

RETO #3 DOCKER

1. Revise el archivo PDF adjunto y realice la actividad allí descrita: Adjunte el link al repositorio en GitHub y las evidencias solicitadas.
1. Open your terminal and create a directory for your project: `mkdir node-docker-app` `cd node-docker-app`
Como primer paso lo que se hizo fue crear el directorio del proyecto con el comando `mkdir`

```
PS C:\Users\julia\Downloads> mkdir node-docker-app
```

Directorio: C:\Users\julia\Downloads

Mode	LastWriteTime	Length	Name
d-----	16/09/2025 21:22		node-docker-app

Seguido de esto, nos ubicamos en la carpeta del proyecto

```
PS C:\Users\julia\Downloads> cd node-docker-app
PS C:\Users\julia\Downloads\node-docker-app> |
```

- 2- Como segundo paso corrimos el siguiente comando para generar un archivo `package.json`

`npm init -y`

```
PS C:\Users\julia\Downloads\node-docker-app> npm init -y
Wrote to C:\Users\julia\Downloads\node-docker-app\package.json:

{
  "name": "docker-app",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

- 3- Creamos un archivo llamado `app.js` y agregamos el código dado

```
Set-Content -Path "app.js" -Value @"
const http = require("http");
const port = 3000;
```

```
const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "text/plain" });
  res.end("Hello, Docker!");
});

server.listen(port, () => {
  console.log(` Server running at http://localhost:${port}/`);
});
'@
```

4-Ejecutamos el siguiente comando Npm install express este añade Express al proyecto y actualiza el archivo package.json para incluirlo como dependencia

```
PS C:\Users\julia\Downloads\node-docker-app> npm install express
```

CREACIÓN DE UNA APLICACIÓN NODE.JS CON DOCKER

Recordemos que se siguen 3 pasos básicos:

1. Crear un Dockerfile.
2. Crear la imagen de Docker.
3. Ejecutar el contenedor de Docker.

PASO 1: Creamos un dockerfile

Entonces lo que hacemos es crear un archivo dockerfile en la raíz del proyecto y copiamos las líneas dadas, como se evidencia en la siguiente imagen:

```
# Use alpine node base image
FROM node:18-alpine

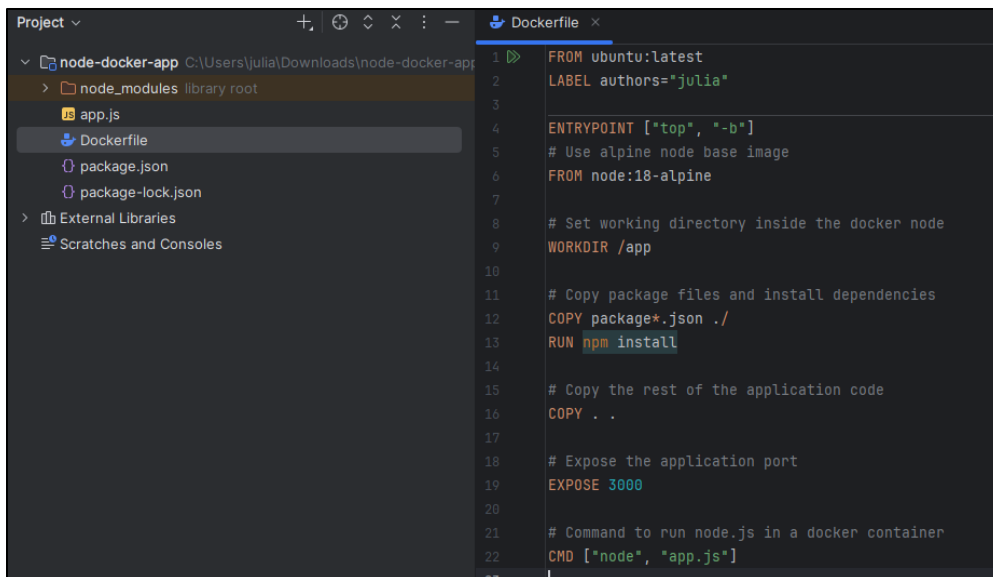
# Set working directory inside the docker node
WORKDIR /app

