

Software de Aplicación - VIRTUALIZACIONES

EJERCICIOS DOCKER - Parte 1

1) Realizar los siguientes tutoriales ONLINE Oficiales de DOCKER

- a) <https://training.play-with-docker.com/ops-s1-hello/>

```
[node1] (local) root@192.168.0.13 ~  
$ uname -a  
Linux node1 4.4.0-210-generic #242-Ubuntu SMP Fri Apr 16 09:57:56 UTC 2  
021 x86_64 Linux
```

```
[node1] (local) root@192.168.0.13 ~  
$ docker container run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
719385e32844: Pull complete  
Digest: sha256:fc6cf906cbfa013e80938cdf0bb199fbdbb86d6e3e013783e5a766f5  
0f5dbce0  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correct  
ly.  
  
To generate this message, Docker took the following steps:  
 1. The Docker client contacted the Docker daemon.  
 2. The Docker daemon pulled the "hello-world" image from the Docker Hu  
b.  
    (amd64)  
 3. The Docker daemon created a new container from that image which run
```

```
[node1] (local) root@192.168.0.13 ~  
$ docker image pull alpine  
Using default tag: latest  
latest: Pulling from library/alpine  
8a49fdb3b6a5: Pull complete  
Digest: sha256:02bb6f428431fbc2809c5d1b41eab5a68350194fb508869a33cb1af4  
444c9b11  
Status: Downloaded newer image for alpine:latest  
docker.io/library/alpine:latest
```

```
[node1] (local) root@192.168.0.13 ~
$ docker image ls
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	5e2b554c1c45	2 weeks ago	7.33MB
hello-world	latest	9c7a54a9a43c	3 weeks ago	13.3kB

```
$ docker container run alpine ls -l
```

```
total 8
```

Permissions	Count	User	Group	Size	Time	Path
drwxr-xr-x	2	root	root	4096	May 9 18:39	bin
drwxr-xr-x	5	root	root	340	May 30 18:45	dev
drwxr-xr-x	1	root	root	66	May 30 18:45	etc
drwxr-xr-x	2	root	root	6	May 9 18:39	home
drwxr-xr-x	7	root	root	243	May 9 18:39	lib
drwxr-xr-x	5	root	root	44	May 9 18:39	media
drwxr-xr-x	2	root	root	6	May 9 18:39	mnt
drwxr-xr-x	2	root	root	6	May 9 18:39	opt
dr-xr-xr-x	1240	root	root	0	May 30 18:45	proc
drwx-----	2	root	root	6	May 9 18:39	root
drwxr-xr-x	2	root	root	6	May 9 18:39	run
drwxr-xr-x	2	root	root	4096	May 9 18:39	sbin
drwxr-xr-x	2	root	root	6	May 9 18:39	srv
drwxrwxrwx	13	root	root	0	May 19 17:25	sys
drwxrwxrwt	2	root	root	6	May 9 18:39	tmp
drwxr-xr-x	7	root	root	66	May 9 18:39	usr
drwxr-xr-x	12	root	root	137	May 9 18:39	var

```
[node1] (local) root@192.168.0.13 ~
$ docker container run alpine echo "hello from alpine"
hello from alpine
```

b) <https://training.play-with-docker.com/ops-s1-images/>

```
[node1] (local) root@192.168.0.28 ~
$ docker container run -ti ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
dbf6a9befcde: Pull complete
Digest: sha256:dfd64a3b4296d8c9b62aa3309984f8620b98d87e47492599ee20739e8eb54fbf
Status: Downloaded newer image for ubuntu:latest
root@b5278a4706a3:/#
```

Hemos aplicado los siguientes comandos

apt-get update

apt-get install -y figlet

```
root@b5278a4706a3:/# figlet "hello docker"
```

Una vez actualizado Windows 10 Home a v.2004 (19018) hay que descargar Docker y seguir instrucciones:
<https://docs.docker.com/docker-for-windows/install/>

Hemos de ejecutar el instalador y en cmd aplicar los comandos siguientes

Start-Process 'Docker Desktop Installer.exe' -Wait install

DOCKER está desarrollado PARA LINUX, en UBUNTU: <https://docs.docker.com/engine/install/ubuntu/>
<https://docs.docker.com/engine/install/>

A continuación he utilizado la secuencia de comando que se nos menciona en la web

