

Construcción de imágenes Docker y despliegue con Docker Compose (laboratorio)

ACTIVIDAD #2 LAB – CONTENEDORES - UNIR
OSCAR QUESADA AVALOS

Contenido

Instrucciones.....	2
Objetivos	2
Pautas de Elaboración	2
Extensión y formato.....	3
Rubrica.....	3
Criterio 1	4
Criterio 2	5
Criterio 3	7
Criterio 4	8
Criterio 5	10
Criterio 6	11
Errores	13
Referencias.....	13
Guia Node + Mongo:	13
Repo 1.....	13

Instrucciones

Objetivos

Desarrollar los conocimientos obtenidos a través de los temas dedicados a Docker en la asignatura de Contenedores.

Pautas de Elaboración

Node.js es un entorno de ejecución de Javascript diseñado para crear aplicaciones escalables, el cual sigue un modelo asíncrono y dirigido por eventos. Por otro lado, los ficheros Dockerfile nos proporcionan una manera declarativa y consistente de construir imágenes de Docker para nuestras aplicaciones.

Esta actividad consiste en «contenerizar» una sencilla aplicación existente de Node.js en una imagen de Docker mediante un fichero Dockerfile. Una vez generada la imagen, desplegaremos la aplicación en contenedores a partir de un fichero de Docker Compose. Además, nuestra aplicación de ejemplo requiere de una base de datos MongoDB, por lo que deberemos desplegarla también en un contenedor.

Algunas consideraciones:

- ▶ Al generar la imagen con Dockerfile deberemos instalar las dependencias de la aplicación necesarias con el gestor de paquetes npm, y exponer el puerto 3000. Además, se deberán seguir las buenas prácticas recomendadas.
- ▶ En el fichero de Docker Compose se deberán definir el servicio de la aplicación Node.js y el de la base de datos MongoDB. La conexión a la base de datos en la aplicación está definida en el fichero db.js.
- ▶ Los contenedores deberán estar conectados a través de una red específica de Docker y la aplicación deberá ser accesible en el puerto 3000.
- ▶ Una vez este desplegada la aplicación y ejecutándose correctamente, obtenga los logs de los contenedores desplegados.
- ▶ Por último, publique en Docker Hub la imagen generada. Si no dispone de usuario, puede crearse una cuenta de manera gratuita.

Extensión y formato

La entrega consistirá en un archivo ZIP con los ficheros de las prácticas (archivos Dockerfile y docker-compose.yaml, logs de los contenedores desplegados) y un informe en PDF explicando los pasos que se han seguido para realizar la actividad, de 1-5 páginas, fuente Arial 11, interlineado 1,5.

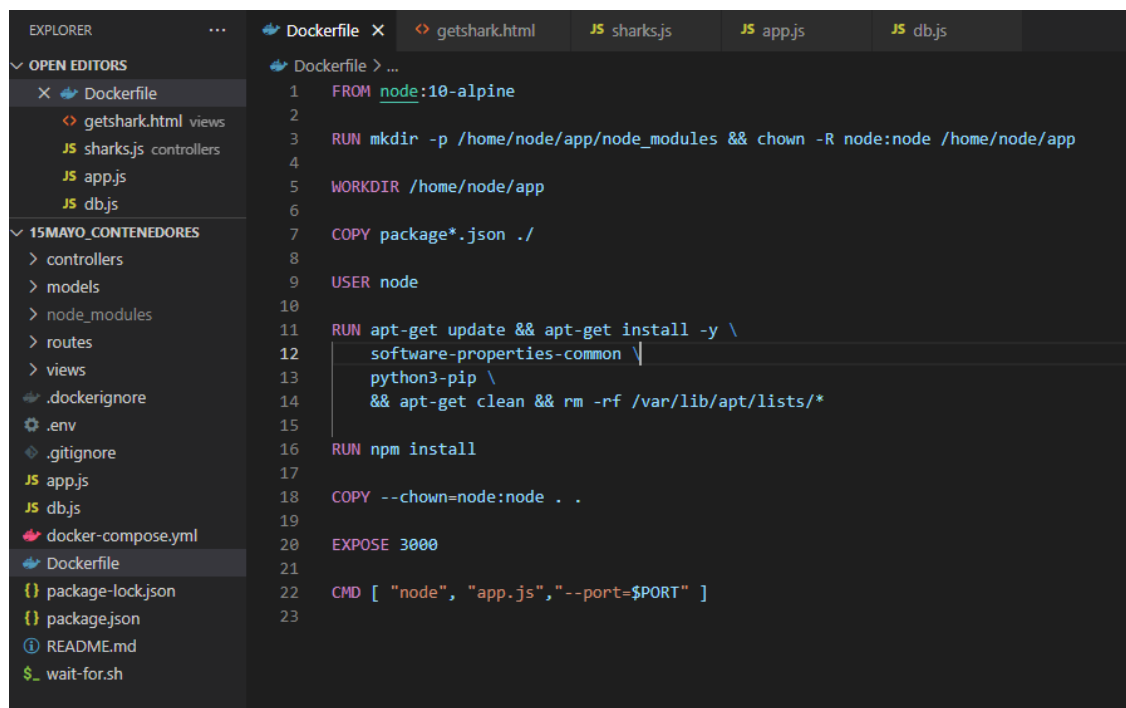
Rubrica

Construcción de imágenes Docker y despliegue con Docker Compose			
	Descripción	Puntuación máxima (10)	Peso %
Criterio 1	Es posible generar la imagen de la aplicación mediante el fichero Dockerfile	2	20%
Criterio 2	Se han seguido las buenas practicas para el Dockerfile (uso imágenes específicas, limpieza de la cache de npm, etc)	1	10%
Criterio 3	El fichero de Docker Compose funciona correctamente y se crean e inician los contenedores, volúmenes y redes.	3	30%
Criterio 4	La aplicación es accesible en el puerto especificado y funciona correctamente	2	20%
Criterio 5	Se han obtenido los logs de los contenedores	1	10%
Criterio 6	Se ha publicado la imagen de la aplicación en Docker Hub	1	10%
		10	100 %

Criterio 1

Es posible generar la imagen de la aplicación mediante el fichero Dockerfile, para cumplir con este criterio vamos a seguir los siguientes pasos:

- Creamos un dockerfile para Node donde vamos a usar la versión 10-alpine para Node.
- Creamos carpetas para los módulos y la aplicación que vamos a ejecutar
- Seteamos como directorio de trabajo el folder de la aplicación
- Copiamos el archivo package.json
- Seteamos el usuario de ejecución
- Ejecutamos update de la versión y hacemos limpieza del Docker(eliminamos paquetes descargados y se hace limpieza de cache)
- Ejecutamos el comando npm install
- Copiamos todos los archivos al Docker
- Configuramos el puerto 3000 para exponer
- Finalmente le enviamos un comando de ejecución para node, el javascript para ejecutar y definimos el puerto
- **Nota:** Se debe recalcar que usamos un archivo para las variables de ambiente utilizado para el Docker de Node y MongoDB.



