

Pasos:

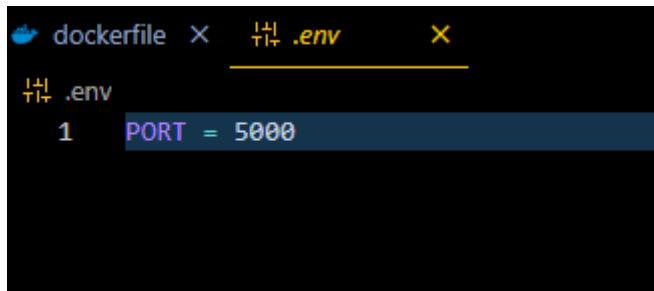
1. Probamos la página instalando sus dependencias en otra carpeta para mantener limpia la imagen (opcional)
2. una vez probado creamos el dockerfile

```
dockerfile X
dockerfile > ...
1  # Indicar que imagen se quiere utilizar
2  FROM node:23.1-slim
3
4  #directorio de trabajo de la aplicacion
5  WORKDIR /app
6
7  #dentro de la imagen se va a copiar los pckg a app
8  COPY package*.json ./
9
10
11  # luego instalar todo los modulos, eliminar el cache
12  RUN npm install \
13      && npm cache clean --force \
14      && rm -rf /tmp/* /root/.npm/_cacache
15  COPY . .
16
17  #indicar el puerto que va a usar
18  EXPOSE 5000
19
20  #LAS IMAGENENES NO SE PUEDEN
21
22  #ejecutar el finchero de la aplicacion
23  CMD ["npm", "start"]
```

3. creamos el Dockerignore

```
dockerfile .dockerignore X
.dockerignore
1  node_modules
2  .git
3  *.log
4  *.zip
5  text
```

4. nos aseguramos que el fichero .env este creado para asegurar el puerto



The screenshot shows a code editor with two tabs: 'dockerfile' and '.env'. The '.env' tab is active, displaying a single line of code: 'PORT = 5000'. The line number '1' is visible on the left margin.

```
1 PORT = 5000
```

5. Cambiamos el despliegue de la aplicación



The screenshot shows a code editor with two tabs: 'dockerfile' and 'package.json'. The 'package.json' tab is active, displaying the following JSON configuration:

```
1 {
2   "name": "exemple_node_docker",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "node .src/index.js",
8     "dev": "node --watch ./src/index.js"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "express": "^4.18.2",
15    "morgan": "^1.10.0"
16  }
17 }
```

6. ejecutamos npm init

```
jazna@MSI MINGW64 ~/Desktop/Curso SOC/FInal
$ npm init
(node:43576) ExperimentalWarning: Support for loading ES Module in require() is an experiment
(Use `node --trace-warnings ...` to show where the warning was created)
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (luis_node_docker)
version: (1.0.0)
license: (ISC)
About to write to C:\Users\jazna\Desktop\Curso SOC\FInal\package.json:

{
  "name": "luis_node_docker",
  "version": "1.0.0",
  "description": "Evaluacion de uso de docker en app node",
  "main": "index.js",
  "scripts": {
    "start": "node src/index.js",
    "dev": "node --watch ./src/index.js"
  },
  "keywords": [],
  "author": "Luis Escobar",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2",
    "morgan": "^1.10.0"
  },
  "repository": {
    "type": "git",
```

7. Creamos la imagen Pruebadocker:v2

```
jazna@MSI MINGW64 ~/Desktop/Curso SOC/FInal
$ docker build -t pruebadocker:v2 .
[+] Building 3.6s (11/11) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 548B
=> [internal] load metadata for docker.io/library/node:23.1-slim
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 78B
=> [1/5] FROM docker.io/library/node:23.1-slim@sha256:952c222213ccf0179745b5180259f104bf4441aec90de4c2f6f38309539b664e
=> => resolve docker.io/library/node:23.1-slim@sha256:952c222213ccf0179745b5180259f104bf4441aec90de4c2f6f38309539b664e
=> [internal] load build context
=> => transferring context: 1.08kB
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY package*.json ./
=> [4/5] RUN npm install && npm cache clean --force && rm -rf /tmp/* /root/.npm/_cacache
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:383a8cef81cab7638fe82bdaa8ba275e807aab8d643ae80f7a0a49fc3edb19a6
=> => exporting config sha256:df8ed3907700364540a3642944886f1df9e93c4fe398e4c30bd529279fe3dd10
=> => exporting attestation manifest sha256:24b26c4c99ad1862411e8f2e8c9cc47e9fc8c0c798c1a6e03a4d56826234baa8
=> => exporting manifest list sha256:18c1cb4b5af23b85d2f644e155dfbf51b5d927b8a1808a8c143af3a812e9fe30
=> => naming to docker.io/library/pruebadocker:v2
=> => unpacking to docker.io/library/pruebadocker:v2
```

