3 - PERSISTENCIA Y DOCKER COMPOSE

Docker Compose, build image y Persistencia

- *Realizar un Docker Compose que sea capaz de levantar el servicio de Node realizado en la práctica anterior, construyendo el mismo con el propio Docker Compose, no usando una imágen ya existente. Este servicio deberá incluir un Bind Mount, que permita alterar el contenido del archivo index.js.*
- *El contenido de este ejercicio deberá ser la carpeta conteniendo el Dockerfile, el código de node, así como el docker-compose.yml.*

*** ***NOTA*** ***

No entiendo por qué se solicita un Dockerfile si se solicita *"construyendo el mismo con el* ***propio Docker Compose*** *no usando una imagen ya existente"* pero se dejará de todas formas dicho Dockerfile en la carpeta

Si lo que se buscaba era crear la imagen en docker-compose a través de un Build, dejo captura de como hacerlo acá

En cualquier caso, esta opción o la que expongo llevan al mismo resultado

Como solicitado, se construye el proyecto "*construyendo el mismo con el **propio Docker Compose***" y decido utilizar como base del dockerhub un alpine:3.15

```yaml
version: "3.9"

services:
  custom_node:
    image: alpine:3.15
    container_name: only_docker_compose_container
    volumes:
      - ./helloworld/package.json:/helloworld/package.json
      - ./helloworld/index.js:/helloworld/index.js
    ports:
      - 8080:8080
    command: sh -c "
      apk add --update nodejs npm
      && npm i --prefix /helloworld
      && npm start --prefix /helloworld
      "
```

Al no poder utilizar una imagen personalizada, no puedo utilizar el comando para copiar los archivos como lo haría en **Dockerfile**. Sin embargo, puedo utilizar volúmenes para permitirle a mi instancia de docker acceder a mis archivos.
Por una parte, necesito que el contenedor lea "**package.json**" para instalar las dependencias y controlar el inicio de la aplicación. Por otra parte, debo poder alterar "**index.js**" y mi contenedor debe escuchar dichos cambios.

Para responder al enunciado del ejercicio, decido crear 2 volúmenes con las rutas específicas de los elementos que participan, de esta manera los node_modules quedarán instalados en Docker y no en mi local pero podré alterar a voluntad "**index.js**"
Igualmente configuro los puertos **8080** ya que mi proyecto corre ahí y lo expongo a mi host.

Finalmente, utilizo un comando para:

- Instalar las dependencias del sistema (**nodejs** y **npm**)
- Instalar las dependencias del proyecto vía **npm**
- Lanzar el proyecto con **npm start**.

Ahora en mi carpeta, utilizo en comando “docker-compose up”

```
[  6:53PM ]  [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/actividad-
 $ docker-compose up
Recreating only_docker_compose_container ... done
Attaching to only_docker_compose_container
only_docker_compose_container | fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/main/x86_64/APKINDEX.tar.gz
only_docker_compose_container | fetch https://dl-cdn.alpinelinux.org/alpine/v3.15/community/x86_64/APKINDEX.tar.gz
only_docker_compose_container | (1/8) Installing ca-certificates (20211220-r0)
only_docker_compose_container | (2/8) Installing nghttp2-libs (1.46.0-r0)
only_docker_compose_container | (3/8) Installing brotli-libs (1.0.9-r5)
only_docker_compose_container | (4/8) Installing c-ares (1.18.1-r0)
only_docker_compose_container | (5/8) Installing libgcc (10.3.1_git20211027-r0)
only_docker_compose_container | (6/8) Installing libstdc++ (10.3.1_git20211027-r0)
only_docker_compose_container | (7/8) Installing nodejs (16.14.0-r0)
only_docker_compose_container | (8/8) Installing npm (8.1.3-r0)
only_docker_compose_container | Executing busybox-1.34.1-r3.trigger
only_docker_compose_container | Executing ca-certificates-20211220-r0.trigger
only_docker_compose_container | OK: 58 MiB in 22 packages
only_docker_compose_container |
only_docker_compose_container | added 290 packages, and audited 291 packages in 14s
only_docker_compose_container |
only_docker_compose_container | 47 packages are looking for funding
only_docker_compose_container |   run `npm fund` for details
only_docker_compose_container |
only_docker_compose_container | found 0 vulnerabilities
only_docker_compose_container |
only_docker_compose_container | > helloworld@1.0.0 start
only_docker_compose_container | > nodemon index.js
only_docker_compose_container |
only_docker_compose_container | [nodemon] 2.0.15
only_docker_compose_container | [nodemon] to restart at any time, enter `rs`
only_docker_compose_container | [nodemon] watching path(s): *.*
only_docker_compose_container | [nodemon] watching extensions: js,mjs,json
only_docker_compose_container | [nodemon] starting `node index.js`
only_docker_compose_container | helloworld: listening on port 8080
```

