

## 2 - PERSISTENCIA Y DOCKER COMPOSE

### **Realizar un bind mount para modificar código dinámicamente**

*Instalar un contenedor con la imagen de la Práctica 1, usando un bind mount para ser capaz de cambiar el texto que salga por pantalla por un mensaje diciendo el nombre del alumno.*

*La entrega de este ejercicio debe ser compuesta por las capturas de:*

- *Lanzamiento del docker*
- *Salida de 'docker volume inspect <container>'*
- *Captura de imagen del display de pantalla del servicio web*

Empiezo por construir la imagen de la práctica 1, le he nombrado “imagen\_practica\_01”

```
[ 4:52PM ] [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/actividad-01(main) ]
$ docker build -t imagen_practica_01 .
[+] Building 29.1s (11/11) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 166B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/alpine:3.15 2.5s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [1/5] FROM docker.io/library/alpine:3.15@sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc011 0.0s
=> => resolve docker.io/library/alpine:3.15@sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc011 0.0s
=> => sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285c70fa8c9300 1.64kB / 1.64kB 0.0s
=> => sha256:e7d88de73db3d3fd9b2d63aa7f447a10fd0220b7cbf39803c803f2af9ba256b3 528B / 528B 0.0s
=> => sha256:c059bfaa849c4d8e4aecaeb3a10c2d9b3d85f5165c66ad3a4d937758128c4d18 1.47kB / 1.47kB 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 2.40kB 0.0s
=> [2/5] RUN apk add --update nodejs npm 1.9s
=> [3/5] WORKDIR src/app 0.0s
=> [4/5] COPY ./helloworld/* . 0.0s
=> [5/5] RUN npm install 23.7s
=> exporting to image 0.8s
=> => exporting layers 0.8s
=> => writing image sha256:35687638a35a975398a0bd4c753f89b22915fb874588610eb9f836fe47e5909a 0.0s
=> => naming to docker.io/library/imagen_practica_01 0.0s
[ 4:53PM ] [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/act
```

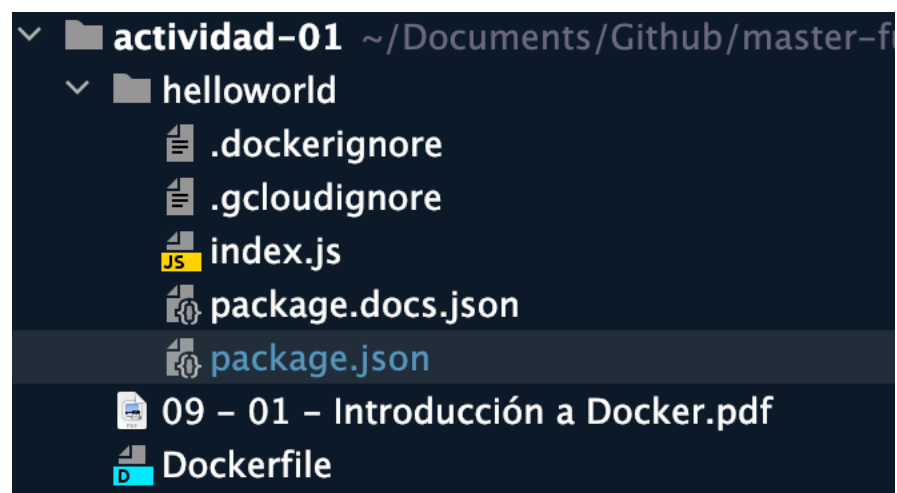
Antes que continuar realizo el siguiente análisis:

- El contenido a cambiar se encuentra en el archivo “index.js” de “Práctica 1”
- Si utilizo **bind mount** a nivel de la **carpeta**, tendré que instalar las dependencias **NPM** en mi máquina ya que no estaré usando las que se encuentran en la imagen (porque viven a nivel de la carpeta), por lo tanto decido utilizar bind mount sobre el **archivo index.js**
- Actualmente, el proyecto corre con node, pero esto impide que se vean cambios en tiempo real si realizo alguna modificación en index.js. En consecuencia, para garantizar el ejercicio, incluyo **nodemon** a mi lista de dependencias en el npm y lo utilizo para iniciar la aplicación en lugar de **node**

Adjunto capturas de las decisiones técnicas anteriormente nombradas en package.json

```
"scripts": {
  "start": "nodemon index.js",
  "test": "mocha test/index.test.js --exit",
  "system-test": "NAME=Cloud mocha test/system.test.js",
},
"engines": {
  "node": ">= 12.0.0"
},
"author": "Google LLC",
"license": "Apache-2.0",
"dependencies": {
  "express": "^4.17.1"
},
"devDependencies": {
  "nodemon": "^2.0.15",
```