```
sudo apt-get update
```

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

sudo docker run hello-world, el hello world lo utilizamos para comprobar la instalación

Insalar DOCKER DESKTOP : <https://www.docker.com/products/docker-desktop>

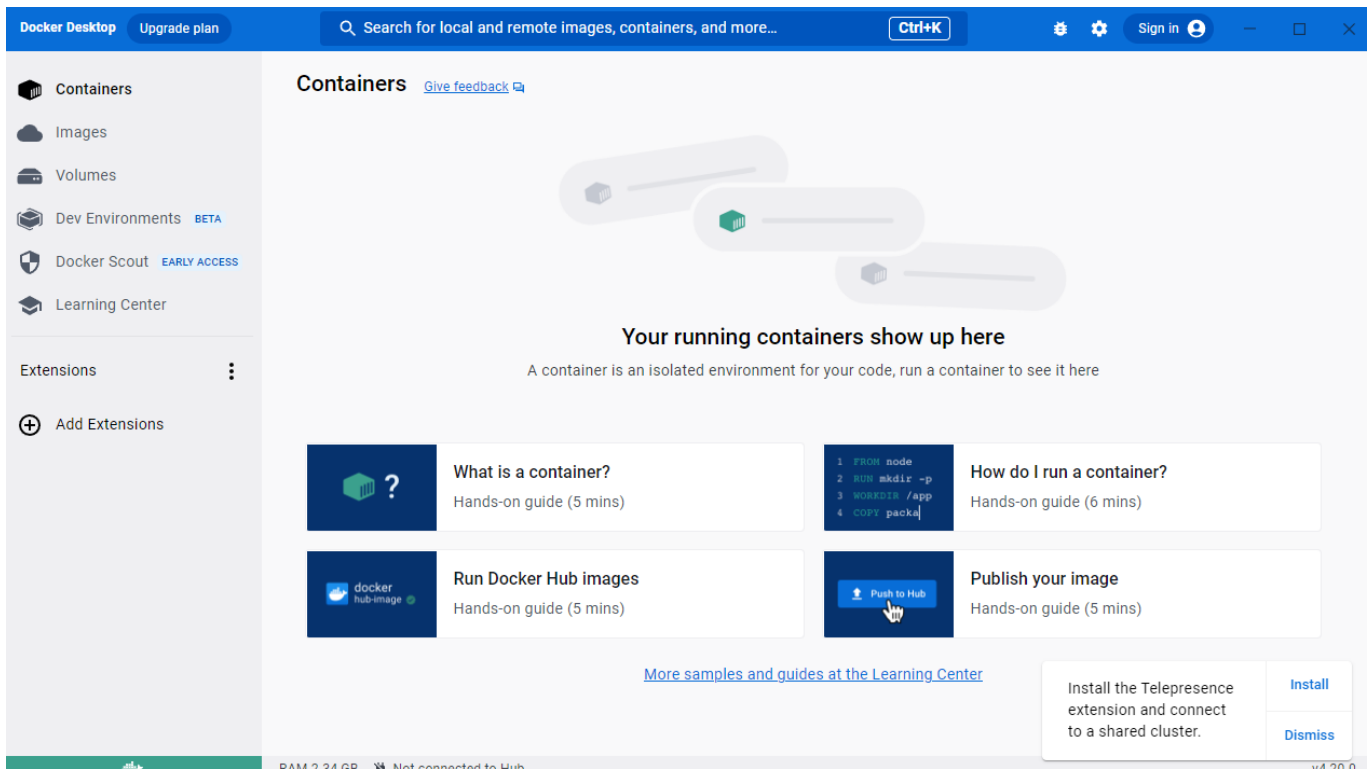
Ya lo he instalado, he instalado en primer lugar el fichero, he descargado el fichero siguiente https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi. A continuación he ido a la terminal del sistema.

