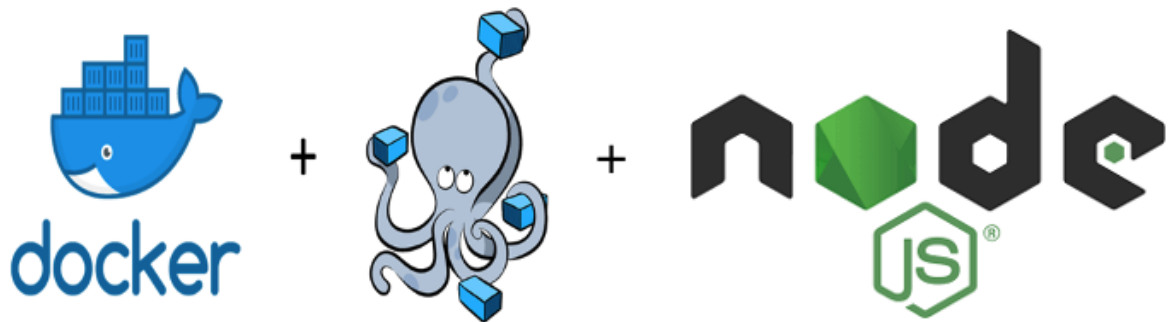


Desplegar una aplicación en Node.js en Docker



Jesús Joel Meneses Meneses
2º DAW A
DPL—Despliegue de Aplicaciones Web

Índice

1. Crear la aplicación Node.js

2. Creando un Dockerfile

3. Construyendo la imagen

4. Ejecutando la imagen

5. Probando la imagen

1. Crear la aplicacion Node.JS

Antes de nada actualizaremos nuestro repositorio de paquetes, con el comando..

```
sudo apt update && apt upgrade
```

```
joel@joel-VirtualBox:~$ sudo apt update && apt upgrade
Obj:1 http://es.archive.ubuntu.com/ubuntu focal InRelease
Obj:2 https://download.docker.com/linux/ubuntu focal InRelease
Des:3 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Des:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Des:7 http://es.archive.ubuntu.com/ubuntu focal-updates/main i386 Packages [563
kB]
Des:8 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1.3
44 kB]
Des:6 https://packages.gitlab.com/gitlab/gitlab-ce/ubuntu focal InRelease [23,3
kB]
Des:9 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [987
kB]
Des:10 http://es.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [27
6 kB]
Des:11 http://es.archive.ubuntu.com/ubuntu focal-updates/main amd64 DEP-11 Metad
ata [279 kB]
Des:12 http://es.archive.ubuntu.com/ubuntu focal-updates/main DEP-11 48x48 Icons
[60,8 kB]
Des:13 http://es.archive.ubuntu.com/ubuntu focal-updates/main DEP-11 64x64 Icons
[98,3 kB]
```

Seguidamente instalaremos un gestor de paquetes con los siguientes comandos

```
curl -fsSL https://deb.nodesource.com/setup\_11.x | sudo -E bash -
```

```
sudo apt-get install -y nodejs
```

```
joel@joel-VirtualBox:~$ curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo
-E bash -

## Installing the NodeSource Node.js 16.x repo...

## Populating apt-get cache...

+ apt-get update
Obj:1 https://download.docker.com/linux/ubuntu focal InRelease
Obj:2 http://es.archive.ubuntu.com/ubuntu focal InRelease
Obj:4 http://es.archive.ubuntu.com/ubuntu focal-updates InRelease
Obj:5 http://es.archive.ubuntu.com/ubuntu focal-backports InRelease
Obj:3 https://packages.gitlab.com/gitlab/gitlab-ce/ubuntu focal InRelease
Obj:6 http://security.ubuntu.com/ubuntu focal-security InRelease
Leyendo lista de paquetes... Hecho

## Confirming "focal" is supported...

+ curl -sLf -o /dev/null 'https://deb.nodesource.com/node_16.x/dists/focal/Relea
se'
```

