

# composer

gestor de despliegue de contenedores

## instalación

/via: <https://docs.docker.com/compose/install/#install-compose>

```
sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-  
compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose
```

## yaml

```
user:mate  
curso:docker
```

```
user:  
  nombre: mate  
  curso: docker  
  anyo: 2018
```

```
cursos:  
  - elemento1  
  - elemento2  
  - elemento3
```

```
cursos:  
  clave=valor
```

## docker-compose.yml

```
version:'3.6'  
services:  
  webapp:  
    image: httpd:alpine  
  bd:  
    image: mongo
```

## ejecución

```
# lanza (previa descarga si no estuviese) los contenedores indicados en el .yml  
docker-compose up  
  
# en modo detach
```

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

## comandos

- `docker-compose ps` : solo muestra los contenedores gestionados por docker-compose (y no todos como lo haríamos a través de `docker ps`)
- `docker-compose ps --services` : lista los «servicios» definidos en el `.yaml`
- `docker-compose logs`
  - `-f` : actualización continua
  - `<servicios>` : filtra por servicio (contenedor)... me es igual como se llamen los contenedores
- `docker-compose stop` : para todos los servicios definidos en `.yaml`
  - `<servicios>` : solo para ese servicio

## más docker-compose.yml

```
version: '3.6'
services:
  webapp:
    image: httpd:alpine
    depends_on: bd
  bd:
    image: mongo
```

esto nos asegura que se lanzará primero **bd**, pero igual no con la suficiente cantidad de tiempo

```
version: '3.6'
services:
  elasticsearch:
    image: elasticsearch
  kibana:
    image: kibana
    ports:
      - 5601:5601
```

## comandos

- `docker-compose start` : idem up
- `docker-compose down` : elimina todo lo relacionado con el `.yaml` → containers, volúmenes, redes...
- `docker-compose config` : verifica el fichero `.yaml`
  - `-q` : poco verboso
  - `--services` : muestra servicios
  - `--volumes` : muestra volúmenes

- `docker-compose exec <service> <comando>` : conexión al contenedor `<service>`
  - `docker run -it <service> <comando>`
- `docker-compose images`
  - `-q`

## docker-compose.yml

se puede modificar el fichero mientras están los contenedores, aunque no le gusta si se tocan volúmenes... mejor parar y volver a lanzarlo. Se puede dar el caso que no permita, a través de docker-compose, parar o modificar los contenedores en ejecución

```
version: '3.6'
services:
  webapp:
    build: path_to_Dockerfile
  bd:
    image: mongo
```

### re-builds

si modificamos el Dockerfile, deberemos usar `docker-compose up --build` para recrear la imagen. Si docker-compose tiene las imágenes creadas, no las re-creará

```
version: '3.6'
services:
  webapp:
    build:
      context: .
      ¿file?: Dockerfile
    environment: # mapa o lista
      - ENTORNO=pre
    env_file:
      - ./env
      - ./common.env
    labels:
      VERSION: 1
    restart: always
    links: # deprecated -> para reconocimiento de equipos en la red default
  bd:
    image: mongo
```

CLAVE=VALOR

Last update:  
19/07/2018 09:25 info:cursos:altran:docker:compose <https://miguelangel.torresegea.es/wiki/info:cursos:altran:docker:compose?rev=1532017518>

---

From:  
<https://miguelangel.torresegea.es/wiki/> - **miguel angel torres egea**

Permanent link:  
<https://miguelangel.torresegea.es/wiki/info:cursos:altran:docker:compose?rev=1532017518>

Last update: **19/07/2018 09:25**

