

Docker

Ejemplo de uso básico de Docker



Gutiérrez Villalobos Dayal

Código: 216586382

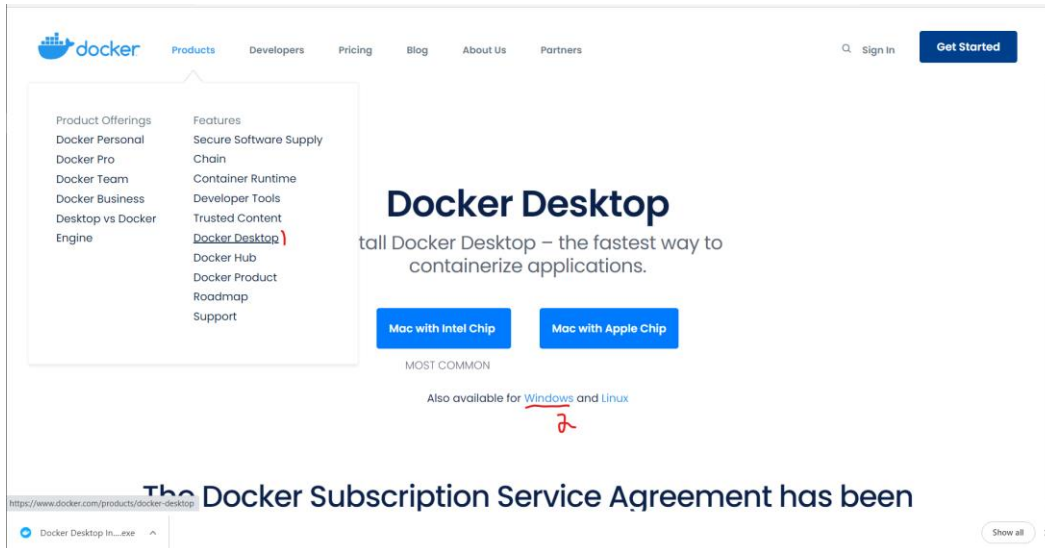
Computación tolerante a fallos

Lunes y miércoles: 11:00-12:55

Sección D06

Maestro: López Franco Michel Emanuel

Lo primero que se tuvo que realizar debido a que no contaba con Docker en mi equipo tuve que ir a la página oficial de Docker, en productos presionar **Docker desktop** y posteriormente descargar los archivos correspondientes para mi sistema operativo.



Una vez instalado Docker habiendo seleccionado todas las opciones para tuviera todos los componentes que este necesita, siendo uno de ellos wsl 2 que ya se tenía previamente instalado en el equipo, se tiene que reiniciar la computadora una vez

Installing Docker Desktop 4.6.0 (75818)

Docker Desktop 4.6.0

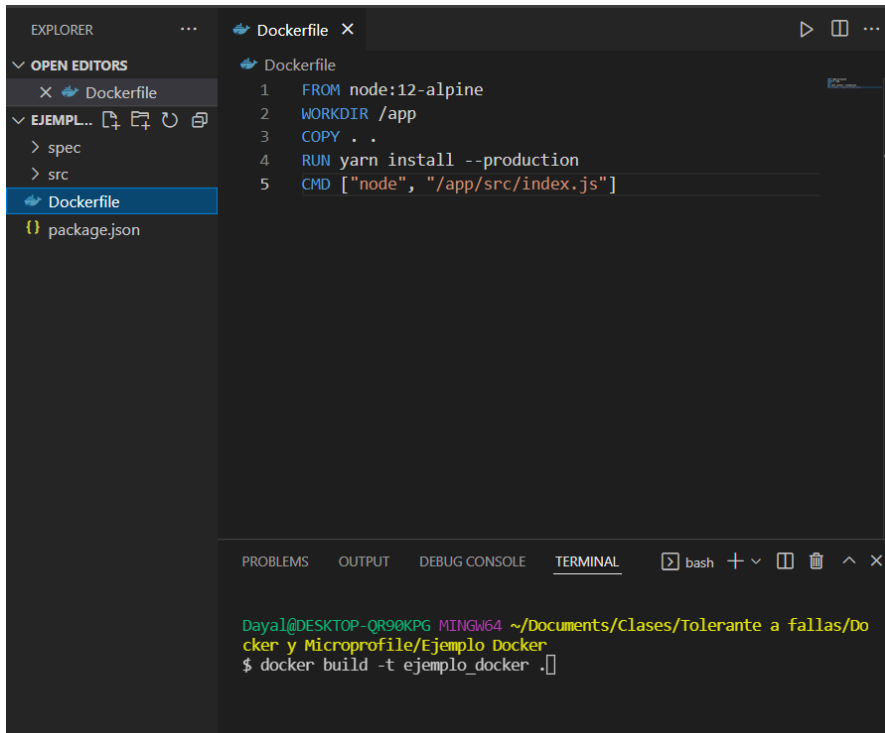
Installation succeeded

You must log out of Windows to complete installation.