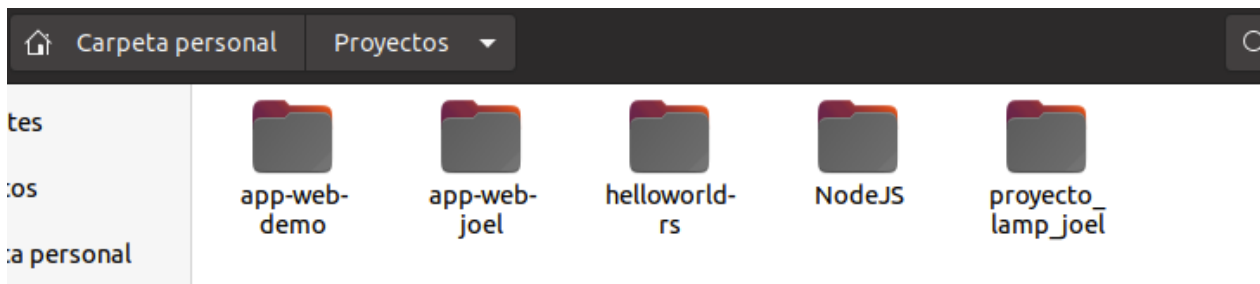
```
joel@joel-VirtualBox:~$ sudo apt-get install -y nodejs
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  gyp javascript-common libc-ares2 libjs-inherits libjs-is-typedarray
  libjs-psl libjs-typedarray-to-buffer libpython2-stdlib libpython2.7-minimal
  libpython2.7-stdlib libraw19 libssl-dev libuv1-dev nodejs-doc
  python-pkg-resources python2 python2-minimal python2.7 python2.7-minimal
  shotwell-common
Utilice «sudo apt autoremove» para eliminarlos.
Los siguientes paquetes se ELIMINARÁN:
  libnode-dev libnode64
Se instalarán los siguientes paquetes NUEVOS:
  nodejs
0 actualizados, 1 nuevos se instalarán, 2 para eliminar y 1 no actualizados.
Se necesita descargar 25,8 MB de archivos.
Se utilizarán 95,5 MB de espacio de disco adicional después de esta operación.
Des:1 https://deb.nodesource.com/node_16.x focal/main amd64 nodejs amd64 16.13.0
-deb-1nodesource1 [25,8 MB]
Descargados 25,8 MB en 2s (11,0 MB/s)
```

Comprobamos que esta instalado la version 8.1.0 con el siguiente comando

```
npm --version
```

```
joel@joel-VirtualBox:~$ npm --version
8.1.0
joel@joel-VirtualBox:~$
```

Primero nos crearemos un nuevo directorio donde estarán todos los archivos



posteriormente abriremos VSCODE, con la carpeta creada y dentro de ella crearemos el archivo `*package.json*` con el contenido siguiente

```
{
  "name": "docker_web_app",

  "version": "1.0.0",

  "description": "Node.js on Docker",

  "author": "First Last first.last@example.com",

  "main": "server.js",

  "scripts": {

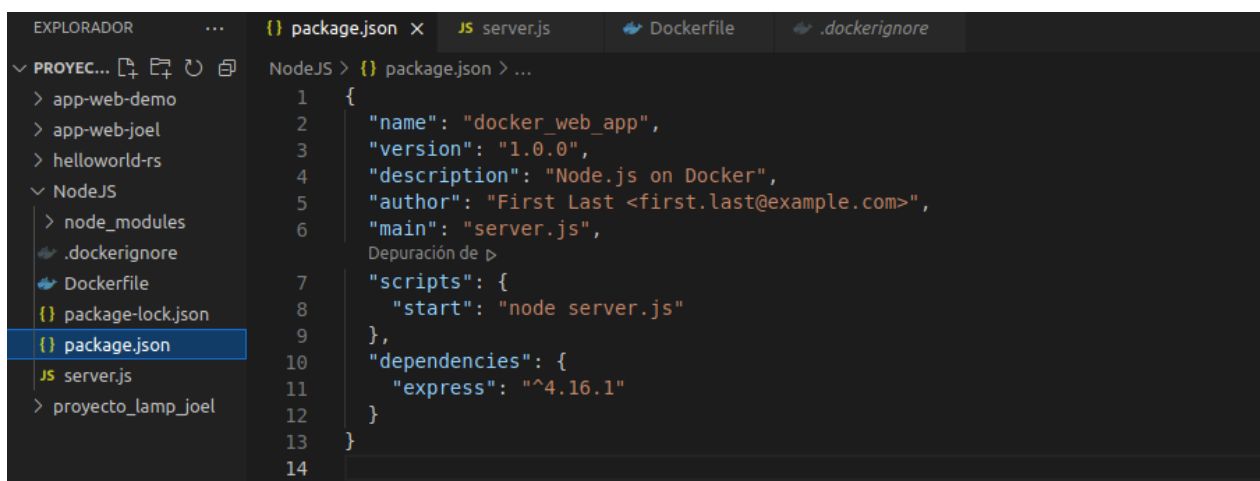
    "start": "node server.js"

  },

  "dependencies": {

    "express": "^4.16.1"

  }
}
```



