

Ruby on Rails



bitnami

Índice

Índice	2
Introducción	3
Configuración y explicación	3

Introducción

En el siguiente documento vamos a ver cómo se configura, junto con una explicación, un entorno de desarrollo para el framework Ruby on Rails utilizando como base el docker-compose.yml de Bitnami

Configuración y explicación

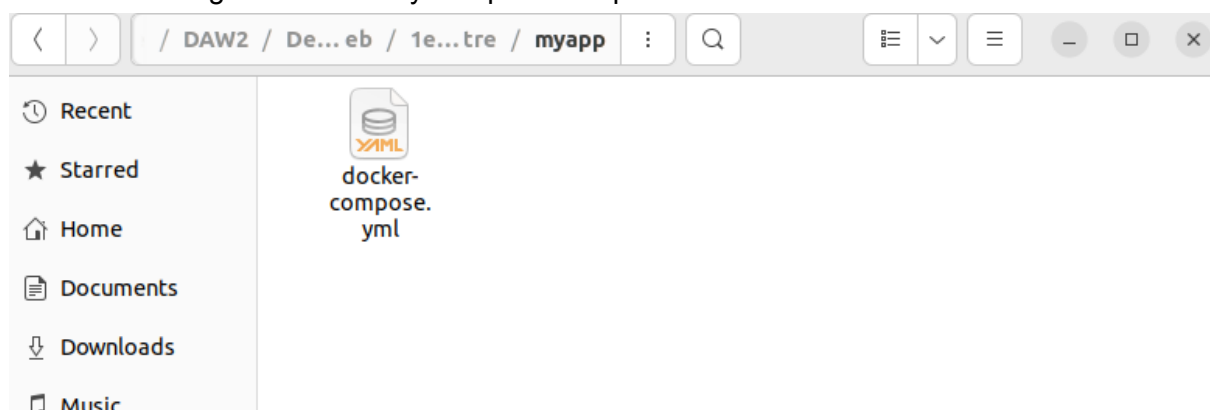
Lo primero que tenemos que hacer es descargarnos el fichero docker-compose.yml Poniendo el siguiente comando nos descargará el fichero en la ruta deseada. Este archivo nos servirá para descargar y configurar los contenedores necesarios para usar Ruby on Rail.

```
estudiante@DAW1:~/DAW2/Despliegue de aplicaciones web/1er Trimestre/myapp$ curl -LO https://raw.githubusercontent.com/bitnami/containers/main/bitnami/rails/docker-compose.yml
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
100	431	100	431	0	0	3074	0
						--:--:--	--:--:--
							3078

En la imagen vemos la información detallada de la descarga del archivo.

Una vez descargado el fichero y comprobado que está correcto



Lo abrimos para cambiar el puerto por el que se lanzará. En el archivo vemos que tenemos una serie de datos que detallaremos. En la imagen vemos la típica estructura de un yaml:

version '2': Esto indica que estamos usando la versión 2 de Docker Compose, y Docker proporcionará las funciones adecuadas.

services: Esta sección define todos los diferentes contenedores que crearemos. En el ejemplo tenemos dos servicios, mariadb y myapp.

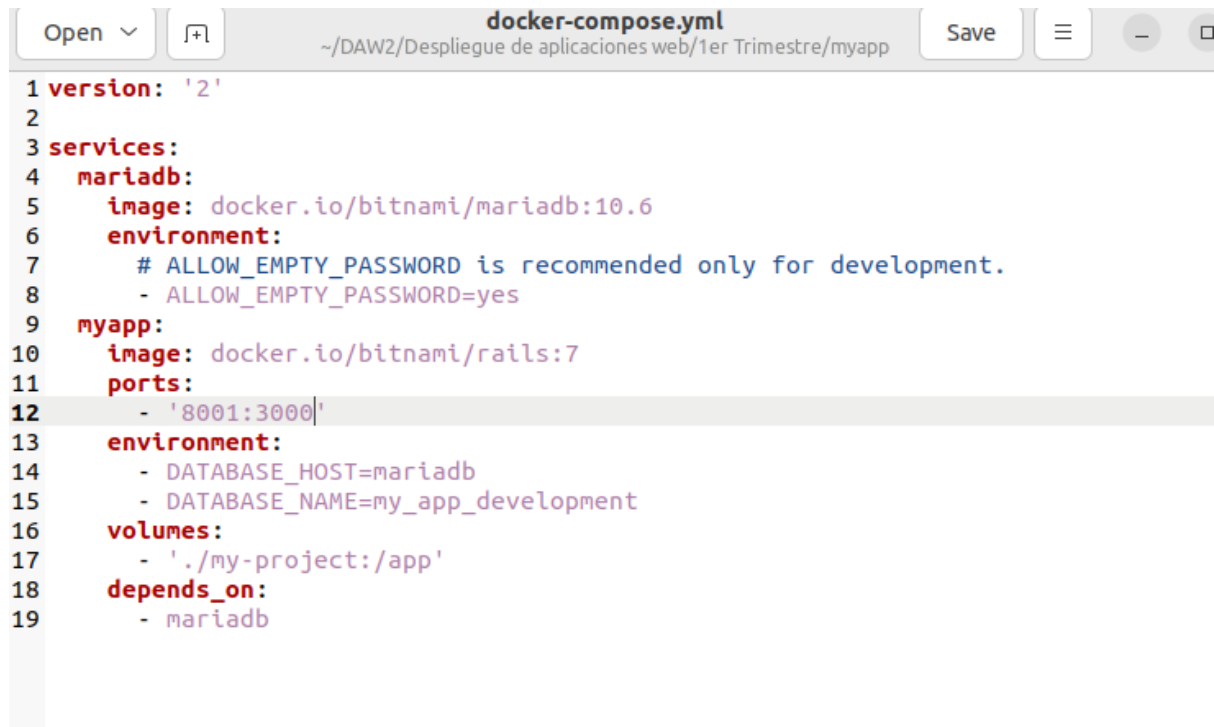
ports: Esto se usa para asignar los puertos del contenedor a la máquina host.