```
curl -o %USERPROFILE%\Downloads\wsl_update_x64.msi
```

https://wslstorestorage.blob.core.windows.net/wslblob/wsl_update_x64.msi.

Luego el comando `msiexec.exe /i %USERPROFILE%\Downloads\wsl_update_x64.msi`.

Finalmente he reiniciado el equipo y he abierto docker desktop.



3) Si Docker está instalado, iniciar el servicio, ir a DASHBOARD

Esperar a que el servicio se acabe de inicializar y en PowerShell: abrir COMO ADMINISTRADOR

Tutorial 1: Virtualiza todo una distribución de LINUX (Alpine)

<https://github.com/docker/labs/blob/master/beginner/chapters/alpine.md>

docker pull alpine

He abierto el docker desktop y he ejecutado el comando que se ha mencionado previamente

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1766]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Propietario>docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
8a49fdb3b6a5: Pull complete
Digest: sha256:02bb6f428431fbc2809c5d1b41eab5a68350194fb508869a33cb1af4444c9b11
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

4) Tutorial “Hello world” con Node

<https://youtu.be/FCz10zapsl8>

La idea es crear una app hecha con Node (Javascript) que responde a una llamada desde una página web. Responde con un “Hello world”. Primero creamos la aplicación, luego la “dockerizamos”, creamos un contenedor para poder ejecutar esta simple aplicación en cualquier sistema.

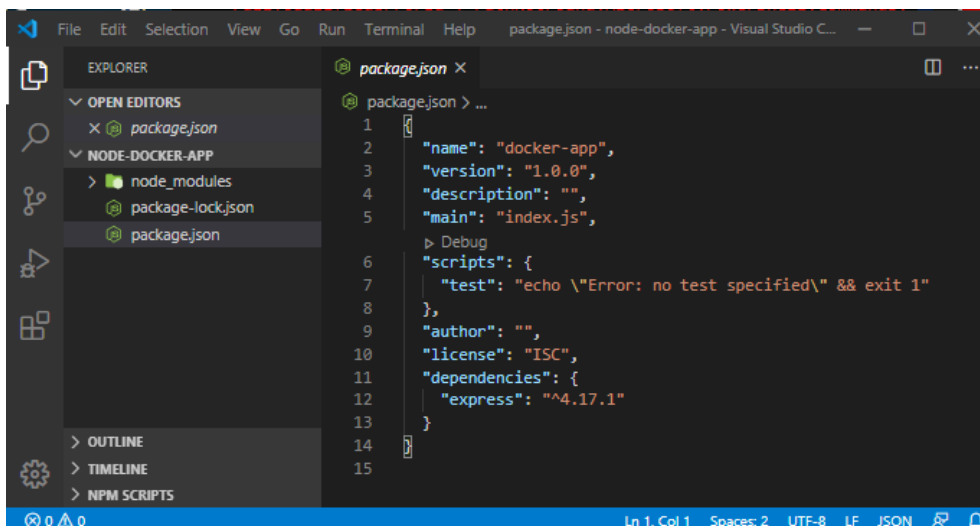
Crear la carpeta de nombre : node-docker-app

dentro de ella , en cmd escribir las instrucciones:

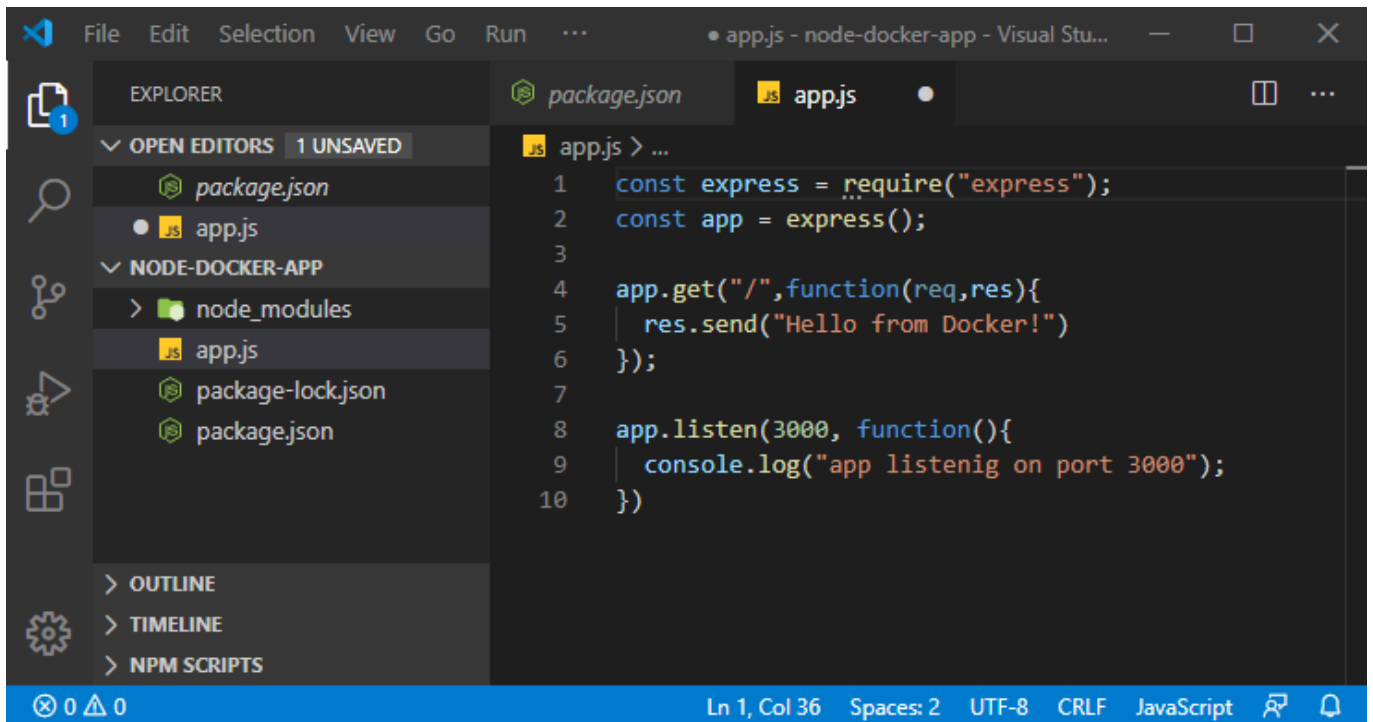
`npm init` (enter a cada opción)

`npm install express`

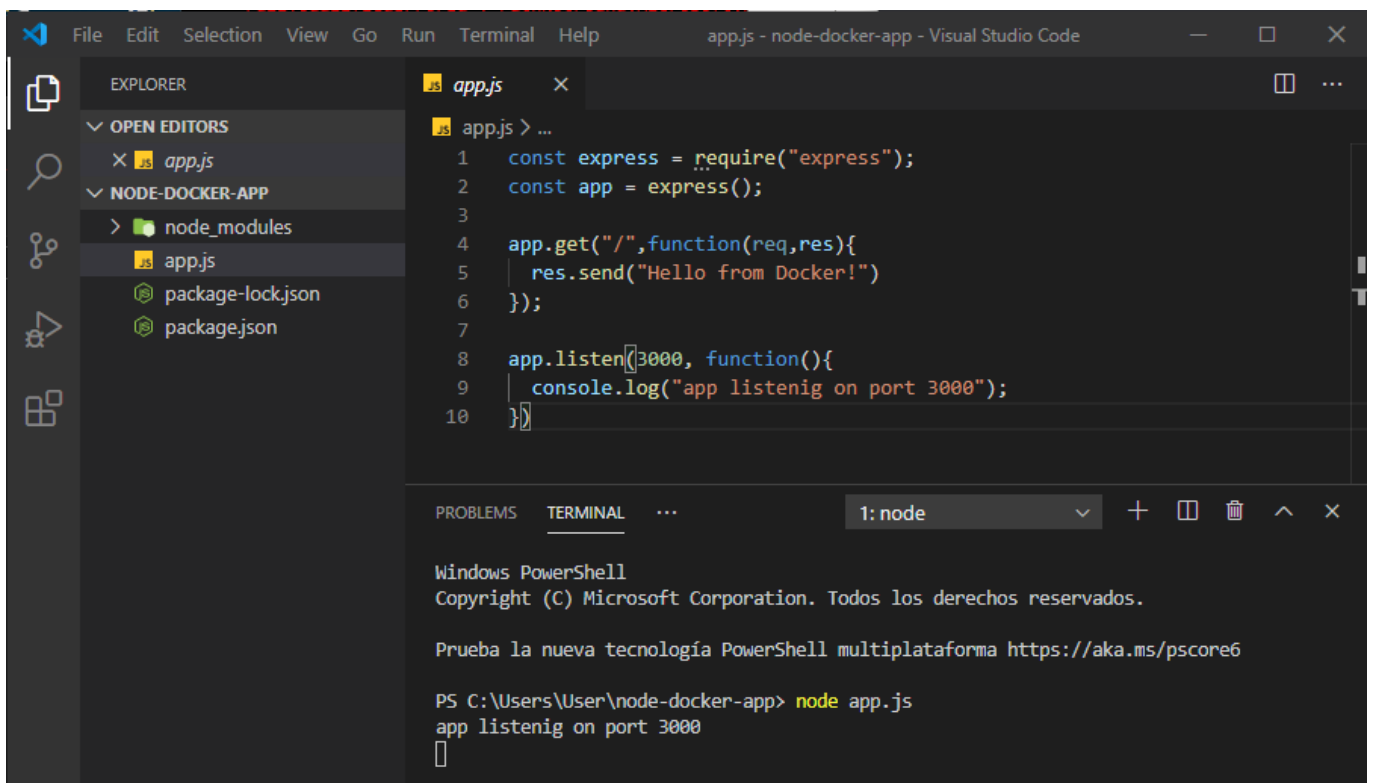
1) Abrir la carpeta desde un editor de texto y ver archivo package.json



2) Crear, dentro de nuestra carpeta ‘node-docker-app’, el siguiente archivo: **app.js**



3) Ver... **Terminal** y escribir instrucción para ejecutar nuestro código: `node app.js`



4) *Con las instrucciones anteriores*, hemos creado una app que responde en la web.

Escribir: **localhost:3000** en nuestro navegador y ver que funciona la app, contesta "Hello from Docker"