Ahora ejecutamos el comando siguiente, que nos generará un `package-lock.json` que se copiará a su imagen de Docker.

```
npm install
```

```

joel@joel-VirtualBox:~/Proyectos/NodeJS$ npm install
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was created with an old version of npm,
npm WARN old lockfile so supplemental metadata must be fetched from the registry.
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please be patient...
npm WARN old lockfile

up to date, audited 51 packages in 6s

found 0 vulnerabilities
npm notice
npm notice New patch version of npm available! 8.1.0 -> 8.1.3
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.1.3
npm notice Run npm install -g npm@8.1.3 to update!
npm notice

```

Luego, cree un server.js archivo que defina una aplicación web usando el marco Express.js con el contenido siguiente

```

•
  'use strict';

const express = require('express');

// Constants

const PORT = 8080;

const HOST = '0.0.0.0';

// App

const app = express();

app.get('/', (req, res) => {

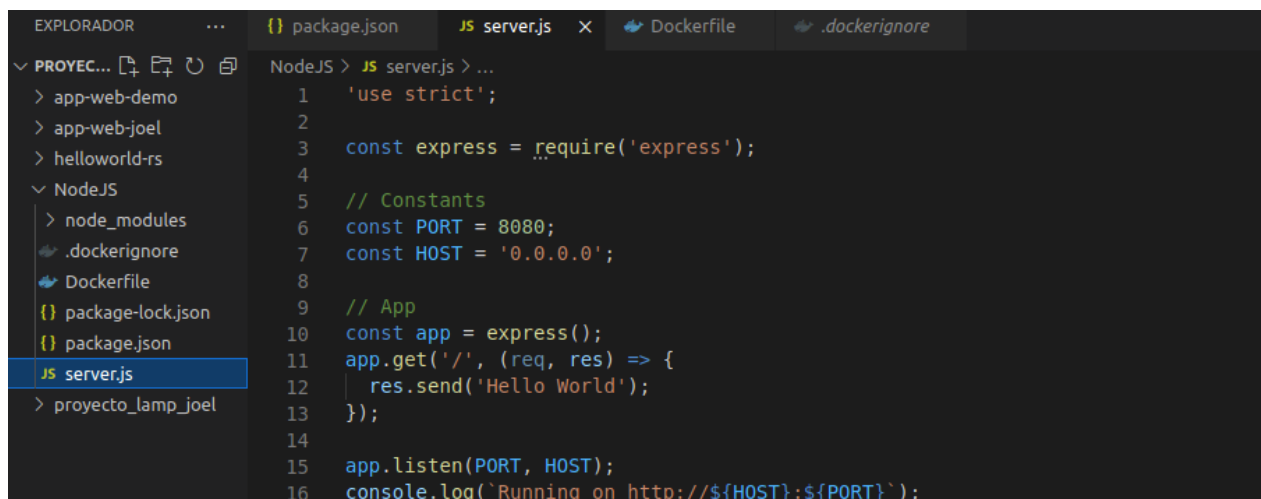
  res.send('Hello World');

});

app.listen(PORT, HOST);

console.log(`Running on http://${HOST}:${PORT}`);

```



The screenshot shows the Visual Studio Code interface. On the left, the 'EXPLORADOR' (Explorer) sidebar is open, showing a project structure with folders like 'app-web-demo', 'app-web-joel', 'helloworld-rs', and 'NodeJS'. The 'NodeJS' folder is expanded, showing files like 'node_modules', '.dockerignore', 'Dockerfile', 'package-lock.json', 'package.json', and 'server.js'. The 'server.js' file is selected and its content is displayed in the main editor area. The code in the editor matches the code block provided in the previous block, showing the setup of an Express.js application.