Se levanta el proyecto con éxito y está escuchando en el puerto 8080

NOTA: hasta este punto, se llega al mismo resultado con el Dockerfile

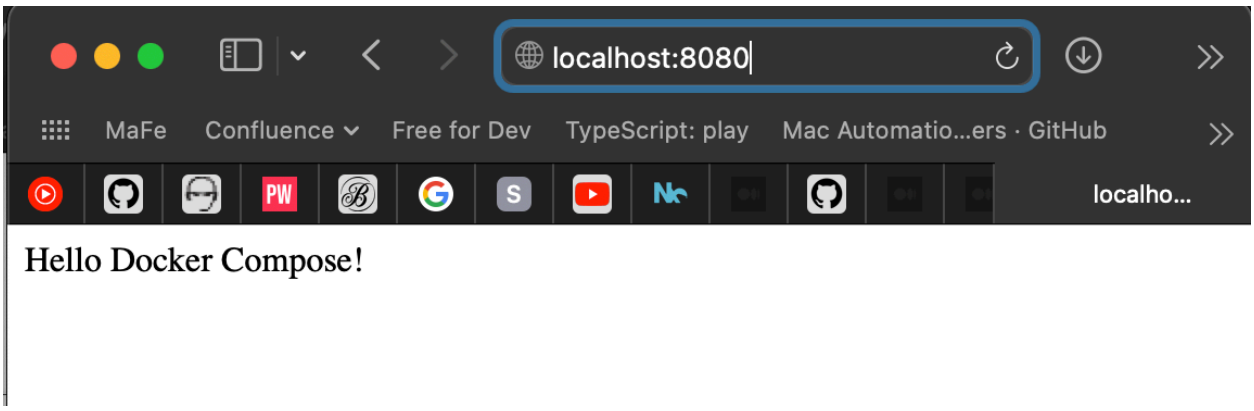
Realizo un cambio en mi archivo “index.js”

```
app.get('/', (req, res) => {
  const name = 'Docker Compose';
  res.send(`Hello ${name}!`);
});
```

Con dicho cambio, se refresca nodemon

```
only_docker_compose_container | [nodemon] starting `node index.js`
only_docker_compose_container | helloworld: listening on port 8080
only_docker_compose_container | [nodemon] restarting due to changes...
only_docker_compose_container | [nodemon] starting `node index.js`
only_docker_compose_container | helloworld: listening on port 8080
```

Mi local muestra “Hello Docker Compose!”



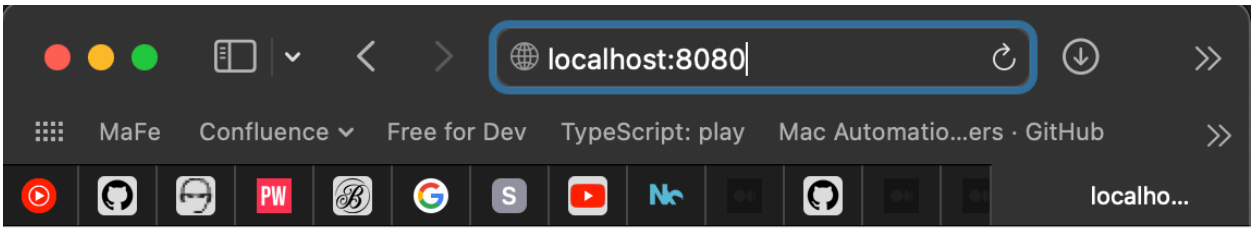
Realizo otro cambio en el mensaje

```
app.get('/', (req, res) => {
  const name = 'From My Computer';
  res.send(`Hello ${name}!`);
});
```

Nodemon vuelve a refrescar la instancia

```
only_docker_compose_container | [nodemon] starting `node index.js`
only_docker_compose_container | helloworld: listening on port 8080
only_docker_compose_container | [nodemon] restarting due to changes...
only_docker_compose_container | [nodemon] starting `node index.js`
only_docker_compose_container | helloworld: listening on port 8080
only_docker_compose_container | [nodemon] restarting due to changes...
only_docker_compose_container | [nodemon] starting `node index.js`
only_docker_compose_container | helloworld: listening on port 8080
```

El servidor responde con este nuevo mensaje



Hello From My Computer!