

Tarea 1 Docker

INDICE

Dockeritzación del despliegue de una aplicación en Node.js.....	2
Despliegue con Docker.....	2
Tarea 1.....	2
BUILD.....	2
Conectarse al contenedor.....	3

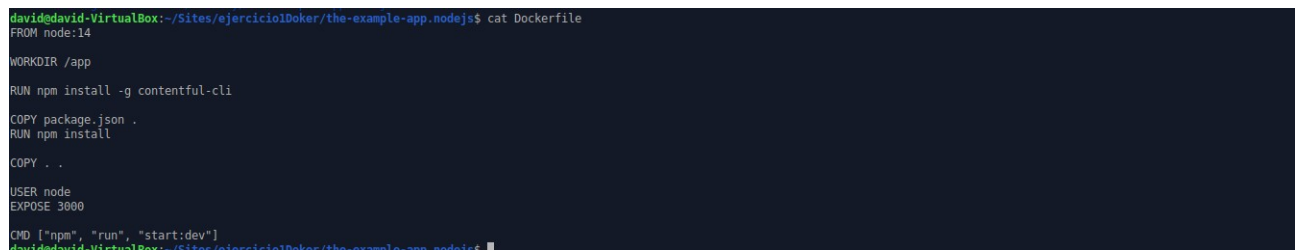
Dockerización del despliegue de una aplicación en Node.js

Despliegue con Docker

Vamos a clonar primero el repositorio the-example-app si todavía no lo tenemos este es el repositorio que tenemos que clonar: (git clone <https://github.com/contentful/the-example-app.nodejs.git>).

Tarea 1

Vamos dentro de la carpeta the-example-app y dentro creamos un Dockerfile con este formato.



```
david@david-VirtualBox:~/Sites/ejercicioDocker/the-example-app.nodejs$ cat Dockerfile
FROM node:14
WORKDIR /app
RUN npm install -g contentful-cli
COPY package.json .
RUN npm install
COPY . .
USER node
EXPOSE 3000
CMD ["npm", "run", "start:dev"]
```

FROM node :14 (indicamos a docker que version de node queremos)

WORKDIR /app (en que directorio se ejecutara las instrucciones de Dockerfile)

RUN npm install -g contentful-cli (Instalación de nueva imagen)

COPY package.json . (donde copiamos los archivos del contenedor)

RUN npm isntall(las dependencias que tiene que instalar para que funcione node)

COPY . . (se copia en el mismo directorio del contenedor)

USER node (Se le dice al usuario que tipo de contenedor va usar)

EXPOSE 3000(el puerto que se va a escuchar)

CMD [“npm”,”run”,”start:dev”](los comandos que nos permite ejecutar dentro del contenedor).