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree. Under 'OPEN EDITORS', there is a 'Dockerfile' tab. The main editor area shows the content of the Dockerfile, which is a multi-line script for building a Node.js application container. The script starts with 'FROM node:10-alpine', followed by directory creation and permissions, setting the workdir, copying package.json, installing dependencies, and finally running the application on port 3000. The file tree on the left includes folders like 'controllers', 'models', 'node_modules', 'routes', and 'views', as well as files like '.dockerignore', '.env', '.gitignore', 'app.js', 'db.js', 'docker-compose.yml', 'Dockerfile', 'package-lock.json', 'package.json', 'README.md', and 'wait-for.sh'.

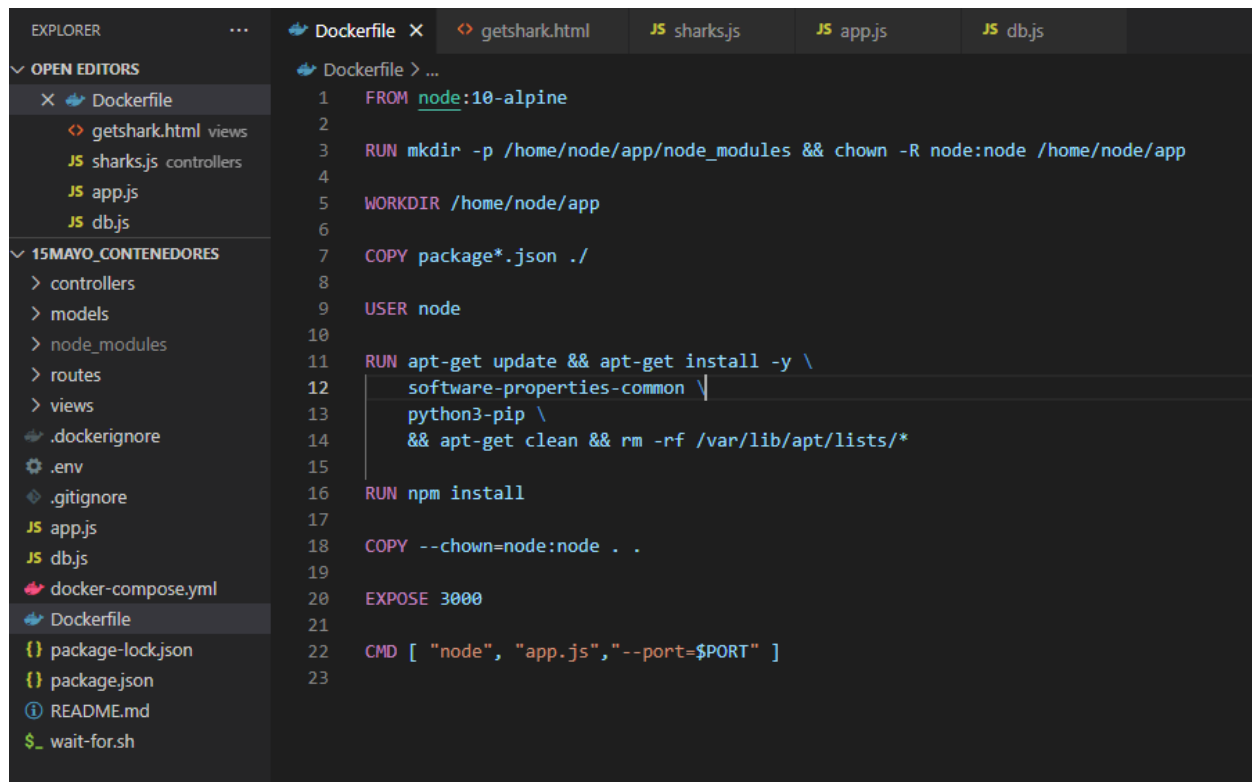
```
1 FROM node:10-alpine
2
3 RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app
4
5 WORKDIR /home/node/app
6
7 COPY package*.json ./
8
9 USER node
10
11 RUN apt-get update && apt-get install -y \
12     software-properties-common \
13     python3-pip \
14     && apt-get clean && rm -rf /var/lib/apt/lists/*
15
16 RUN npm install
17
18 COPY --chown=node:node . .
19
20 EXPOSE 3000
21
22 CMD [ "node", "app.js", "--port=$PORT" ]
23
```

Criterio 2

Se han seguido las buenas prácticas para el Dockerfile (uso imágenes específicas, limpieza de la cache de npm, etc)

Este criterio nos lleva a investigar acerca de las buenas practicas para dockerfile:

1. Usa una imagen base oficial: Es recomendable utilizar una imagen base oficial de la organización o comunidad responsable del software que se va a ejecutar en el contenedor. Las imágenes oficiales tienen actualizaciones de seguridad regulares y están bien documentadas.
2. Minimiza el tamaño de la imagen: Una imagen más pequeña significa menos superficie de ataque para posibles vulnerabilidades, y también reducirá el tiempo de descarga y la cantidad de almacenamiento necesario. Se puede minimizar el tamaño de la imagen utilizando capas intermedias y eliminando los archivos innecesarios.
3. Usa el comando COPY en lugar de ADD: El comando COPY es más simple y directo que el comando ADD. Si solo se necesita copiar archivos locales al contenedor, el comando COPY es suficiente.
4. Usa capas intermedias para reducir la cantidad de capas finales: Cada instrucción en un Dockerfile crea una nueva capa en la imagen final, lo que aumenta el tamaño de la imagen. Es recomendable usar capas intermedias para agrupar comandos relacionados y reducir la cantidad de capas finales.
5. Ejecuta solo un proceso por contenedor: Es recomendable tener solo un proceso principal en un contenedor. Esto hace que el contenedor sea más fácil de administrar y evitará problemas de escalabilidad.
6. Limpia el contenedor en la misma instrucción que se crea: Cada instrucción crea una nueva capa en la imagen final, por lo que es recomendable realizar la limpieza (eliminar archivos temporales, limpiar cachés, etc.) en la misma instrucción que se crea el archivo.
7. Usa variables de entorno: Utiliza variables de entorno en lugar de valores codificados en el Dockerfile, esto permite que la configuración pueda ser más fácilmente personalizada en diferentes entornos.
8. Comprueba los errores de ejecución: Es importante verificar si el comando que se ejecuta en el contenedor finaliza correctamente y si no lo hace, debe detenerse el contenedor para evitar fallos inesperados.
9. Documenta tu Dockerfile: Agrega comentarios y documentación detallada sobre la imagen, su uso y los comandos utilizados en el Dockerfile. Esto facilitará el mantenimiento y la actualización de la imagen en el futuro.
10. Usa nombres de imágenes y etiquetas claras: Es importante nombrar las imágenes y etiquetas de manera clara y coherente para facilitar la identificación y el uso en diferentes entornos.



The image shows a code editor interface with a sidebar on the left and a main editor area on the right. The sidebar has two sections: 'OPEN EDITORS' and '15MAYO_CONTENEDORES'. The 'OPEN EDITORS' section lists 'Dockerfile', 'getshark.html', 'sharks.js', 'app.js', and 'db.js'. The '15MAYO_CONTENEDORES' section lists various files including 'controllers', 'models', 'node_modules', 'routes', 'views', '.dockerignore', '.env', '.gitignore', 'app.js', 'db.js', 'docker-compose.yml', 'Dockerfile', 'package-lock.json', 'package.json', 'README.md', and 'wait-for.sh'. The main editor area shows the content of the 'Dockerfile' file, which is a Dockerfile for a Node.js application. The Dockerfile starts with 'FROM node:10-alpine', followed by 'RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app', 'WORKDIR /home/node/app', 'COPY package*.json ./', 'USER node', 'RUN apt-get update && apt-get install -y \ software-properties-common \ python3-pip \ && apt-get clean && rm -rf /var/lib/apt/lists/*', 'RUN npm install', 'COPY --chown=node:node . .', 'EXPOSE 3000', and 'CMD ["node", "app.js", "--port=\$PORT"]'.

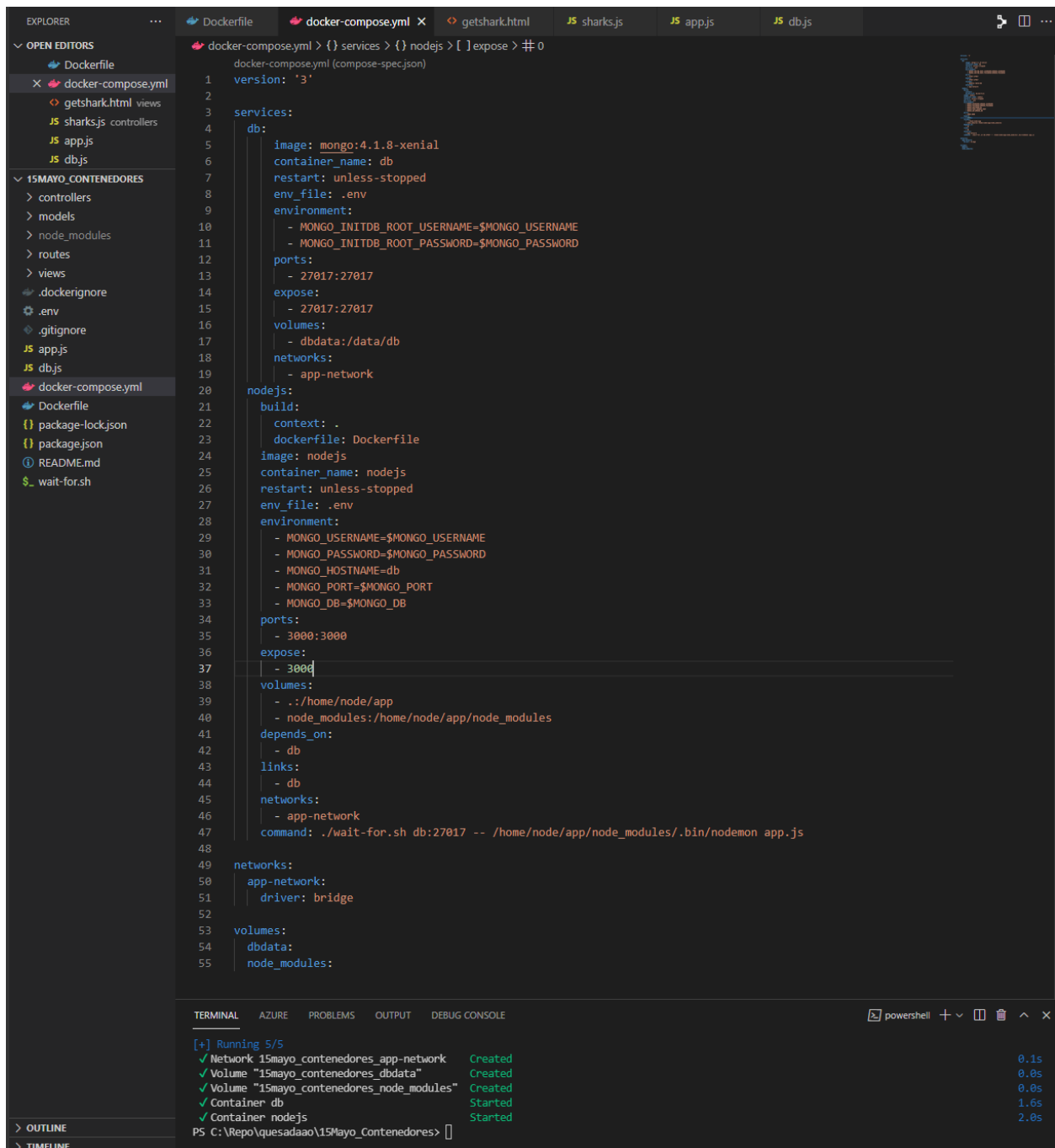
```
1 FROM node:10-alpine
2
3 RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app
4
5 WORKDIR /home/node/app
6
7 COPY package*.json ./
8
9 USER node
10
11 RUN apt-get update && apt-get install -y \
12     software-properties-common \
13     python3-pip \
14     && apt-get clean && rm -rf /var/lib/apt/lists/*
15
16 RUN npm install
17
18 COPY --chown=node:node . .
19
20 EXPOSE 3000
21
22 CMD [ "node", "app.js", "--port=$PORT" ]
23
```