volumes: Cuando se ejecuta el archivo docker-compose monta el archivo de directorios de origen o volúmenes del ordenador, en las rutas de destino dentro del contenedor. Si ya existe la ruta de destino, como parte de la imagen del contenedor, la ruta montada se sobrescribirá.

image: Sirve si no tenemos un Dockerfile y queremos ejecutar un servicio usando una imagen preconstruida, especificamos la ubicación de la imagen usando la cláusula image. Compose bifurcará un contenedor de esa imagen.

environment: La cláusula nos permite configurar una variable de entorno en el contenedor. Este es el mismo -e argumento en Docker cuando se ejecuta un contenedor.

depends_on: Con depends_on se puede controlar el orden de inicio y cierre del servicio. Compose siempre inicia y detiene los contenedores en orden de dependencias.



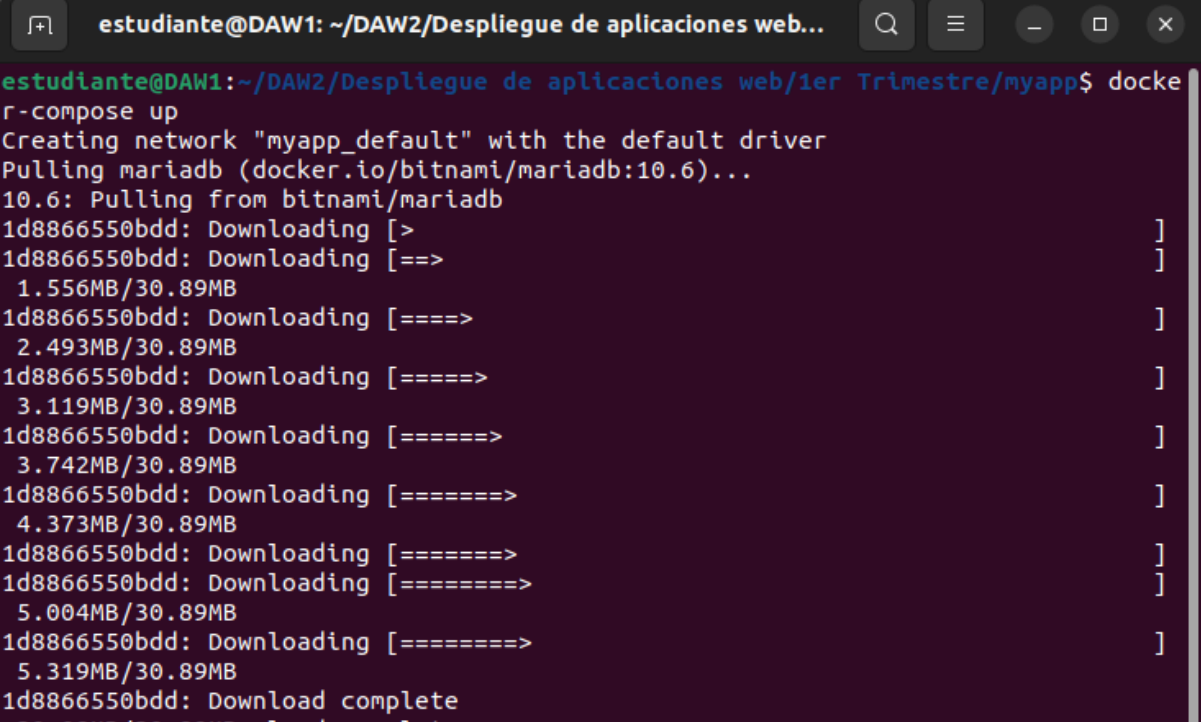
```
1 version: '2'
2
3 services:
4   mariadb:
5     image: docker.io/bitnami/mariadb:10.6
6     environment:
7       # ALLOW_EMPTY_PASSWORD is recommended only for development.
8       - ALLOW_EMPTY_PASSWORD=yes
9   myapp:
10    image: docker.io/bitnami/rails:7
11    ports:
12      - '8001:3000'
13    environment:
14      - DATABASE_HOST=mariadb
15      - DATABASE_NAME=my_app_development
16    volumes:
17      - './my-project:/app'
18    depends_on:
19      - mariadb
```

Una vez visto los detalles del archivo docker-compose.yml y cambiado el puerto a 8001:3000

Ya podemos ejecutar el archivo con docker-compose up.

Esto inicialmente intentará levantar la configuración del archivo pero como es la primera vez que lo ejecutamos lo que hará es descargar y configurar tal como le hemos indicado dentro del documento.

Las demás veces que lo ejecutemos, el programa lo levantará siendo más rápido.