```
package name: (docker-app)
version: (1.0.0)




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.

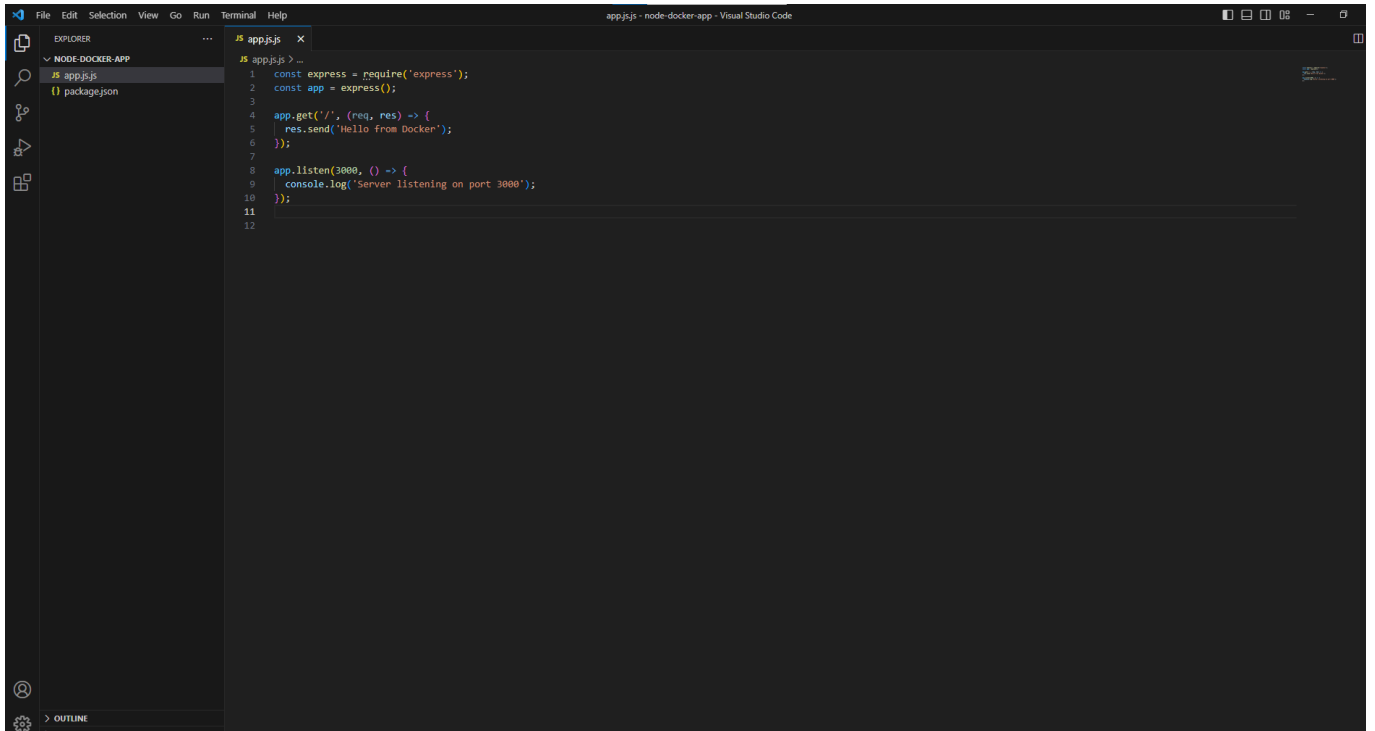
Press ^C at any time to quit.
```

▼ hoy (3)

 wsl_update_x64.msi	31/05/2023 13:49	Paquete de Windo...	16.704 KB
 Docker Desktop Installer.exe	31/05/2023 13:24	Aplicación	603.682 KB
 node-docker-app	31/05/2023 14:09	Carpeta de archivos	