Acá tenemos el Docker file cumpliendo con las mejores practicas como:

1. Uso memoria oficial
2. Uso de variables de ambiente
3. Limpieza de cache
4. Limpieza de paquetes
5. Uso de una imagen lo más liviana posible

Criterio 3

El fichero de Docker Compose funciona correctamente y se crean e inician los contenedores, volúmenes y redes. Se agrega la siguiente imagen como evidencia acerca de la ejecución del Docker compose:



The screenshot shows a Visual Studio Code editor with a Docker Compose file open. The file is named `docker-compose.yml` and is located in the `15MAYO_CONTENEDORES` directory. The file defines two services: `db` (MongoDB) and `nodejs` (Node.js application). The `db` service uses the `mongo:4.1.8-xenial` image and is connected to the `app-network`. The `nodejs` service uses the `nodejs` image and is also connected to the `app-network`. It depends on the `db` service and links to it. The command for the `nodejs` service is `./wait-for.sh db:27017 -- /home/node/app/node_modules/.bin/nodemon app.js`.

The terminal output shows the successful execution of the `docker-compose up` command, creating the network, volumes, and containers.

```
docker-compose.yml (compose-spec:json)
version: '3'

services:
  db:
    image: mongo:4.1.8-xenial
    container_name: db
    restart: unless-stopped
    env_file: .env
    environment:
      - MONGO_INITDB_ROOT_USERNAME=$MONGO_USERNAME
      - MONGO_INITDB_ROOT_PASSWORD=$MONGO_PASSWORD
    ports:
      - 27017:27017
    expose:
      - 27017:27017
    volumes:
      - dbdata:/data/db
    networks:
      - app-network
  nodejs:
    build:
      context: .
      dockerfile: Dockerfile
    image: nodejs
    container_name: nodejs
    restart: unless-stopped
    env_file: .env
    environment:
      - MONGO_USERNAME=$MONGO_USERNAME
      - MONGO_PASSWORD=$MONGO_PASSWORD
      - MONGO_HOSTNAME=db
      - MONGO_PORT=$MONGO_PORT
      - MONGO_DB=$MONGO_DB
    ports:
      - 3000:3000
    expose:
      - 3000
    volumes:
      - ../home/node/app
      - node_modules:/home/node/app/node_modules
    depends_on:
      - db
    links:
      - db
    networks:
      - app-network
    command: ./wait-for.sh db:27017 -- /home/node/app/node_modules/.bin/nodemon app.js

networks:
  app-network:
    driver: bridge

volumes:
  dbdata:
  node_modules:
```

Terminal Output:

```
[+] Running 5/5
✓ Network 15mayo_contenedores_app-network Created 0.1s
✓ Volume "15mayo_contenedores_dbdata" Created 0.0s
✓ Volume "15mayo_contenedores_node_modules" Created 0.0s
✓ Container db Started 1.6s
✓ Container nodejs Started 2.0s
PS C:\Repo\quesadaa\15Mayo_Contenedores>
```


Criterio 4

La aplicación es accesible en el puerto especificado y funciona correctamente. Para cumplir con este criterio se configura lo siguiente:

- Puerto de ejecución en el Docker file

```
Dockerfile > ...
1 FROM node:10-alpine
2
3 RUN mkdir -p /home/node/app/node_modules && chown -R node:node /home/node/app
4
5 WORKDIR /home/node/app
6
7 COPY package*.json ./
8
9 USER node
10
11 RUN apt-get update && apt-get install -y \
12     software-properties-common \
13     python3-pip \
14     && apt-get clean && rm -rf /var/lib/apt/lists/*
15
16 RUN npm install
17
18 COPY --chown=node:node . .
19
20 EXPOSE 3000
21
22 CMD [ "node", "app.js", "--port=$PORT" ]
23
```

- Puerto en el archivo de variables de ambiente

```
.env
1 MONGO_USERNAME=sammy
2 MONGO_PASSWORD=oracle
3 MONGO_PORT=27017
4 MONGO_DB=sharkinfo
5 PORT=3000
```

- Puerto de ejecución en el app.js, archivo de ejecución de la aplicación de Node.js

```
JS app.js > [0] port
1 const express = require('express');
2 const app = express();
3 const router = express.Router();
4 const db = require('./db');
5 const sharks = require('./routes/sharks');
6
7 const path = __dirname + '/views/';
8 const port = 3000;
9
10 app.engine('html', require('ejs').renderFile);
11 app.set('view engine', 'html');
12 app.use(express.urlencoded({ extended: true }));
13 app.use(express.static(path));
14 app.use('/sharks', sharks);
15
16 app.listen(port, function () {
17   console.log('Example app listening on ${port}!');
18 })
19
```