Verificamos que la imagen se haya creado en este caso Pruebadocker:v2

```
jazna@MSI MINGW64 ~/Desktop/Curso SOC/FInal
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
pruebadocker	v2	18c1cb4b5af2	7 seconds ago	330MB
pruebadocker	v1	5c926b509612	10 minutes ago	330MB
pruebadocker	v4	294d9a13ff30	23 hours ago	335MB
luizanghel/agencia-viajes	latest	294d9a13ff30	23 hours ago	335MB
pruebadocker	v3	af7327dee4d0	24 hours ago	1.62GB
docker/welcome-to-docker	latest	eedaff45e3c7	12 months ago	29.5MB

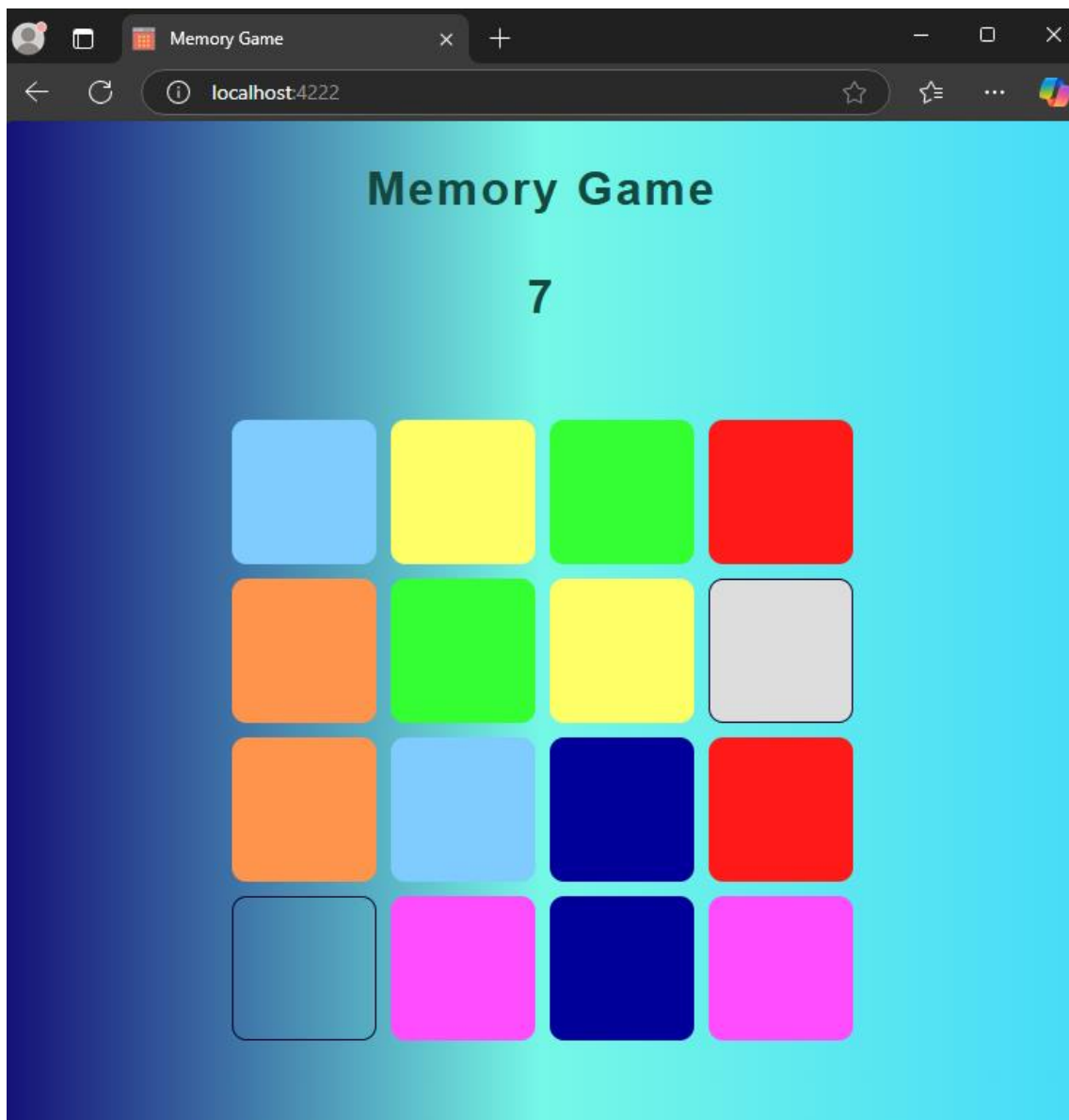
Arrancamos y probamos el servicio puerto 8000:3000

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

jazna@MSI MINGW64 ~/Desktop/Curso SOC/FInal
$ docker run -it -p 8000:3000 pruebadocker:v2

> luis_node_docker@1.0.0 start
> node src/index.js

Servidor web arrancat a http://localhost:3000
█
```

