

Universidad de Guadalajara.

Computación Tolerante a fallas.



Diego Ivan Becerra Gonzalez.

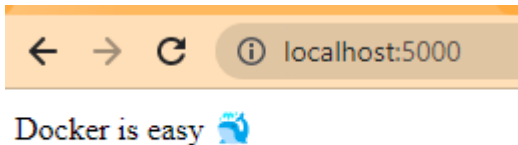
Sección D06.

## Objetivo.

Genera un programa en el cual utilices Docker.

## Desarrollo.

En esta práctica se realizó un programa el cual muestra en pantalla el mensaje de “Docker is easy 🐳”, esto utilizando el paquete express de node js.



Primero se tuvo que instalar el programa de Docker, esto descargándolo desde la página oficial, su instalación es bastante simple. También se tuvo que descargar node js, igualmente desde su página oficial.

Una vez instalados los programas, ejecutamos el comando `npm i express`, para instalar el paquete de express en nuestro pc.

Después ejecutamos `npm init` para que nos cree el archivo `package.json`, el cual tiene la configuración de las dependencias, en este nuevo archivo escribimos la siguiente línea de código:

```
"scripts": {"start": "node index.js"},
```

Esto para indicarle a Docker con que archivo va a iniciar.

Ahora creamos el archivo `index.js`, que es el encargado de mostrar el mensaje en pantalla.

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

app.get('/', (req, res) => {
  res.send('Docker is easy 🌐')
});

const port = process.env.PORT || 8080;

app.listen(port, () => console.log(`app listening on http://localhost:${port}`) );
```

Creamos un archivo Dockerfile, el cual será usado después para crear una imagen la cual pueda ser montada en un contenedor.

```
FROM node:12

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

ENV PORT=8080

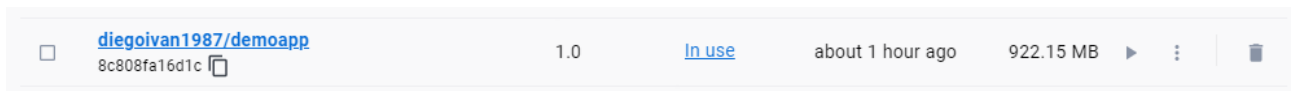
EXPOSE 8080

CMD [ "npm", "start" ]
```






El directorio de trabajo será /app. Copiamos los archivos .json para poder hacer la configuración con node desde docker. Npm install instalara los módulos de node e indicamos que trabajara con el puerto 8080. Al final se ejecuta npm start, que correrá el archivo index.

También creamos un archivo .dockerignore, el cual funciona similar al archivo .gitignore. Ahí escribimos node-modules.

Creamos la imagen usando el comando 'docker build -t diegoivan1987 /dockerdemo:1.0 .' . Es importante poner un punto al final. Esto nos crea lo siguiente en la interfaz de Docker, en el apartado de imágenes:



Para crear el contenedor corremos el siguiente comando `docker run -p 5000:8080 <idDeImagen>`. Esto lo que nos permite es iniciar el contenedor y que lo que se ejecute en su puerto 8080, se ejecute en nuestra máquina, pero en el puerto 5000.

Name	Image	Status	Port(s)	Last started	Actions
 <b>strange_colden</b> 94131047d31e 	<a href="#">8c808fa16d1cecb93b9476bf</a>	Running	<a href="#">5000:8080</a> 	38 minutes ago	 

## Conclusión.

Docker es una herramienta muy útil si de compartir nuestros proyectos se trata, ya que nos permite correr cualquier tipo de programas sin importar las dependencias que tenga, gracias al uso de los contenedores, los cuales tienen todo lo necesario para hacer funcionar un programa. Además, en caso de que un contenedor falle u ocurra algún error, siempre puedes crear otro nuevo contenedor con la misma imagen.

## Bibliografía.

*Docker Basics Tutorial with Node.js.* (2020, 24 agosto).

<https://fireship.io/lessons/docker-basics-tutorial-nodejs/>

Sebastian, N. (2023, 21 enero). *How to fix npm ERR! missing script: start error.*

Nathan Sebastian. <https://sebastian.com/npm-err-missing-script-start/>