress será datos_wp
- El volumen de datos para mariadb será datos_mariadb

Ya que el fichero no cuenta con el nombre habitual por defecto (docker-compose.yml), la composición debe ser lanzada con la orden:

```
docker-compose -f wp_mariadb.yml up -d
```

Accede al <http://localhost> y completa la instalación de wp



```
act7 > act7-ejer2 > [ ] wp_mariadb.yml
  1 services:
  2   wordpress:
  3     container_name: contenedor_wp
  4     image: wordpress
  5     restart: always
  6     environment:
  7       WORDPRESS_DB_HOST: contenedor_mariadb
  8       WORDPRESS_DB_USER: admin
  9       WORDPRESS_DB_PASSWORD: naranco23
 10      WORDPRESS_DB_NAME: bbdd_wp
 11     ports:
 12       - 80:80
 13     volumes:
 14       - datos_wp:/var/www/html
 15   db:
 16     container_name: contenedor_mariadb
 17     image: mariadb
 18     restart: always
 19     environment:
 20       MYSQL_DATABASE: bbdd_wp
 21       MYSQL_USER: admin
 22       MYSQL_PASSWORD: naranco23
 23       MYSQL_ROOT_PASSWORD: asdasd
 24     volumes:
 25       - datos_mariadb:/var/lib/mysql
 26     volumes:
 27       - datos_wp:
 28       - datos_mariadb:
```

<input type="checkbox"/>	<input checked="" type="checkbox"/>	act7-ejer2	-	-	-	0.03%	152.71MB / 23.06	1.3%	131.5MB / 1	<input type="checkbox"/>	⋮	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	contenedor_v df3f04fe059f	wordpress	80:80 ↗		0.01%	58.54MB / 11.54G	0.5%	99.8MB / 8	<input type="checkbox"/>	⋮	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	contenedor_n e840e73c5156	mariadb			0.02%	94.17MB / 11.54G	0.8%	31.7MB / 2	<input type="checkbox"/>	⋮	<input type="checkbox"/>

5. Ahora desplegaremos una aplicación java sobre un servidor Tomcat y con proxy inverso nginx que permita el acceso a la web de la aplicación. Para ello:

- a) Vamos a utilizar los volúmenes para copiar el archivo sample.war al directorio de publicación de tomcat y el fichero default.conf para copiar una versión modificada del fichero por defecto de nginx. Estos dos archivos deben copiarse una carpeta particular de nuestro sistema (por ejemplo c:\users\usuario\tomcat)
- b) Copiamos a la misma carpeta el fichero tomcat-nginx.yml
- c) Desplegamos como siempre con docker-compose
- d) Verificamos el acceso a la aplicación con <http://localhost>



Sample "Hello, World" Application

This is the home page for a sample application used to illustrate the source directory organization of a web application utilizing the principles outlined in the Application Developer's Guide.

To prove that they work, you can execute either of the following links:

- To a [JSP page](#).
- To a [servlet](#).

```

[Y] tomcat_nginx.yml • [⚙] default.conf
act7 > act7-ejer3 > tomcat_nginx > [Y] tomcat_nginx.yml
  1 services:
  2   aplicacionjava:
  3     container_name: tomcat
  4     image: tomcat:9.0
  5     restart: always
  6     volumes:
  7       - ./sample.war:/usr/local/tomcat/webapps/sample.war:ro
  8   proxy:
  9     container_name: nginx
 10     image: nginx
 11     ports:
 12       - 80:80
 13     volumes:
 14       - ./default.conf:/etc/nginx/conf.d/default.conf:ro
 15

```

<input type="checkbox"/>	<input checked="" type="checkbox"/>	tomcat_nginx	-	-	-	0.06%	147.73MB / 23.0€	1.25%	15.86MB /	<input type="checkbox"/>	⋮	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	tomcat	e4195ada9dd3	tomcat:9.0		0.06%	132.7MB / 11.54€	1.12%	2.56MB / 10	<input type="checkbox"/>	⋮	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	nginx	54fa970eb275	nginx	80:80 ↗	0%	15.03MB / 11.54€	0.13%	13.3MB / 4	<input type="checkbox"/>	⋮	<input type="checkbox"/>