- Puerto de ejecución del Node en el Docker compose

```
docker-compose.yml > {} services > {} nodejs > [] expose > # 0
docker-compose.yml (compose-spec:json)
1  version: '3'
2
3  services:
4    db:
5      image: mongo:4.1.8-xenial
6      container_name: db
7      restart: unless-stopped
8      env_file: .env
9      environment:
10       - MONGO_INITDB_ROOT_USERNAME=$MONGO_USERNAME
11       - MONGO_INITDB_ROOT_PASSWORD=$MONGO_PASSWORD
12     ports:
13       - 27017:27017
14     expose:
15       - 27017:27017
16     volumes:
17       - dbdata:/data/db
18     networks:
19       - app-network
20   nodejs:
21     build:
22       context: .
23       dockerfile: Dockerfile
24     image: nodejs
25     container_name: nodejs
26     restart: unless-stopped
27     env_file: .env
28     environment:
29       - MONGO_USERNAME=$MONGO_USERNAME
30       - MONGO_PASSWORD=$MONGO_PASSWORD
31       - MONGO_HOSTNAME=db
32       - MONGO_PORT=$MONGO_PORT
33       - MONGO_DB=$MONGO_DB
34     ports:
35       - 3000:3000
36     expose:
37       - 3000
38     volumes:
39       - ./home/node/app
40       - node_modules:/home/node/app/node_modules
41     depends_on:
42       - db
43     links:
44       - db
45     networks:
46       - app-network
47     command: ./wait-for.sh db:27017 -- /home/node/app/node_modules/.bin/nodemon app.js
48
49   networks:
50     app-network:
51       driver: bridge
52
53   volumes:
54     dbdata:
55     node_modules:
```

- Evidencia de ejecución correcta

UNIR - Everything Sharks Home Sharks

Want to Learn About Sharks?

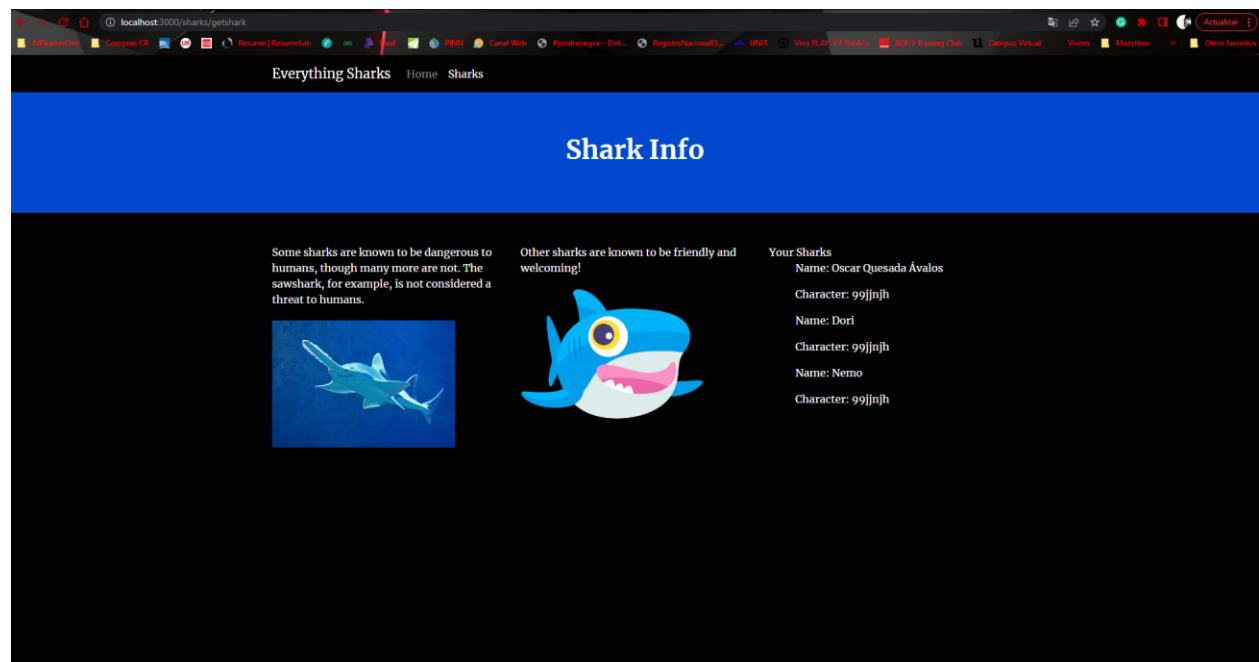
Are you ready to learn about sharks?

[Get Shark Info](#)

CONTENEDORES :) - Not all sharks are alike
Though some are dangerous, sharks generally do not attack humans. Out of the 500 species known to researchers, only 30 have been known to attack humans.

15 - Mayo - 2023 / Sharks are ancient
There is evidence to suggest that sharks lived up to 400 million years ago.
Hecha por Oscar Quesada Avalos para el curso de Contenedores

La siguiente imagen muestra la sesión Your Sharks, donde vemos reflejados los tibureros guardados en la base de datos