Estructura de carpeta por la que usaré bind mount en index.js, como se aprecia, los node\_modules no se consumirán desde mi local



Me posiciono en proyecto “helloworld”

```
[ 5:20PM ] [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/actividad-01(mainx) ]
$ cd helloworld
[ 5:20PM ] [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/actividad-01/helloworld(mainx) ]
$ pwd
/Users/andresmaldonado/Documents/Github/master-fullstack/09-contenedores/actividad-01/helloworld
```

Levanto la imagen configurando como bind mount únicamente el archivo “index.js” que editaré en este ejercicio:

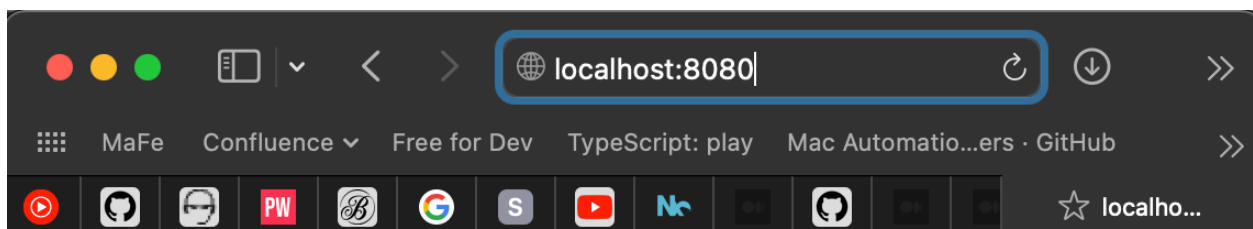
```
[ 5:22PM ] [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/actividad-01/helloworld(mainx) ]
$ docker run -it -p 8080:8080 -v $(pwd)/index.js:/src/app/index.js imagen_practica_01

> helloworld@1.0.0 start
> nodemon index.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
helloworld: listening on port 8080
```

La instancia se encuentra exitosamente corriendo en Docker exponiendo el puerto 8080

```
Last login: Fri Mar 4 16:58:31 on ttys002
[ 5:24PM ] [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/actividad-01/helloworld(mainx) ]
$ docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED        STATUS        PORTS                               NAMES
71319f1f384d   imagen_practica_01   "npm start"             About a minute ago   Up About a minute   0.0.0.0:8080->8080/tcp   great_wilson
```



Hello World!

Cambio el código en index.js con mi nombre como se solicita

```
const express = require('express');
const app = express();

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

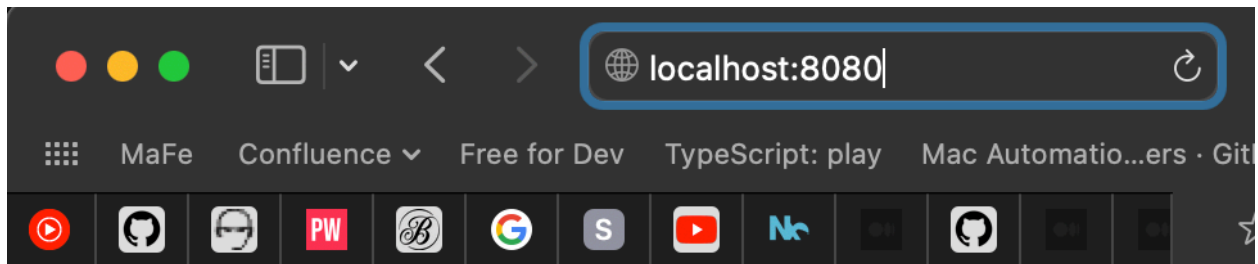
La instancia **NODEMON** que corre actualmente en Docker, escucha el cambio en el archivo y vuelve a lanzar el servidor

```
[ 5:22PM ] [ andresmaldonado@MacBook-Pro-de-Andrés-Maldonado:~/proyectos/ainx ]
$ docker run -it -p 8080:8080 -v $(pwd)/index.js:/index.js

> helloworld@1.0.0 start
> nodemon index.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
helloworld: listening on port 8080
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
helloworld: listening on port 8080
```

Mi localhost:8080 responde esta vez con mi nombre en pantalla, demostrando así que lee el index.js de mi máquina y no el de Docker



Hello Andrés Maldonado!

Finalmente, se solicita un la “Salida de 'docker volume inspect <container>” yo asumo que dado el contexto del ejercicio, se requiere “docker **container** inspect” en lugar de “docker **volume** inspect”

Primero, obtengo el container ID