6. Se pide ahora realizar el despliegue con docker-compose de la aplicación phpmyadmin que hará uso de un servidor de bases de datos mariadb. Para ello:

- a) Debe crearse un fichero Docker-compose.yml que permita el desplegar los dos servicios
- b) Se crearán también volúmenes de persistencia de datos para los dos contenedores
- c) Se verificará el acceso a phpmyadmin desde la URL cargada en un navegador

The screenshot shows a web browser window with the URL 'localhost:82'. The page displays the phpMyAdmin login interface. At the top, there is a logo of a sailboat and the text 'phpMyAdmin'. Below it, a banner says 'Bienvenido a phpMyAdmin'. There is a language selection dropdown labeled 'Idioma (Language)' with 'Español - Spanish' selected. Below that is a login form with fields for 'Usuario:' and 'Contraseña:', and a 'Iniciar sesión' button.

<input type="checkbox"/>	<input checked="" type="checkbox"/>	act7-ejer4	-	-	-	0.26%	137.22MB / 23.0€	1.16%	43.6MB / 2	<input type="checkbox"/>	⋮	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	phpmyadmin	e966537e3c8a ↗	phpmyadm	82:80 ↗	0.2%	38.34MB / 11.54€	0.32%	18.5MB / 8	<input type="checkbox"/>	⋮	<input type="checkbox"/>
<input type="checkbox"/>	<input checked="" type="checkbox"/>	mariadb	d4a83348a665	mariadb		0.06%	98.88MB / 11.54€	0.84%	25.1MB / 2	<input type="checkbox"/>	⋮	<input type="checkbox"/>

```

1    ▷Run All Services
2    services:
3        ▷Run Service
4        frontend:
5            container_name: phpmyadmin
6            image: phpmyadmin
7            restart: always
8            ports:
9                - 82:80
10           volumes:
11               - datos_phpmyadmin:/var/www/html
12       ▷Run Service
13       db:
14           container_name: mariadb
15           image: mariadb
16           restart: always
17           volumes:
18               - datos_mariadb:/var/lib/mysql
19       volumes:
20           datos_phpmyadmin:
21           datos_mariadb:

```

7. Se pide ahora desplegar la tienda virtual Prestashop. Para ello se entrega el fichero prestashop.yml y se deben realizar los siguientes cambios:

- a) El usuario de la base de datos para prestashop será user_prestashop
- b) El nombre de la base de datos será mitienda

Se debe acceder a la URL de la tienda y mostrar la aplicación online en el navegador

```
services:
  mariadb:
    image: docker.io/bitnami/mariadb:10.6
    environment:
      - ALLOW_EMPTY_PASSWORD=yes
      - MARIADB_USER=user_prestashop
      - MARIADB_DATABASE=mitienda
    volumes:
      - 'mariadb_data:/bitnami/mariadb'
  prestashop:
    image: docker.io/bitnami/prestashop:8
    ports:
      - '80:8080'
      - '443:8443'
    environment:
      - PRESTASHOP_HOST=localhost
      - PRESTASHOP_DATABASE_HOST=mariadb
      - PRESTASHOP_DATABASE_PORT_NUMBER=3306
      - PRESTASHOP_DATABASE_USER=user_prestashop
      - PRESTASHOP_DATABASE_NAME=mitienda
      - ALLOW_EMPTY_PASSWORD=yes
    volumes:
      - 'prestashop_data:/bitnami/prestashop'
    depends_on:
      - mariadb
  volumes:
    mariadb_data:
      driver: local
    prestashop_data:
      driver: local
```

Docker no encuentra las imágenes, también las busque por el desktop