```
estudiante@DAW1: ~/DAW2/Despliegue de aplicaciones web/1er Trimestre/myapp$ docker-compose up
Creating network "myapp_default" with the default driver
Pulling mariadb (docker.io/bitnami/mariadb:10.6)...
10.6: Pulling from bitnami/mariadb
1d8866550bdd: Downloading [>]
1d8866550bdd: Downloading [==>]
1.556MB/30.89MB
1d8866550bdd: Downloading [====>]
2.493MB/30.89MB
1d8866550bdd: Downloading [=====>]
3.119MB/30.89MB
1d8866550bdd: Downloading [=====>]
3.742MB/30.89MB
1d8866550bdd: Downloading [=====>]
4.373MB/30.89MB
1d8866550bdd: Downloading [=====>]
5.004MB/30.89MB
1d8866550bdd: Downloading [=====>]
5.319MB/30.89MB
1d8866550bdd: Download complete
```

Una vez descargado y configurado todo correctamente el programa levantará los contenedores que vienen configurados.

En la imagen nos muestra los contenedores en funcionamiento junto a información de estos. Para parar el contenedor solo necesitamos pulsar Ctrl + c y se pararan todos los contenedores levantados por el docker-compose

```
at 221104 8:03:25
mariadb_1 | 2022-11-04 8:03:25 0 [Note] /opt/bitnami/mariadb/sbin/mysqld: ready for connections.
mariadb_1 | Version: '10.6.10-MariaDB' socket: '/opt/bitnami/mariadb/tmp/mysql.sock' port: 3306 Source distribution
mariadb_1 | 2022-11-04 8:03:25 3 [Warning] Aborted connection 3 to db: 'unconnected' user: 'unauthenticated' host: '172.20.0.3' (This connection closed normally without authentication)
myapp_1 | rails 08:03:25.48 INFO ==> Initializing database (db:prepare)
myapp_1 | rails 08:03:28.98 INFO ==> Database was successfully initialized
myapp_1 | rails 08:03:29.00 INFO ==> ** Rails setup finished! **
myapp_1 |
myapp_1 | => Booting Puma
myapp_1 | => Rails 7.0.4 application starting in development
myapp_1 | => Run `bin/rails server --help` for more startup options
myapp_1 | Puma starting in single mode...
myapp_1 | * Puma version: 5.6.5 (ruby 2.7.6-p219) ("Birdie's Version")
myapp_1 | * Min threads: 5
myapp_1 | * Max threads: 5
myapp_1 | * Environment: development
myapp_1 | * PID: 1
myapp_1 | * Listening on http://0.0.0.0:3000
myapp_1 | Use Ctrl-C to stop
```

Una vez levantado el docker-compose. Ya podemos ir a nuestro navegador para ver que todo funciona correctamente

← → ↻ ⓘ 127.0.0.1:8001 🔍 ☆ ⚙️ □

