



UD3.5 Docker Compose

Contenidos

- ❖ ¿Qué es Docker y cómo funciona?
- ❖ Introducción a los contenedores y a Docker
- ❖ Instalación de Docker
- ❖ Primeros pasos con Docker
- ❖ Creación de imágenes personalizadas con Docker.
- ❖ Redes y volúmenes en Docker.
- ❖ **Docker compose.**
- ❖ Utilidades para gestionar Docker fácilmente.

Índice

- ❖ Docker Compose
- ❖ ¿Qué es Docker Compose?
 - Instalación de Docker Compose
- ❖ Formato YAML y Fichero *"docker-compose.yml"*
 - ¿Qué es YAML?
 - ¿Qué es el fichero *docker-compose.yml*?
 - Información especificada en *docker-compose.yml*

Índice

- ❖ Ejemplo de `docker-compose.yml` con Wordpress
 - *“version”*
 - *“services”* y contenedor *“db”*
 - *“services”* y contenedor *“wordpress”*
 - Palabra clave *“volumes”* fuera de las plantillas de contenedores
 - Ejecución del ejemplo Wordpress con *docker-compose.yml*

Índice

❖ Comandos de Docker Compose

- *up*
- *down*
- *build*
- *pull*
- *run*
- *pause / unpause*
- *start / stop*
- *ps*
- *exec*
- *rm*

¿Qué es Docker Compose?

Hasta ahora hemos visto aplicaciones alojadas en contenedores Docker que para ponerse en marcha no utilizaban un único contenedor, sino que requerían el uso de varios contenedores.

El proceso de poner en marcha estas aplicaciones era tedioso, ya que usualmente debíamos levantar “a mano” esos contenedores, respetando el orden de levantado, etc.

“**Docker Compose**” es una aplicación para simplificar la tarea de lanzar múltiples contenedores con una configuración específica y enlazarlos entre sí.

Instalación de Docker Compose

Para instalar/actualizar “Docker Compose”, seguiremos los pasos indicados para nuestro sistema operativo en <https://docs.docker.com/compose/install/>.

Podéis seguir también los pasos indicados en el fichero *instalacion_docker_compose.txt* adjunto en los apuntes de Aules.

Una vez instalado “Docker Compose”, podemos comprobar si la instalación es correcta con:

```
docker-compose --version
```

```
Docker Compose version v2.12.1
```

```
🍏 ~ /Sites/add/add-multiverse/mysql-rds
```

¿Qué es YAML?

YAML es a efectos prácticos una forma de definir información utilizando formato texto. Funciona de forma similar a XML o JSON. Si tenéis experiencia con XML o JSON, en este blog plantean y explican un ejemplo representando la misma información en los 3 formatos:

<https://journey2theccie.wordpress.com/2020/02/28/devnet-associate-1-1-compare-data-formats-xml-json-yaml/>

El uso de YAML dentro de “Docker Compose” es sencillo y fácilmente entendible con cada uno de los ejemplos. Podéis hacer un repaso previo a los principales elementos de YAML utilizados en ficheros “Docker Compose” revisando los ejemplos de YAML en la Wikipedia, disponibles en: [YAML - Wikipedia, la enciclopedia libre](#).

¿Qué es el fichero `docker-compose.yml`?

El fichero “**docker-compose.yml**” es un fichero en formato **YAML** que definirá el comportamiento de cada configuración de “**Docker Compose**”. Lo habitual, es tener ese fichero en la raíz de nuestro proyecto.

Veremos a continuación la especificación para un fichero `docker-compose.yml` y un ejemplo.

Información especificada en docker-compose.yml

version: permite indicar la versión de la especificación del fichero “docker-compose.yml”. No es necesario desde la versión 1.27.0 de “Docker Compose”.

services: array asociativo con las diferentes plantillas de cada contenedor.

build: usado en las plantillas de contenedores. Sirve para indicar si debemos construir la imagen a partir de un “Dockerfile”. Aunque tiene varias formas de uso, detalladas en [Compose file version 3 reference | Docker Documentation](#), la forma más habitual de usarla es indicar en qué directorio está nuestro “Dockerfile” de la forma:

`“build: ./directorio”` o `“build: .”`

Información especificada en docker-compose.yml

command: sobrescribe el comando por defecto a la imagen. Se usa de la forma:

`"command: /bin/bash"`.

container_name: especifica un nombre de contenedor (si no, se generará automáticamente). Se usa de la forma `"container_name: micontenedor"`

depends_on: indica que esta plantilla de contenedor, depende de que se haya creado un contenedor previo de la/s plantilla/s especificada/s. Los servicios también son detenidos en orden inverso a la dependencia. Se pueden ver ejemplos en [Compose file version 3 reference | Docker Documentation](#)

Información especificada en docker-compose.yml

env_file y environment: permite definir variables de entorno en la plantilla del contenedor.

env_file especifica un fichero o lista de ficheros donde están definidas las variables de entorno, similar a "**env_file: .env**"

environment especifica una lista de variables de entorno con su valor.

Ejemplos en: [Compose file version 3 reference | Docker Documentation](#)

expose/ports: permite definir un conjunto de puertos que se exportarán en el contenedor. Ejemplos en [Compose file version 3 reference | Docker Documentation](#) y en [Compose file version 3 reference | Docker Documentation](#).