BUILD

Ahora vamos a hacer un build(montar) a nuestra imagen de docker. El comando seria docker build -t imagen.

```
david@david-VirtualBox:~/Sites/ejercicio1Docker/the-example-app.nodejs$ sudo docker build -t the-example-app.nodejs .
[+] Building 50.7s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 207B
=> [internal] load metadata for docker.io/library/node:latest
=> [internal] load .dockerignore
=> => transferring context: 67B
=> [stage-1 1/1] FROM docker.io/library/node:latest@sha256:3af9f785cb8fcla9c60a77c7b31b1ba7f5c74a066d142c996fbce61d2420dd8c
=> resolve docker.io/library/node:latest@sha256:3af9f785cb8fcla9c60a77c7b31b1ba7f5c74a066d142c996fbce61d2420dd8c
=> sha256:2cc34c90ae08e9a7447d01513af15a1f37193b04bc3a84a2c13f7f3065665b3a0 2.00kB / 2.00kB
=> sha256:6a299ae9cf996c1149a699d36cdaa76fa332c8e9d66d6678fa9a231d9ead04c 49.58MB / 49.58MB
=> sha256:3af9f785cb8fcla9c60a77c7b31b1ba7f5c74a066d142c996fbce61d2420dd8c 1.21kB / 1.21kB
=> sha256:8d6c41e6504f6b9d7a8c2a82803a4d158660ae24056ad20bef092ef21d646253 7.34kB / 7.34kB
=> sha256:e08e8703b2fb5e50153f792f3192087d26970d262806b397049d61b9a14b3af5 24.05MB / 24.05MB
=> sha256:68e92d11b04ec0fe48e60d59964704aca234084f87af5d1a068c49456b37fe3d 64.14MB / 64.14MB
=> sha256:5b9fe7fe9befda786bc8e1dd1ae42ffd8b9c37a4cce3c433e65ebb898cb1672 211.11MB / 211.11MB
=> sha256:278d467c182fb935287ed6f4be3d44f8ac2714264bcb58b883d0cd25754f75d6 3.37kB / 3.37kB
=> sha256:aab6430b55a2372670b8797fbedeae0e36526282a1bf55ed8489ed5617045574 49.30MB / 49.30MB
=> sha256:b6f92f4f624059d4cf4ea96bc1f2512f74d63ebabccb16963fa50ad7cbec938e 2.23MB / 2.23MB
=> sha256:2e24338062e4bb2910914b5ba45c6bad6985b6f31a6b0dc95f748a5d67f3001 451B / 451B
=> extracting sha256:6a299ae9cf996c1149a699d36cdaa76fa332c8e9d66d6678fa9a231d9ead04c
=> extracting sha256:e08e8703b2fb5e50153f792f3192087d26970d262806b397049d61b9a14b3af5
=> extracting sha256:68e92d11b04ec0fe48e60d59964704aca234084f87af5d1a068c49456b37fe3d
=> extracting sha256:5b9fe7fe9befda786bc8e1dd1ae42ffd8b9c37a4cce3c433e65ebb898cb1672
=> extracting sha256:278d467c182fb935287ed6f4be3d44f8ac2714264bcb58b883d0cd25754f75d6
=> extracting sha256:aab6430b55a2372670b8797fbedeae0e36526282a1bf55ed8489ed5617045574
=> extracting sha256:b6f92f4f624059d4cf4ea96bc1f2512f74d63ebabccb16963fa50ad7cbec938e
=> extracting sha256:2e24338062e4bb2910914b5ba45c6bad6985b6f31a6b0dc95f748a5d67f3001
=> exporting to image
=> exporting layers
=> writing image sha256:f56064d04613a06094877061d97e9f519294579f390aaab94e73a8942b549ca8
=> naming to docker.io/library/the-example-app.nodejs
```

Conectarse al contenedor

Vamos a iniciar nuestro contenedor de la aplicación para ello utilizamos el comando `docker run -p (puerto que escucha nuestro pc):(puerto que escucha el contenedor) -d(dominio) imagen`.

```
david@david-VirtualBox:~/Sites/ejercicio1Docker/the-example-app.nodejs$ docker run -p 3000:3000 -d the-example-app.nodejs
e30e8c3acf9d4c4e35a230e12bb4fbc2a2734261705036e8be4f8b8366a69cad
david@david-VirtualBox:~/Sites/ejercicio1Docker/the-example-app.nodejs$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e30e8c3acf9d	the-example-app.nodejs	"docker-entrypoint.s..."	3 seconds ago	Up 2 seconds	0.0.0.0:3000->3000/tcp, :::3000->3000/tcp	interesting chatterjee
55abbf027eb0	wordpress:4.9.8	"docker-entrypoint.s..."	3 days ago	Up 25 hours	0.0.0.0:8080->80/tcp, :::8080->80/tcp	wordpress_web_1
f37150067064	mariadb:10.3.9	"docker-entrypoint.s..."	3 days ago	Up 25 hours	3306/tcp	wordpress_db_1

Al hacer `docker ps` vemos que el contenedor esta con el status en up (como encendido o operativo) si queremos entrar en el como le hemos indicado el puerto 3000:3000 con poner la IP de nuestra maquina virtual:3000 deberíamos poder entrar.

! Your Preview API key is invalid.