```
package name: (docker-app)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\Propietario\Downloads\node-docker-app\package.json:
save it as a dependency in the package.json file.
{
  "name": "docker-app", quit.
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes)
npm notice
npm notice New minor version of npm available! 9.5.1 -> 9.6.7
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.7
npm notice Run npm install -g npm@9.6.7 to update!
npm notice
```



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a file tree for a project named 'NODE-DOCKER-APP', containing 'app.js' and 'package.json'. The main editor window is open to 'app.js', which contains the following JavaScript code:

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/', (req, res) => {
5   res.send('Hello from Docker');
6 });
7
8 app.listen(3000, () => {
9   console.log('Server listening on port 3000');
10 });
11
12
```

```
C:\Users\Propietario\Downloads\node-docker-app>npm install express
added 58 packages, and audited 59 packages in 3s

8 packages are looking for funding
  run `npm fund` for details

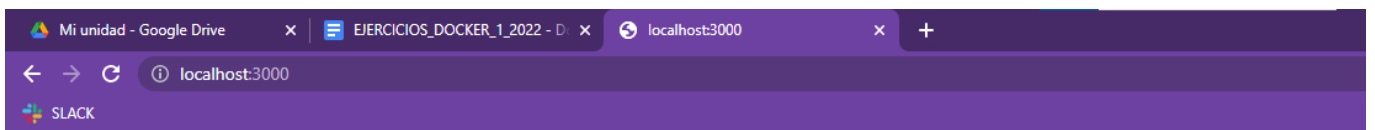
found 0 vulnerabilities

C:\Users\Propietario\Downloads\node-docker-app>
C:\Users\Propietario\Downloads\node-docker-app>npm install express
up to date, audited 59 packages in 2s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Propietario\Downloads\node-docker-app>node app.js
Server listening on port 3000
```



Hello from Docker

5) AHORA es cuando empezamos a utilizar DOCKER. – Servicio DOCKER INICIADO

Creamos contenedor para nuestra app:

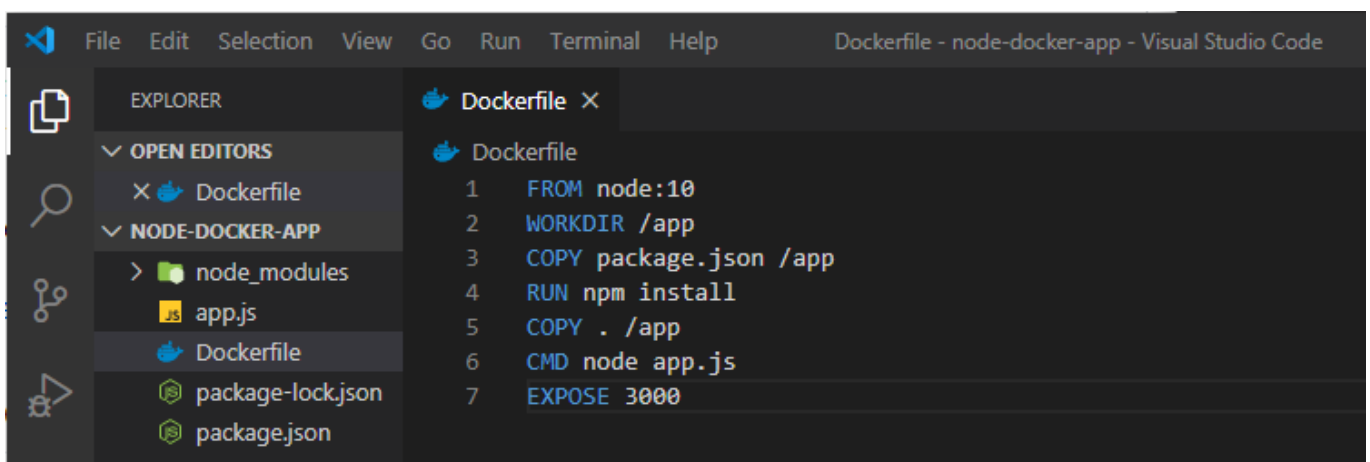
a) Visitar <https://hub.docker.com/> el listado de contenedores que tenemos disponibles ya preparados para contener nuestra app.

Buscamos : node y elegimos la image oficial.

Ejecutar: docker pull node

<https://github.com/nodejs/docker-node/blob/master/README.md#how-to-use-this-image>

b) En el archivo *Dockerfile* , creado anteriormente, escribimos las siguientes instrucciones:



```
File Edit Selection View Go Run Terminal Help Dockerfile - node-docker-app - Visual Studio Code

EXPLORER
OPEN EDITORS
  X Dockerfile
  v NODE-DOCKER-APP
    > node_modules
      app.js
      Dockerfile
      package-lock.json
      package.json

Dockerfile
1 FROM node:10
2 WORKDIR /app
3 COPY package.json /app
4 RUN npm install
5 COPY . /app
6 CMD node app.js
7 EXPOSE 3000
```

FROM node:10 - Desde el contenedor 'node version 10' :

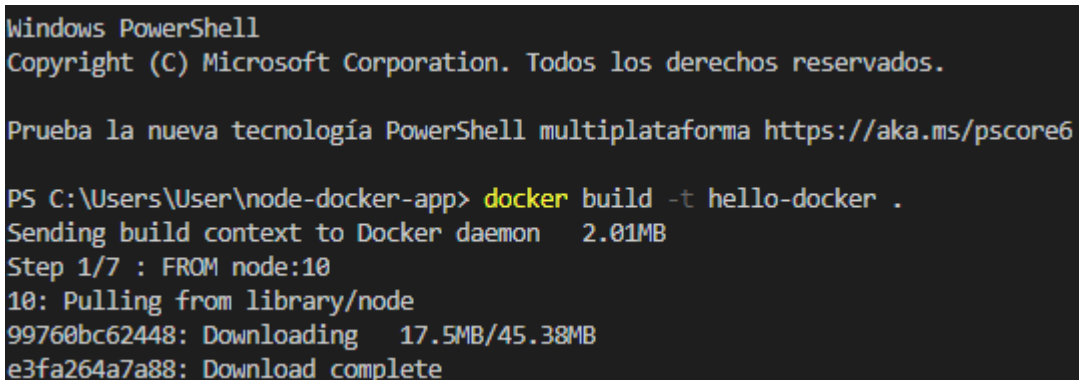
CMD node app.js - ejecuta la aplicación app.js