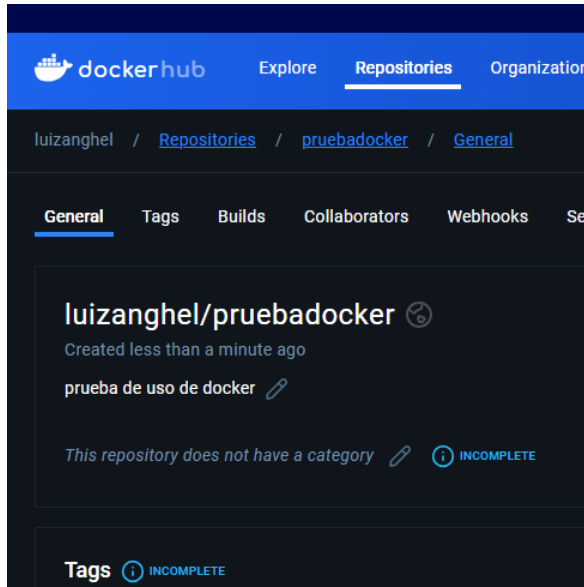


Enlace a git:

<https://github.com/luizanghel/NodeDocker.git>

DOCKERHUB

Entramos a dockerhub y creamos un repositorio



Creamos una versión para la imagen en el repositorio

```
jazna@MSI MINGW64 ~/Desktop/Curso SOC/Final (master)
$ docker tag pruebadocker:v2 luizanghel/pruebadocker:v1
```

Nos autenticamos en Docker

```
jazna@MSI MINGW64 ~/Desktop/Curso SOC/Final (master)
$ docker login -u "luizanghel" -p " "
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
```

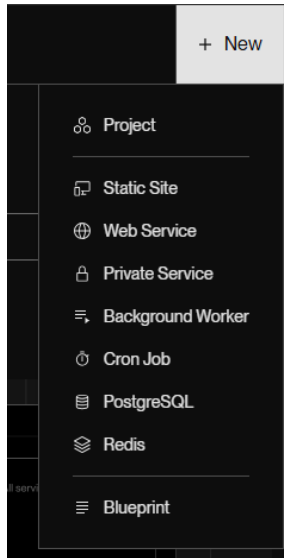
Hacemos un push

```
jazna@MSI MINGW64 ~/Desktop/Curso SOC/Final (master)
$ docker push luizanghel/pruebadocker:v1
The push refers to repository [docker.io/luizanghel/pruebadocker]
8cf2826cb83b: Pushed
76559740dfd0: Pushed
85c5cddd0333: Mounted from luizanghel/agencia-viajes
593492f73912: Mounted from luizanghel/agencia-viajes
740021bec81b: Mounted from luizanghel/agencia-viajes
2d429b9e73a6: Mounted from luizanghel/agencia-viajes
3a45e26a1412: Mounted from luizanghel/agencia-viajes
71dfc14115a4: Pushed
7a3d616bc486: Pushed
d212f687a707: Mounted from luizanghel/agencia-viajes
v1: digest: sha256:18c1cb4b5af23b85d2f644e155dfbf51b5d927b8a1808a8c143af3a812e9fe30 size: 856
```

`docker pull luizanghel/pruebadocker:v1`

RENDER

Entramos a render y creamos un nuevo servicio web



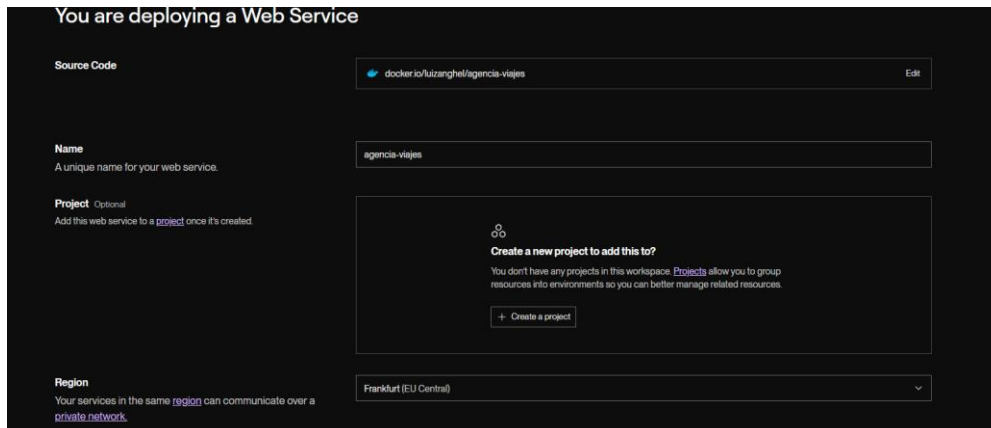
Colocamos la ruta de nuestro repo con su respectiva versión

Docker.io/luizanghel/pruebadocker:v1

connectamos

El nombre del despliegue Luis_escobar-game

Región EU y DEPLOY



Comprobamos que funcione el vinculo