Criterio 5

Se han obtenido los logs de los contenedores (si se desea puede revisar un archivo llamado LogEvidencia.txt que contiene los logs por escrito)

```
PS C:\Users\OscarQ\Documents> docker-compose logs
nodejs | [Entrypoint] 1.0.0
nodejs | [Entrypoint] to restart at any time, enter 'rs'
nodejs | [Entrypoint] watching extensions: js,css,json
nodejs | [Entrypoint] watching extensions: js,css,json
nodejs | [Entrypoint] watching extensions: js,css,json
nodejs | Example app listening on $(port)
nodejs | MongoDB is connected
db | 2023-05-15T03:42:09.225+0000 W CONTROL [main] Option: saslMode is deprecated. Please use tlsMode instead.
db | about to fork child process, waiting until server is ready for connections.
db | forked process: 26
db | 2023-05-15T03:42:09.239+0000 I CONTROL [main] ***** SERVER RESTARTED *****
db | 2023-05-15T03:42:09.243+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
db | 2023-05-15T03:42:09.248+0000 I CONTROL [initandlisten] MongoDB starting : pid=26 port=27017 dbpath=/data/db 64-bit host=f4cbe06c123
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] db version v4.8
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] git version: 532a2b353f7e6d11a1889a5c2323ccf3228
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.2g 1 Mar 2016
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] allocator: tcmalloc
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] modules: none
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] build environment:
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] distarch: x86_64
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] distarch: x86_64
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] target_arch: x86_64
db | 2023-05-15T03:42:09.249+0000 I CONTROL [initandlisten] options: { net: { bindIp: '*127.0.0.1', port: 27017, ssl: { mode: 'disabled' } }, processManagement: { fork: true, pidFilePath: '/tmp/docker-entrypoint-temp-mongodb.pid' }, systemLog: { destination: 'file', logAppend: true, path: '/tmp/1/f6/f1' } }
db | 2023-05-15T03:42:09.250+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
db | 2023-05-15T03:42:09.250+0000 I STORAGE [initandlisten] ** See http://wiki.mongodb.org/wiki/xfsrecommendedfilesystem
db | 2023-05-15T03:42:09.250+0000 I STORAGE [initandlisten] WiredTiger open config: create,cache_size=628M,session_max=9999,eviction{(threads_min=4,threads_max=4),config_base=false,statistics{(fast),log{(enabled=true,archive=true,path=journal,compressor=snappy)},file_manager{(close,(file_timeout=60)),statistics_log{(wait=6),verbosity(recovery_progress),0}},2023-05-15T03:42:09.799+0000 I STORAGE [initandlisten] WiredTiger message [1684122129:788990][26:b7f4d5cf8aa8], txn-recover: Set global recovery timestamp: (0,0)
db | 2023-05-15T03:42:09.816+0000 I RECOVERY [initandlisten] WiredTiger recovery/restore: To: timestamp(0, 0)
db | 2023-05-15T03:42:09.849+0000 I STORAGE [initandlisten] Timestamp monitor starting
db | 2023-05-15T03:42:09.850+0000 I CONTROL [initandlisten]
db | 2023-05-15T03:42:09.850+0000 I CONTROL [initandlisten] ** NOTE: This is a development version (4.8.0) of MongoDB.
db | 2023-05-15T03:42:09.850+0000 I CONTROL [initandlisten] ** Not recommended for production.
db | 2023-05-15T03:42:09.850+0000 I CONTROL [initandlisten]
db | 2023-05-15T03:42:09.859+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
db | 2023-05-15T03:42:09.859+0000 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
db | 2023-05-15T03:42:09.859+0000 I CONTROL [initandlisten]
db | 2023-05-15T03:42:09.859+0000 I CONTROL [initandlisten]
db | 2023-05-15T03:42:09.859+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mem/transparent_hugepage/enabled is 'always'.
db | 2023-05-15T03:42:09.859+0000 I CONTROL [initandlisten] ** We suggest setting it to 'never'
db | 2023-05-15T03:42:09.859+0000 I CONTROL [initandlisten]
db | 2023-05-15T03:42:09.860+0000 I STORAGE [initandlisten] createCollection: admin.system.version with provided UUID: 87977183-602d-40b0-b074-e21ae74e278f
db | 2023-05-15T03:42:09.860+0000 I INDEX [initandlisten] index build: done building index _id, on ns admin.system.version
db | 2023-05-15T03:42:09.891+0000 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: 'unsharded'
db | 2023-05-15T03:42:09.891+0000 I SHARDING [initandlisten] Marking collection local.system.replset as collection version: 'unsharded'
db | 2023-05-15T03:42:09.891+0000 I SHARDING [initandlisten] Marking collection admin.system.roles as collection version: 'unsharded'
db | 2023-05-15T03:42:09.897+0000 I STORAGE [initandlisten] createCollection: local.startup.log with generated UUID: 407f452-7aed-431e-a111-231d48747f34
db | 2023-05-15T03:42:09.915+0000 I INDEX [initandlisten] index build: done building index _id, on ns local.startup.log
db | 2023-05-15T03:42:09.916+0000 I SHARDING [initandlisten] Marking collection local.startup.log as collection version: 'unsharded'
db | 2023-05-15T03:42:09.916+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory '/data/db/diagnostic.data'
db | 2023-05-15T03:42:09.917+0000 I NETWORK [initandlisten] Listening on /tmp/mongodb-27017.sock
db | 2023-05-15T03:42:09.917+0000 I SHARDING [initandlisten] LogicalSessionCacheRefresh Marking collection config.system.sessions as collection version: 'unsharded'
db | 2023-05-15T03:42:09.917+0000 I NETWORK [initandlisten] Listening on 127.0.0.1
db | 2023-05-15T03:42:09.917+0000 I NETWORK [initandlisten] Marking for connections on port 27017
db | 2023-05-15T03:42:09.918+0000 I STORAGE [initandlisten] LogicalSessionCacheRefresh createCollection: config.system.sessions with generated UUID: a5b386c6-af1f-4b22-a5dc-a5f047aaad1
db | child process started successfully, parent exiting
db | 2023-05-15T03:42:09.937+0000 I INDEX [LogicalSessionCacheRefresh] index build: done building index _id, on ns config.system.sessions
db | 2023-05-15T03:42:09.957+0000 I INDEX [LogicalSessionCacheRefresh] index build: starting on config.system.sessions properties: { v: 2, key: { lastuse: 1 }, name: 'lsidTTLIndex', ns: 'config.system.sessions', expireAfterSeconds: 1800 } using method: Hybrid
db | 2023-05-15T03:42:09.957+0000 I INDEX [LogicalSessionCacheRefresh] build may temporarily use up to 2MB separate of RAM
db | 2023-05-15T03:42:09.957+0000 I INDEX [LogicalSessionCacheRefresh] index build: collection scan done. scanned 0 total records in 0 seconds
db | 2023-05-15T03:42:09.957+0000 I INDEX [LogicalSessionCacheRefresh] index build: inserted 0 keys from external source into index in 0 seconds
db | 2023-05-15T03:42:09.958+0000 I INDEX [LogicalSessionCacheRefresh] index build: done building index lsidTTLIndex on ns config.system.sessions
db | 2023-05-15T03:42:09.970+0000 I NETWORK [listener] connection accepted from 127.0.0.1:33538 #1 (1 connection now open)
db | 2023-05-15T03:42:09.970+0000 I NETWORK [conn=1] received client metadata from 127.0.0.1:33538 conn=1 { application: { name: 'MongoDB Shell', driver: { name: 'MongoDB Internal Client', version: '4.8.1' }, os: { type: 'Linux', name: 'Ubuntu', architecture: 'x86_64', ver: '16.04' } } }
db | 2023-05-15T03:42:09.983+0000 I NETWORK [conn=1] connection 127.0.0.1:33538 (0 connections now open)
db | 2023-05-15T03:42:10.004+0000 I NETWORK [listener] connection accepted from 127.0.0.1:33532 #2 (1 connection now open)
db | 2023-05-15T03:42:10.004+0000 I NETWORK [conn=2] received client metadata from 127.0.0.1:33532 conn=2 { application: { name: 'MongoDB Shell', driver: { name: 'MongoDB Internal Client', version: '4.8.1' }, os: { type: 'Linux', name: 'Ubuntu', architecture: 'x86_64', ver: '16.04' } } }
```

```
db | 2023-05-15T03:42:18.080+0000 I SHARDING [conn0] Marking collection admin.system.users as collection version: <unsharded>
db | 2023-05-15T03:42:18.080+0000 I STORAGE [conn0] createCollection: admin.system.users with generated UUID: 426f6a82-6d3c-4b72-8d4c-b9160a7311c3
db | 2023-05-15T03:42:18.106+0000 I INDEX [conn0] index build: done building index_id_0n ns admin.system.users$0 | 2023-05-15T03:42:18.117+0000 I INDEX [conn0] index build: done building index_user_1_db_1 on ns admin.system.users
db | Successfully added user: {
db |   "user": "sammy",
db |   "roles": [
db |     {
db |       "role": "root",
db |       "db": "admin"
db |     }
db |   ]
db | }
db | 2023-05-15T03:42:18.120+0000 E - [main] Error saving history file: fileOpenFailed: unable to open() file /home/mongodb/.dshshell: Unknown error
db | 2023-05-15T03:42:18.121+0000 I NETWORK [conn0] end connection 127.0.0.1:33332 (0 connections now open)
db | /usr/local/bin/docker-entrypoint.sh: Ignoring /docker-entrypoint-initdb.d/*
db | 2023-05-15T03:42:18.136+0000 W CONTROL [main] Option: sslMode is deprecated. Please use tlsMode instead.
db | 2023-05-15T03:42:18.137+0000 I CONTROL [main] *** SERVER STARTED ***
db | 2023-05-15T03:42:18.138+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
db | Killing process with pid: 26
db | 2023-05-15T03:42:18.161+0000 I CONTROL [signalProcessingThread] got signal 15 (Terminated), will terminate after current cmd ends
db | 2023-05-15T03:42:18.161+0000 I NETWORK [signalProcessingThread] shutdown: going to close listening sockets...db | 2023-05-15T03:42:18.161+0000 I NETWORK [signalProcessingThread] removing socket file: /tmp/mongod-27017.sockdb | 2023-05-15T03:42:18.162+0000 I
db | 2023-05-15T03:42:18.163+0000 I FTDC [signalProcessingThread] Shutting down full-time diagnostic data capture
db | 2023-05-15T03:42:18.166+0000 I STORAGE [signalProcessingThread] Timestamp monitor shutting down
db | 2023-05-15T03:42:18.166+0000 I STORAGE [signalProcessingThread] WiredTigerEngine shutting down
db | 2023-05-15T03:42:18.166+0000 I STORAGE [signalProcessingThread] Shutting down session sweeper thread
db | 2023-05-15T03:42:18.224+0000 I STORAGE [signalProcessingThread] Finished shutting down session sweeper threaddb | 2023-05-15T03:42:18.166+0000 I STORAGE [signalProcessingThread] Shutting down journal flusher thread
db | 2023-05-15T03:42:18.224+0000 I STORAGE [signalProcessingThread] Finished shutting down journal flusher threaddb | 2023-05-15T03:42:18.224+0000 I STORAGE [signalProcessingThread] Shutting down checkpoint thread
db | 2023-05-15T03:42:18.283+0000 I STORAGE [signalProcessingThread] shutdown: reoving fs lock...
db | 2023-05-15T03:42:18.283+0000 I STORAGE [signalProcessingThread] Now exiting
db | 2023-05-15T03:42:18.283+0000 I CONTROL [signalProcessingThread] shutting down with code 8
db |
db | MongoDB init process complete; ready for start up.
db |
db | 2023-05-15T03:42:11.179+0000 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] MongoDB starting : pid=1 port=27017 dbpath=/data/db 64-bit host=acbe96cd13
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] db version v6.1.0
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] git version: 638a26bc5387de1dd131a18881ad3253cccf3220
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.2g 1 Mar 2016
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] Allocator: tcmalloc
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] modules: none
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] build environment:
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] distarch: shuntuishm
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] distarch: x86_64
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] target_arch: x86_64
db | 2023-05-15T03:42:11.179+0000 I CONTROL [initandlisten] options: { net: { bindip: "*" }, security: { authentication: "enabled" } }
db | 2023-05-15T03:42:11.179+0000 I STORAGE [initandlisten] Detected data files in /data/db created by the "wiredtiger" storage engine, an setting the active storage engine to "wiredtiger".
db | 2023-05-15T03:42:11.179+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
db | 2023-05-15T03:42:11.179+0000 I STORAGE [initandlisten] See http://dochub.mongodb.org/core/products-filesystem
db | 2023-05-15T03:42:11.179+0000 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=320M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager(
db | 2023-05-15T03:42:11.179+0000 I STORAGE [initandlisten] Verbose(recovery_progress)db | 2023-05-15T03:42:11.802+0000 I STORAGE [initandlisten] WiredTiger message [1680122131.966370][1:8c7f0eef6da8b], txn-recover: Recovering log 1 through 1
db | 2023-05-15T03:42:11.966+0000 I STORAGE [initandlisten] WiredTiger message [1680122131.966370][1:8c7f0eef6da8b], txn-recover: Recovering log 1 through 1
db | 2023-05-15T03:42:12.123+0000 I STORAGE [initandlisten] WiredTiger message [1680122132.123799][1:8c7f0eef6da8b], txn-recover: Set global recovery timestamp: (0,0)
db | 2023-05-15T03:42:12.150+0000 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp: Ts: Timestamp(0, 0)
db | 2023-05-15T03:42:12.161+0000 I STORAGE [initandlisten] Timestamp monitor starting
db | 2023-05-15T03:42:12.166+0000 I CONTROL [initandlisten] ** NOTE: This is a development version (6.1.0) of MongoDB.
db | 2023-05-15T03:42:12.166+0000 I CONTROL [initandlisten] Not recommended for production.
db | 2023-05-15T03:42:12.260+0000 I CONTROL [initandlisten]
db | 2023-05-15T03:42:12.260+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/debug/transparent_hugepage/enabled is 'always'.
db | 2023-05-15T03:42:12.260+0000 I CONTROL [initandlisten] We suggest setting it to 'never'.
db | 2023-05-15T03:42:12.366+0000 I CONTROL [initandlisten]
db | 2023-05-15T03:42:12.366+0000 I CONTROL [initandlisten] Marking collection local.system.replset as collection version: <unsharded>
db | 2023-05-15T03:42:12.366+0000 I SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <unsharded>
db | 2023-05-15T03:42:12.366+0000 I SHARDING [initandlisten] Marking collection admin.system.version as collection version: <unsharded>
```