EXPOSE 3000 - En el port 3000

(no tiene porque ser el mismo que la app, este port es el del contenedor de docker)

c) Desde la ventana de Terminal (o desde CMD), escribir la instrucción: (con punto final)

docker build -t hello-docker . (ojo, no dejarse el punto final!!)



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\User\node-docker-app> docker build -t hello-docker .
Sending build context to Docker daemon 2.01MB
Step 1/7 : FROM node:10
10: Pulling from library/node
99760bc62448: Downloading 17.5MB/45.38MB
e3fa264a7a88: Download complete
```

Va descargando y ejecutando los diferentes pasos que hay en el DOCKERFILE

YA está la imagen de DOCKER creada CON NUESTRA app incorporada.

d) Podemos ver las opciones del comando run, ejecutando en terminal : **docker run --help**

e) Escribimos el comando: **docker run -it -p 3000:3000 hello-docker**

El primer 3000 es del contenedor de Docker, el segundo 3000 del port utilizado en la app

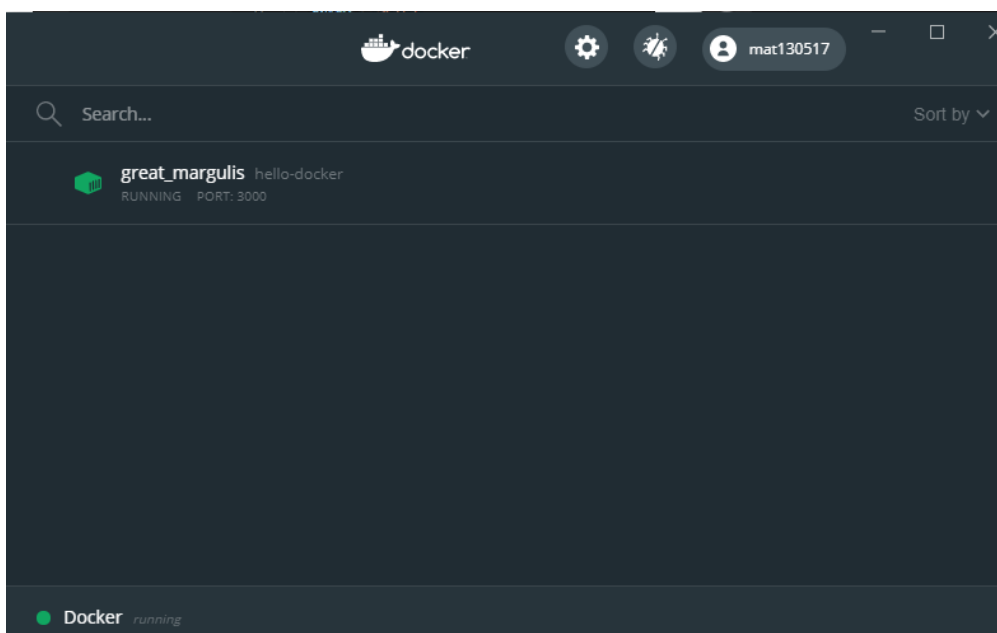
f) YA PODEMOS VER QUE LA APP funciona DESDE el contenedor. Visitando la web: **localhost/3000**

(podríamos cambiar **EXPOSE 3000** por otro número, por ejemplo **EXPOSE 8888**

Habría que hacer un build de nuevo **docker build -t hello-docker** y ejecutar de nuevo **docker run -it -p 8888:3000 hello-docker**

Hemos conseguido un contenedor que podemos publicar en CUALQUIER otro sistema que no sea el nuestro y va a funcionar independientemente del sistema donde se ejecute.

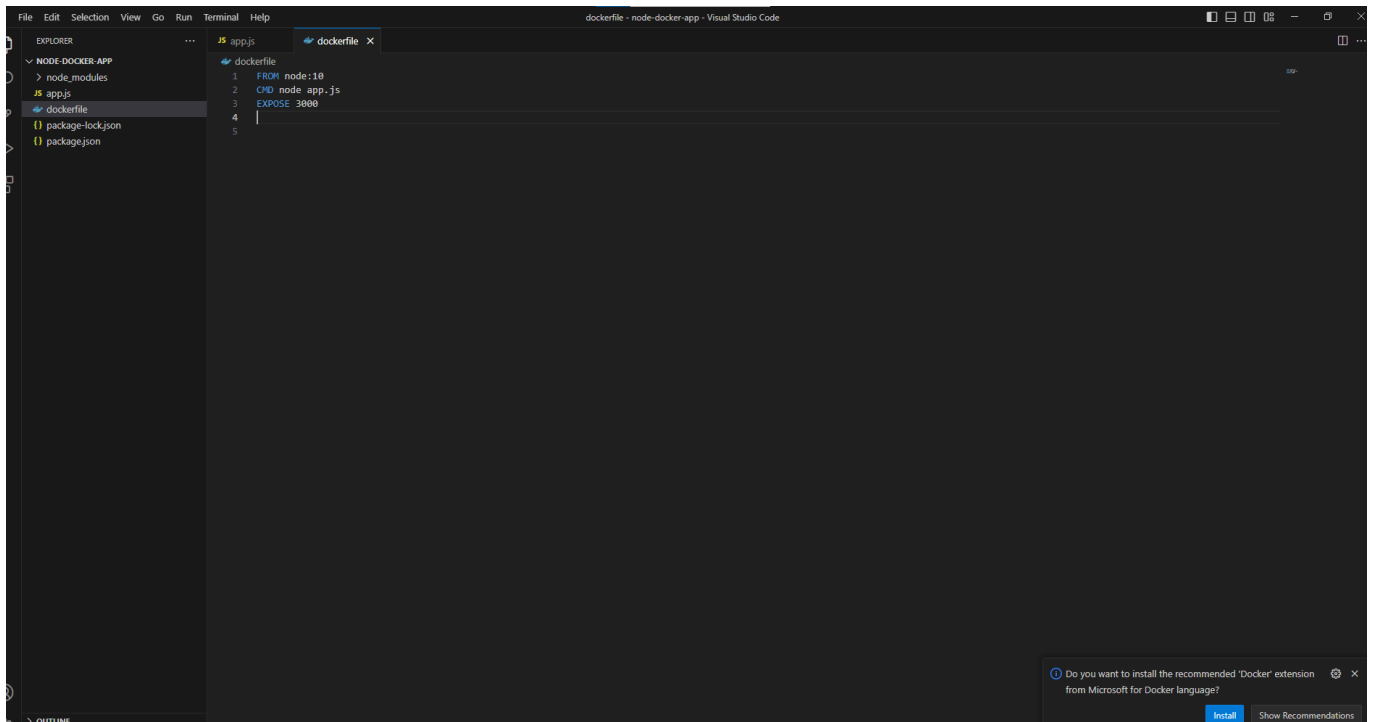
Desde el icono de DOCKER en la barra de inicio, a la derecha: DASHBOARD



a.