Información especificada en docker-compose.yml

image: especifica la imagen en la que se basa el contenedor. No es necesario cuando se especifica a partir de un “Dockerfile”.

network_mode: especifica el modo de red, de forma similar al parámetro `--network` de Docker. Los modos soportados se detallan en el siguiente enlace [Compose file version 3 reference | Docker Documentation](#).

networks: define las redes a crear para poner en marcha nuestros contenedores. Ejemplos en [Compose file version 3 reference | Docker Documentation](#)

Información especificada en `docker-compose.yml`

restart: indica cuando debe reiniciarse el contenedor. El valor por defecto es `"no"` (no se reinicia).

Otros valores soportados son `"always"` (se reinicia cuando el contenedor se para) y `"on-failure"` (se reinicia si el contenedor se para y devuelve un valor de salida distinto de cero) y `"unless-stopped"` (se reinicia siempre, excepto si el contenedor es parado manualmente con `"docker stop"`).

- Ejemplos [Compose file version 3 reference | Docker Documentation](#)

tmpfs: establece una lista de directorios a montar en formato `"tmpfs"` en el contenedor.

Ejemplos en [Compose file version 3 reference | Docker Documentation](#)

Información especificada en docker-compose.yml

volumes: establece una lista de volúmenes, ya sea en formato “**bind mount**” o volumen Docker. Si quieres reutilizar un volumen entre distintas plantillas, además debes definir la variable “**volumes**” fuera de las plantillas de contenedores.

Ejemplos [Compose file version 3 reference | Docker Documentation](#)

Ejemplo de docker-compose.yml con Wordpress

```
version: "3.9"
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data:
```

```
version:
services:
  - db
    - imagen
    - volumes
    - environment
  - wordpress
    - depends_on
    - image
    - ports
    - restart
    - environment
  - volumes
```


Ejemplo de docker-compose.yml con Wordpress

❖ up/build

Una vez creado y entendido nuestro fichero “docker-compose.yml”, podemos poner en marcha nuestro servicio situándonos en el directorio donde está este fichero y escribiendo:

docker-compose up -d ó **docker-compose up --build**

Con la opción “**up**” indicamos que se interprete la plantilla definida en “**docker-compose.yml**” y con “**-d**” indicamos la ejecución en segundo plano

Lanzamos el comando y esperamos a que todo se cree basado en el fichero *docker-compose.yml*

Accedemos a <http://localhost:8000>

Comandos de Docker Compose

❖ up

Con el comando “**up**” se interpretará la plantilla “**docker-compose.yml**” y se lanzarán los contenedores necesarios.

Algunos de los usos más típicos son:

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

Interpreta la plantilla y crea la aplicación, pero con el parámetro “-d” actúa en segundo plano.

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

El parámetro “-f” permite indicar un nombre de fichero YAML para “**Docker Compose**”.

```
docker-compose up -d --force-recreate
```

El parámetro “**--force-recreate**” fuerza a reconstruir contenedores incluso si ya existen en tu máquina.

Comandos de Docker Compose

❖ up

```
docker-compose up -d --scale db=3
```

El parámetro “**--scale**” indica que del servicio “**db**” se creen tres contenedores, realizando un escalado local. El propio “**Docker Compose**” se encargará mediante algoritmo “**round robin**” de distribuir las peticiones al host “**db**” (como el nombre del servicio) a cada uno de los contenedores escalados.

Comandos de Docker Compose

❖ down

Con el comando “**down**” se interpretará la plantilla y se paran los contenedores necesarios.

Algunos de los usos más típicos son:

```
docker-compose down
```

Detiene todos los contenedores.

```
docker-compose down -v
```

Elimina los volúmenes creados en la plantilla

```
docker-compose down -t 5
```

Establece un “**timeout**” para la parada de contenedores de 5 segundos. El valor por defecto si no se indica nada es de 10 segundos. Esto se verá más claramente en uno de los casos prácticos propuestos.

Comandos de Docker Compose

❖ **build**

Comando que simplemente crea las imágenes definidas en la plantilla "docker-compose.yml".

❖ **pull**

Comando que descarga las imágenes de contenedores necesarias para "docker-compose.yml".

❖ **run**

Comando que permite lanzar un comando contra un servicio concreto definido en la plantilla "docker-compose.yml".

❖ **pause/unpause**

Con el comando "pause"/"unpause" se pausan/retoman los contenedores de la plantilla.

Comandos de Docker Compose

❖ start/stop

Con el comando “start”/”stop” podemos iniciar/parar sin parámetros todos los servicios de la plantilla o con parámetros, el servicio que deseemos de esa plantilla. También inicia/para sus dependencias, si las tiene.

```
docker-compose start db
```

Pone en marcha el servicio “db” de la plantilla.

```
docker-compose stop db
```

Detiene el servicio “db” de la plantilla.

❖ exec

El comando “exec” es similar a “docker exec”. Nos permite ejecutar un comando los contenedores con una plantilla determinada.

```
docker-compose exec db mysql --version
```

Bibliografía

- ❖ <https://docs.docker.com/compose/>
- ❖ <https://docs.docker.com/compose/install/>
- ❖ <https://docs.docker.com/compose/compose-file/compose-file-v3/>