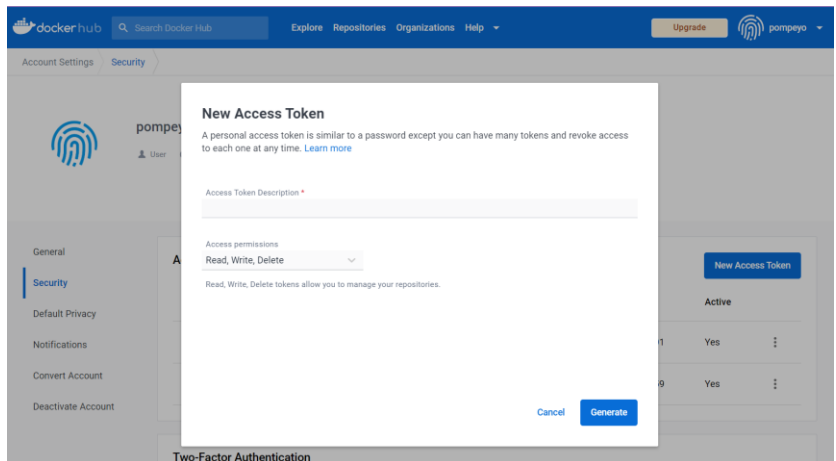
```
db | 2023-05-15T03:42:12.380+0000 I SHARDING [initandlisten] Marking collection local.startup_log as collection version: <unsharded>
db | 2023-05-15T03:42:12.390+0000 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory /data/db/diagnostic.data
db | 2023-05-15T03:42:12.391+0000 I NETWORK [initandlisten] listening on /tmp/mongod-27017.sock
db | 2023-05-15T03:42:12.391+0000 I NETWORK [initandlisten] listening on 0.0.0.0
db | 2023-05-15T03:42:12.391+0000 I SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as collection version: <unsharded>
db | 2023-05-15T03:42:12.391+0000 I NETWORK [initandlisten] waiting for connections on port 27017
db | 2023-05-15T03:42:12.432+0000 I NETWORK [listener] connection accepted from 172.29.0.3:35803 #1 (1 connection now open)
db | 2023-05-15T03:42:12.432+0000 I NETWORK [conn0] and connection 172.29.0.3:35803 (0 connections now open)
db | 2023-05-15T03:42:12.432+0000 I NETWORK [listener] connection accepted from 172.29.0.3:40828 #2 (1 connection now open)
db | 2023-05-15T03:42:12.432+0000 I NETWORK [conn2] received client metadata from 172.29.0.3:40828 conn2: { driver: { name: "nodejs", version: "3.1.13" }, os: { type: "Linux", name: "Linux", architecture: "x64", version: "5.10.102.1-microsoft-standard-WSL2" }, platform: "N
db | 2023-05-15T03:42:12.432+0000 I NETWORK [conn2] received client metadata from 172.29.0.3:40828 conn2: { driver: { name: "nodejs", version: "3.1.13" }, os: { type: "Linux", name: "Linux", architecture: "x64", version: "5.10.102.1-microsoft-standard-WSL2" }, platform: "N
db | 2023-05-15T03:42:12.432+0000 I SHARDING [conn0] Marking collection admin.system.users as collection version: <unsharded>
db | 2023-05-15T03:42:12.432+0000 I ACCESS [conn0] Successfully authenticated as principal sammy on admin
db | PS C:\Users\quasadaa\15days\Containers>
```