```
C:\Users\Propietario>docker pull node
Using default tag: latest
latest: Pulling from library/node
bd73737482dd: Pull complete
6710592d62aa: Pull complete
75256935197e: Downloading [=====>] 53.09MB/54.58MB
c1e5026c6457: Downloading [=====>] 137.6MB/196.9MB
f2e4b4cbd0b8: Download complete
ea5ab4043c9d: Downloading [=====>] 42.39MB/47.55MB
00aabd86b438: Waiting
e3f550576775: Waiting
```

b.



The screenshot shows the Visual Studio Code interface. The Explorer view on the left shows a project structure with folders 'node_modules', 'app.js', 'dockerfile', 'package-lock.json', and 'package.json'. The 'dockerfile' folder is selected. The Dockerfile content is shown in the main editor area:

```
1 FROM node:10
2 CMD node app.js
3 EXPOSE 3000
4
5
```

A notification at the bottom right asks: "Do you want to install the recommended 'Docker' extension from Microsoft for Docker language?" with an "Install" button and a "Show Recommendations" link.

c.

```
C:\Users\Propietario\Downloads\node-docker-app>docker build -t hello-docker C:\Users\Propietario\Downloads\node-docker-app
[+] Building 83.4s (3/4)
=> [internal] load .dockerignore 0.0s
=> [internal] transferring context: 2B 0.0s
=> [internal] load build definition from dockerfile 0.0s
=> [internal] transferring dockerfile: 83B 0.0s
=> [internal] load metadata for docker.io/library/node:10 3.6s
=> [1/1] FROM docker.io/library/node:10@sha256:59531d283edd5101c8f9512f9e095b1836f7a1fc0bab73e005ec46047384911 79.8s
=> resolve docker.io/library/node:10@sha256:59531d283edd5101c8f9512f9e095b1836f7a1fc0bab73e005ec46047384911 0.0s
=> sha256:686e0e85935f28bfe1641e1627540b9c0ead74f222b953d74209213488c6858 2.21kB / 2.21kB 0.0s
=> sha256:28d1a6642d82aaecbed1810109966f001e8de6601e1a1718c2927c124081262 7.83kB / 7.83kB 0.0s
=> sha256:2e2ef8e8a0f40599c10249087268e6a62e433f10e598a09abb5687038a57 11.29MB / 11.29MB 13.0s
=> sha256:59531d283edd5101c8f9512f9e095b1836f7a1fc0bab73e005ec46047384911 776B / 776B 0.0s
=> sha256:76b8ef87096fa726adbe8f073ef09bb5664bac19474c5c4dd69e08a234903b 45.38MB / 45.38MB 27.5s
=> sha256:b53ce1fd2746e8d2037f1b0b91ddea0cc7411eb3e5949fe10c0320aca8f7392b 4.34MB / 4.34MB 0.8s
=> sha256:444c1bd583fcca0be1f286fed9ea13ce12d0a9f6813cf14082ada3e9ab6f63 49.79MB / 49.79MB 58.9s
=> sha256:7a803dc0b40fcd10faee3fb3ebb2d7aaa88500520e6295295f5163c4bb48548b 127.93MB / 214.35MB 79.8s
=> extracting sha256:76b8ef87096fa726adbe8f073ef09bb5664bac19474c5c4dd69e08a234903b 2.4s
=> sha256:b080e0a7e7303e2f608f04911e400fe28f46180b097022c60bf4e02f07b80721 4.19kB / 4.19kB 20.1s
=> sha256:0dad9f7fedd485c74d153hfc7502f34533550a3dd8aa78f508c2476ed500e0073 21.42MB / 21.42MB 41.8s
=> extracting sha256:2e2baf8e8a0f40599c10249087268e6a62e433f10e598a09abb5687038a57 0.4s
=> extracting sha256:b53ce1fd2746e8d2037f1b0b91ddea0cc7411eb3e5949fe10c0320aca8f7392b 0.1s
=> extracting sha256:444c1bd583fcca0be1f286fed9ea13ce12d0a9f6813cf14082ada3e9ab6f63 43.9s
=> sha256:73269809f6fd96184d8d0eafe3cf697b0c8d0e1e7a077f0fab445d74372c78 294B / 294B 44.2s
=> extracting sha256:84a8c1bd5887cca489e1f286fed9ea13ce12d0a9f6813cf14082ada3e9ab6f63 2.6s
```

d.