Close and log out

que todos los archivos ya se han instalado correctamente

Ya que todo está instalado correctamente nos abre una pestaña en la que podemos seguir un tutorial que el mismo desarrollador nos proporciona para poder entender cómo es que Docker funciona sin embargo el ejemplo que yo seguí es uno realizado por Microsoft.



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with a 'Dockerfile' and a 'package.json' file. The 'Dockerfile' is open in the editor, showing the following content:

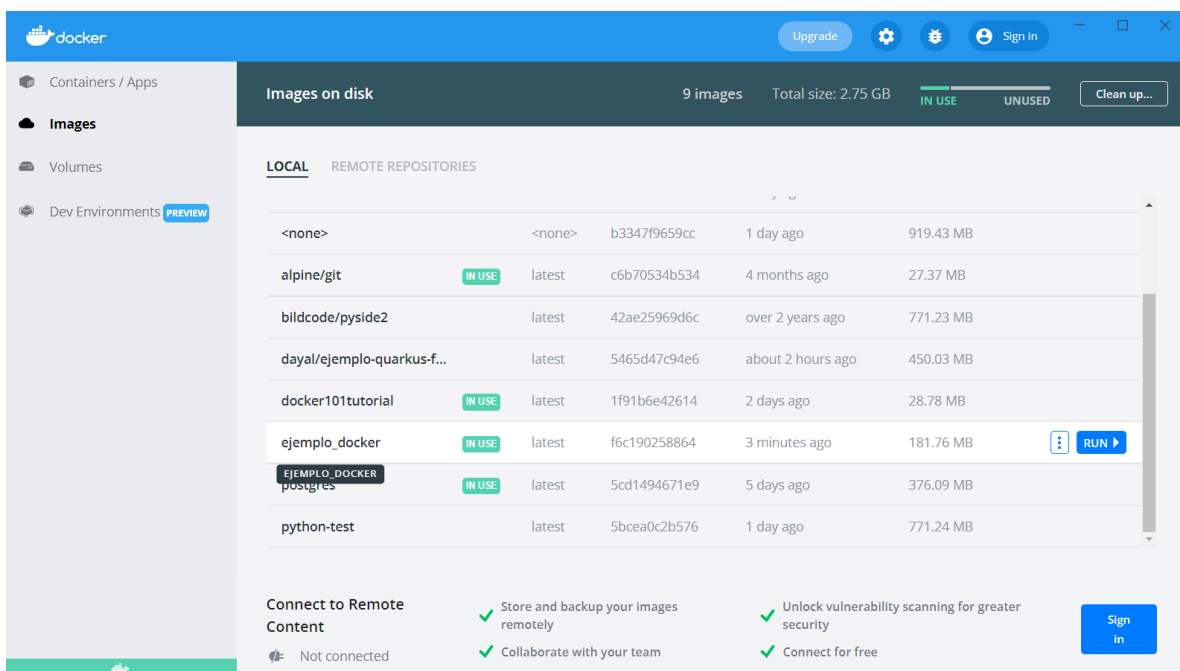
```
1 FROM node:12-alpine
2 WORKDIR /app
3 COPY . .
4 RUN yarn install --production
5 CMD ["node", "/app/src/index.js"]
```

The terminal window shows the command to build the Docker image:

```
Dayal@DESKTOP-QR90KPG MINGW64 ~/Documents/Clases/Tolerante a fallas/Docker y Microprofile/Ejemplo Docker
$ docker build -t ejemplo_docker .
```

Para empezar con el ejemplo una vez que tenemos los archivos de la aplicación ya hechos procedemos a crear un archivo llamado **Dockerfile** el cual cumple con el propósito de poder las instrucciones que se realizaran a Docker para poder crear una imagen de forma exitosa, ya que se tenía el archivo se uso el