# Copy package files and install dependencies
COPY package*.json ./
RUN npm install

# Copy the rest of the application code
COPY . .

# Expose the application port
EXPOSE 3000

# Command to run node.js in a docker container
CMD ["node", "app.js"]
```



```
1 FROM ubuntu:latest
2 LABEL authors="julia"
3
4 ENTRYPOINT ["top", "-b"]
5 # Use alpine node base image
6 FROM node:18-alpine
7
8 # Set working directory inside the docker node
9 WORKDIR /app
10
11 # Copy package files and install dependencies
12 COPY package*.json ./
13 RUN npm install
14
15 # Copy the rest of the application code
16 COPY . .
17
18 # Expose the application port
19 EXPOSE 3000
20
21 # Command to run node.js in a docker container
22 CMD ["node", "app.js"]
```

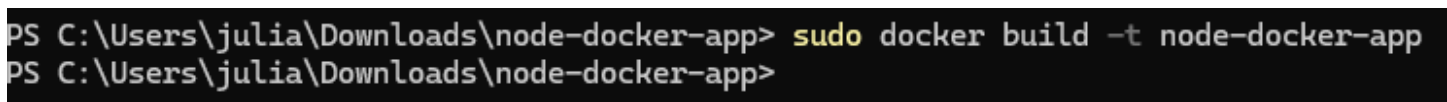
- **Tener en cuenta los siguientes comandos:**

- **FROM:** Especifica la imagen base desde la que se compilará.
- **WORKDIR:** Establece el directorio de trabajo dentro del contenedor.
- **COPY:** Se utiliza para copiar archivos del host al contenedor.
- **RUN:** Ejecuta el comando bash especificado.
- **CMD:** Especifica el comando que se ejecutará al iniciar el contenedor.

PASO 2: CONSTRUIR LA IMAGEN DE DOCKER

Esto lo realizamos abriendo la terminal en el directorio del proyecto y ejecutamos el siguiente comando:

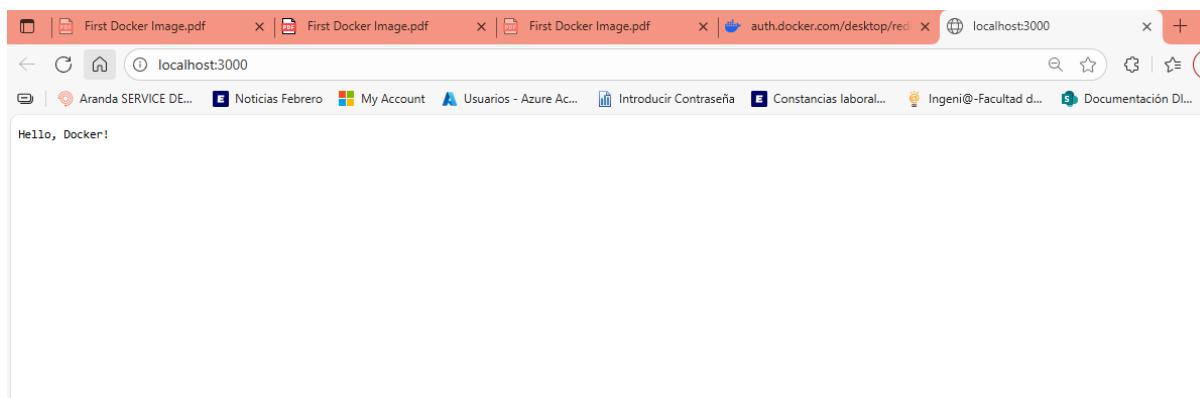
```
sudo docker build -t node-docker-app
```



```
PS C:\Users\julia\Downloads\node-docker-app> sudo docker build -t node-docker-app
PS C:\Users\julia\Downloads\node-docker-app>
```

PASO 3: EJECUTAMOS EL CONTENEDOR

Visitamos **<http://localhost:3000>**



Para comprobar si todo lo anterior esta configurado correctamente, si es el caso veremos “Hola, Docker!”

Como vemos en la imagen anterior la condiguracion se realizo correctamente