```
C:\Users\Propietario\Downloads\node-docker-app>docker run --help
Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Create and run a new container from an image

Aliases:
  docker container run, docker run

Options:
  --add-host list          Add a custom host-to-IP mapping
                           (host:ip)
  --annotation map         Add an annotation to the container
                           (passed through to the OCI
                           runtime) (default map[])
  -a, --attach list        Attach to STDIN, STDOUT or STDERR
  --blkio-weight uint16    Block IO (relative weight),
                           between 10 and 1000, or 0 to
                           disable (default 0)
  --blkio-weight-device list
                           Block IO weight (relative device
                           weight) (default [])
  --cap-add list           Add Linux capabilities
  --cap-drop list          Drop Linux capabilities
  --cgroup-parent string   Optional parent cgroup for the
                           container
  --cgroupns string        Cgroup namespace to use
                           (host|private)
                           'host': Run the container in
                           the Docker host's cgroup namespace
                           'private': Run the container in
                           its own private cgroup namespace
                           ':': Use the cgroup
                           namespace as configured by the
                           default-cgroupns-mode
                           option on the daemon (default)
  --cidfile string         Write the container ID to the file
  --cpu-period int         Limit CPU CFS (Completely Fair
                           Scheduler) period
  --cpu-quota int          Limit CPU CFS (Completely Fair
                           Scheduler) quota
  --cpu-rt-period int      Limit CPU real-time period in
                           microseconds
  --cpu-rt-runtime int     Limit CPU real-time runtime in
                           microseconds
  -c, --cpu-shares int      CPU shares (relative weight)
  --cpus decimal           Number of CPUs
  --cpuset-cpus string     CPUs in which to allow execution
                           (0-3, 0,1)
  --cpuset-mems string     MEMS in which to allow execution
                           (0-3, 0,1)
  -d, --detach             Run container in background and
                           print container ID
  --detach-keys string     Override the key sequence for
                           detaching a container
  --device list            Add a host device to the container
  --device-cgroup-rule list
                           Add a rule to the cgroup allowed
                           devices list
  --device-read-bps list   Limit read rate (bytes per second)
                           from a device (default [])
  --device-read-ops list   Limit read rate (IO per second)
```

e.

Da error, por lo tanto no puedo continuar

4) Videos explicativos:

¿Qué es Docker? - Webinar Docker: <https://youtu.be/bOAicRfdZHs>

En este video se puede observar que nos presentan los conceptos básicos de contenedores y Docker, incluida la forma en que ofrecen una alternativa liviana a las máquinas virtuales y los beneficios de usarlos para entornos de desarrollo y producción. El orador describe los componentes de las imágenes y los contenedores de Docker, como Dockerfile y el registro de Docker, y también nos demuestra cómo administrar y asignar recursos a los contenedores. También exploran cómo crear y ejecutar contenedores usando imágenes específicas y cómo conservar datos usando volúmenes de datos. Finalmente, el video cubre las mejores prácticas para trabajar con Docker, incluido el uso de imágenes oficiales, y presenta el concepto de redes en contenedores Docker.

El video nos brinda una introducción a Docker y los contenedores y explica cómo ejecutar contenedores con Docker, conectarlos en una red y ejecutarlos en producción usando Docker Compose y Kubernetes. El orador analiza los factores que afectan la ejecución de una aplicación en contenedores en producción, como la resiliencia, la alta disponibilidad, la escalabilidad, la persistencia de datos, la recuperación ante desastres, la gestión de costos y la integración. También comparan Kubernetes con otras plataformas de contenedores y analizan el uso de Jenkins u otro software de integración continua con Docker. El video incluye una sesión de preguntas y respuestas.

5) TUTORIAL : puede hacerse online, o desde la instalación de Docker en nuestro PC:

Realizar el siguiente TUTORIAL: <https://www.docker.com/101-tutorial>

Puede hacerse en tu propio ordenador, si has instalado DOCKER o en la nube si NO lo has instalado.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1766]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Propietario>docker run -dp 80:80 docker/getting-started
docker: error during connect: this error may indicate that the docker daemon is not running: Post "http://%2F%2F.%2Fpipe/docker_engine/v1.24/containers/create": open //./pipe/docker_engine: El sistema no puede encontrar el archivo especificado.
See 'docker run --help'.

C:\Users\Propietario>docker run -dp 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
c158987b0551: Pull complete
1e35f6679fab: Pull complete
cb9626c74200: Pull complete
b6334b6ace34: Pull complete
f1d1c9928c82: Pull complete
9b6f639ec6ea: Pull complete
ee68d3549ec8: Pull complete
B3e0cbbb4673: Pull complete
4f7e34c2de10: Pull complete
Digest: sha256:d79336f4812b6547a53e735480dde67f8f7071b414fbd9297609ffb989abc1
Status: Downloaded newer image for docker/getting-started:latest
19718dea3f7f92e0846e85a041d9e40ec224e2a1154a0660f594002de8231ff2

C:\Users\Propietario>
```