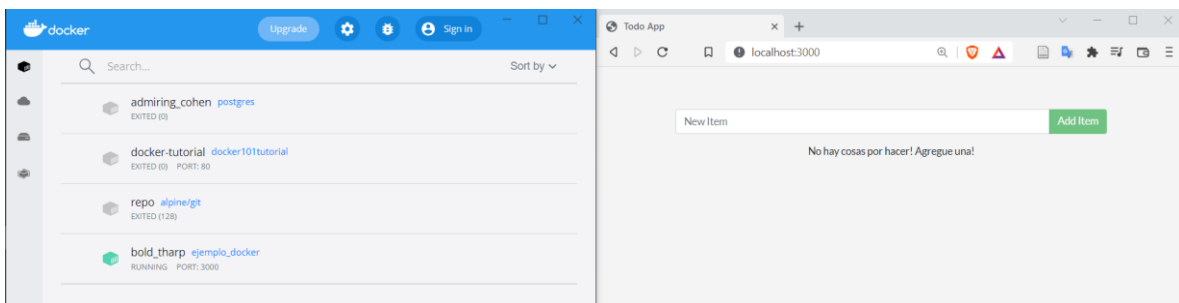
comando que se ve en la terminal para poder crear de una forma exitosa nuestra aplicación



Como se puede observar en el Docker desktop nuestra imagen se ha creado de una manera correcta

```
Dayal@DESKTOP-QR90KPG MINGW64 ~/Documents/Clases/Tolerante a fallas/Docker y Microprofile/Ejemplo Docker
$ docker run -dp 3000:3000 ejemplo_docker
d235f619249ef291836d76cc28e2ee5f85df8cac6681f47ecc1851702034ce58
```

Corriendo este comando podemos crear un contenedor a partir de nuestra imagen, se ha usado la bandera -d para que el proceso sea desasociado, es decir que esté en el fondo y la bandera -p seguido de 3000:3000 para hacer que el puerto 3000 del host que en este caso es mi máquina se vincule con el puerto 3000 del Docker



Y como se puede observar la aplicación funciona de una manera correcta

Debido a que hasta este punto los datos que se ponían en la aplicación solo se guardan de forma local en el contenedor y un contenedor está hecho para que pueda ser prescindible, la forma de almacenar los datos no es la óptima si es que si importan los datos, es por esto que se crea un volumen en el cual los datos serán exportados.

```
Dayal@DESKTOP-QR90KPG MINGW64 ~/Documents/Clases/Tolerante a fallas/Docker y Microprofile/Ejemplo Docker
$ docker volume create lista-db
lista-db
```

Por lo que ahora se debe de correr un contenedor de la siguiente manera

```
Dayal@DESKTOP-QR90KPG MINGW64 ~/Documents/Clases/Tolerante a fallas/Docker y Microprofile/Ejemplo Docker
$ docker run -dp 3000:3000 -v lista-db:/etc/todos ejemplo_docker
```

En donde se especifica en dónde se almacenarán los datos y de donde es que vendrán los datos.

Para poder saber realmente en dónde los datos están almacenados se almacena el siguiente comando en donde podemos observar lo siguiente.

```
Dayal@DESKTOP-QR90KPG MINGW64 ~/Documents/Clases/Tolerante a fallas/Docker y Microprofile/Ejemplo Docker
$ docker volume inspect lista-db
[
  {
    "CreatedAt": "2022-03-22T04:05:15Z",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/lista-db/_data",
    "Name": "lista-db",
    "Options": {},
    "Scope": "local"
  }
]
```

Haciendo que incluso cuando el contenedor se cierra y se crea otro contenedor, pero con el mismo comando con el que creamos este los datos persistan.

Conclusiones

Docker es una herramienta sumamente útil y muy interesante debido a que a pesar de ser tan poderosa en cuanto a la cantidad de cosas que se pueden hacer con la herramienta el comienzo no se me hizo muy difícil de entender. Sin embargo, pienso que la verdadera utilidad está en ambientes de producción en donde se puede observar con mayor amplitud la capacidad que tiene Docker.

Link a repositorio

<https://github.com/DayalGutierrez/Ejemplo-b-sico-de-Docker.git>

Bibliografía

- Microsoft. (2022, 16 marzo). Tutorial: Introducción a aplicaciones de Docker en Visual Studio Code. Microsoft Docs. Recuperado 21 de marzo de 2022, de <https://docs.microsoft.com/es-es/visualstudio/docker/tutorials/docker-tutorial>
- Microsoft. (2022, 16 marzo). Tutorial: Conservación de datos en una aplicación de contenedor mediante volúmenes en VS Code. Microsoft Docs. Recuperado 21 de marzo de 2022, de <https://docs.microsoft.com/es-es/visualstudio/docker/tutorials/tutorial-persist-data-layer-docker-app-with-vscode>