```
[ 5:31PM ] [ andresmaldonado@MacBook-Pro-de-Andr  
ainx) ]  
$ docker ps  
CONTAINER ID   IMAGE             COMMAND  
71319f1f384d   imagen_practica_01  "npm start"  
[ 5:31PM ] [ andresmaldonado@MacBook-Pro-de-Andr
```

Procedo a ejecutar el comando “docker container inspect 71319f1f384d”

```
71319f1f384d   imagen_practica_01  "npm start"  
[ 5:31PM ] [ andresmaldonado@MacBook-Pro-de-An  
ainx) ]  
$ docker container inspect 71319f1f384d  
[
```

Esto devuelve un enorme JSON

```
71319f1f384d  imagen_practica_01  npm  start  0 minutes ago  Up 0 minutes  0.0.0.0:8080->8080/tcp  great_w
[ 5:31PM ] [ andresmaldonado@MacBook-Pro-de-Andres:~/Documents/Github/master-fullstack/09-contenedores/actividad-01 ]
$ docker container inspect 71319f1f384d
[
  {
    "Id": "71319f1f384dc53ae4d0ed4bec47b8524ed13d2ad74ad57865c7661b15403d70",
    "Created": "2022-03-04T20:22:46.703913642Z",
    "Path": "npm",
    "Args": [
      "start"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 5108,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2022-03-04T20:22:46.980870823Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:4d9fbcc719167ced2b7d90f4dbd53c909f45bf7d4caa1f71afc97bac8345177a",
    "ResolvConfPath": "/var/lib/docker/containers/71319f1f384dc53ae4d0ed4bec47b8524ed13d2ad74ad57865c7661b15403d70/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/71319f1f384dc53ae4d0ed4bec47b8524ed13d2ad74ad57865c7661b15403d70/hostname",
    "HostsPath": "/var/lib/docker/containers/71319f1f384dc53ae4d0ed4bec47b8524ed13d2ad74ad57865c7661b15403d70/hosts",
    "LogPath": "/var/lib/docker/containers/71319f1f384dc53ae4d0ed4bec47b8524ed13d2ad74ad57865c7661b15403d70/localfs-journald.log",
    "Name": "/great_wilson",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": [
        "/Users/andresmaldonado/Documents/Github/master-fullstack/09-contenedores/actividad-01/helloworld:/helloworld"
      ],
      "ContainerIDFile": "",
      "LogConfig": {
        "Type": "json-file",
        "Config": {}
      },
      "NetworkMode": "bridge",
      "PortBindings": {
        "8080/tcp": [
          {
            "HostPort": "8080",
            "ContainerPort": "8080"
          }
        ]
      },
      "RestartPolicy": {
        "Name": "no",
        "Condition": "any"
      },
      "SecurityOpt": [],
      "StorageOpt": null,
      "VolumeDriver": "local",
      "VolumesFrom": null,
      "Workdir": "/helloworld"
    },
    "NetworkSettings": {
      "Bridge": "br-bd36788b",
      "Sandwich": null,
      "HairpinMode": null,
      "LinkLocalIPv6Address": null,
      "LinkLocalIPv6Prefix": null,
      "MacAddress": "02:42:9d:0c:00:00",
      "NetworkInterface": "veth-pair",
      "PortMapping": null,
      "Ports": {
        "8080/tcp": [
          {
            "HostPort": "8080",
            "ContainerPort": "8080",
            "Protocol": "tcp"
          }
        ]
      },
      "SandboxID": "b300000000000000000000000000000000000000000000000000000000000000",
      "SandboxKey": "/dev/null",
      "SecondaryIPAddresses": null,
      "SecondaryPorts": {},
      "StaticPorts": null,
      "Subnet": "172.17.0.1/16",
      "Switch": "veth-pair",
      "ThrottlePriority": null,
      "VethPair": null,
      "VolumeMounts": null,
      "VolatilePorts": {}
    },
    "SystemID": "5d619f2a02159a9030710a1c01f1f5e200000000000000000000000000000000"
  }
]
```

Considerando el contexto del ejercicio, adjunto captura de “**Mounts**” para demostrar que se trabajó con **bind mount** en **index.js**

```
    "Name": "overlay2"
  },
  "Mounts": [
    {
      "Type": "bind",
      "Source": "/Users/andresmaldonado/Documents/Github/master-fullstack/09-contenedores/actividad-01/helloworld/index.js",
      "Destination": "/src/app/index.js",
      "Mode": "",
      "RW": true,
      "Propagation": "rprivate"
    }
  ],
  "Config": {
```