2. Creando un Dockerfile

Creemos un archivo vacío llamado Dockerfile con el siguiente comando

```
touch Dockerfile
```

Dentro del archivo escribiremos el siguiente contenido

```
NodeJS > Dockerfile > ...
1 FROM node:16
2 # Create app directory
3 WORKDIR /usr/src/app
4 # Install app dependencies
5 # A wildcard is used to ensure both package.json AND package-lock.json are copied
6 # where available (npm@5+)
7 COPY package*.json ./
8
9 RUN npm install
10 # If you are building your code for production
11 # RUN npm ci --only=production
12
13 # Bundle app source
14 COPY . .
15
16 EXPOSE 8080
17 CMD [ "node", "server.js" ]
18
19
20
```

Ahora crearemos un .dockerignore en el mismo directorio que el Dockerfile con el siguiente contenido:

```
NodeJS > .dockerignore
1 node_modules
2 npm-debug.log
```

Esto evitará que sus módulos locales y registros de depuración se copien en la imagen de Docker y posiblemente sobrescriban los módulos instalados dentro de su imagen.

3. Construyendo la imagen

Vamos al directorio que tiene Dockerfile y ejecutamos el siguiente comando para construir la imagen de Docker. La -t bandera permite etiquetar la imagen para que sea más fácil encontrarla más tarde usando el docker images el comando es

```
docker build . -t joel/node-web-app
```

```
joel@joel-VirtualBox:~/Proyectos/NodeJS$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
joel@joel-VirtualBox:~/Proyectos/NodeJS$ sudo docker build . -t joel/node-web-app
Sending build context to Docker daemon 37.38kB
Step 1/7 : FROM node:16
16: Pulling from library/node
07471e81507f: Pull complete
c6ceflaa2170: Pull complete
13a51f13be8e: Pull complete
def39d67a1a7: Pull complete
a8367252e08e: Pull complete
5402d6c1eb0a: Pull complete
f06bf0d834a6: Pull complete
a05e3c4fa294: Pull complete
01b4cbb6b46e: Pull complete
Digest: sha256:683b8ea4ebc033a0f9060501fc31c1481d3f7232cc032851abbd8cc8d91fdff7
Status: Downloaded newer image for node:16
--> 15ddf4b49c29
Step 2/7 : WORKDIR /usr/src/app
--> Running in 33eb7fc2b489
Removing intermediate container 33eb7fc2b489
--> 94c2831c5266
```

Listamos las imagenes de docker

```
joel@joel-VirtualBox:~/Proyectos/NodeJS$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
joel/node-web-app latest 1a4650f63514 50 seconds ago 909MB
node 16 15ddf4b49c29 2 weeks ago 905MB
```

4. Ejecutando la imagen

Crearemos un contenedor con la imagen creada anteriormente con el comando...

```
docker run -p 49160:8080 -d joel/node-web-app
```

```
joel@joel-VirtualBox:~/Proyectos/NodeJS$ sudo docker run -p 49160:8080 -d joel/node-web-app
d74ed93678663f9a26a8f6d88a4a2dc77570a138e831daa5a931307f05d79cd0
```

Obtenemos el ID del contenedor con el comando siguiente

```
sudo docker ps
```

```
joel@joel-VirtualBox:~/Proyectos/NodeJS$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
d74ed9367866 joel/node-web-app "docker-entrypoint.s... 42 seconds ago Up 40 seconds 0.0.0.0:49160->8080/tcp, :::49160->8080/tcp awesome_archimedes
```

Obtenemos la salida de la aplicacion

```
sudo docker logs
```

```
joel@joel-VirtualBox:~/Proyectos/NodeJS$ sudo docker logs d74ed9367866
Running on http://0.0.0.0:8080
```

Ingresamos el contenedor con el comando

```
docker exec -it /bin/bash
```

```
joel@joel-VirtualBox:~/Proyectos/NodeJS$ sudo docker exec -it d74ed9367866 /bin/bash
root@d74ed9367866:/usr/src/app# sudo docker ps
```

5. Probando la imagen

Ahora probamos la aplicación

Ingresamos en el navegador la siguiente dirección de la imagen



Enlace github: [enlace](#)