Criterio 6

Se ha publicado la imagen de la aplicación en Docker Hub

Para poder cumplir con este criterio necesitamos tener claridad de los pasos para subir una imagen en el repositorio de Docker Hub:

1. Crear un token con los permisos requeridos



2. Crear un repositorio para guardar las imágenes

The screenshot shows the 'Create repository' page on Docker Hub. The 'Namespace' is set to 'pompeyo'. The 'Repository Name' field is empty. There is a 'Description' field. Under 'Visibility', the 'Public' option is selected, indicating it will appear in search results. A 'Pro tip' section on the right provides CLI commands: `docker tag local-image:tagname new-repo:tagname` and `docker push new-repo:tagname`, with a note to change the tagname. At the bottom are 'Cancel' and 'Create' buttons.

3. Hacer login con el usuario y utilizando el token creado

4. Se debe establecer un tag a la imagen creada

`docker tag mi-imagen:latest usuario/repo:latest`

5. Hacer push de la imagen hacia el repositorio

`docker push usuario/repo:latest`

```
PS C:\Repo\quesadaao\15Mayo_Contenedores> docker push pompeyo/boom:latest
7056e146b2aa: Pushed
3c93619e4bc4: Pushed
614ab7d38592: Pushed
8b27f67bc3d8: Pushed
de915e575d22: Pushed
47b1abdddeb29: Pushed
4f4fb700ef54: Pushed
a1db1b91f474: Pushed
ddad3d7c1e96: Pushed
7150aa69525b: Pushed
d7aa47be044e: Pushed
latest: digest: sha256:614ab7d38592c7f952e1f38bb3ab135073406da9d07acaacf29cbea9d639ff4, size: 2199
PS C:\Repo\quesadaao\15Mayo_Contenedores>
```

The screenshot shows the Docker Hub page for the repository 'pompeyo/boom'. The 'General' tab is active. A message prompts the user to add a short description. The repository is currently empty, with a message 'This repository does not have a description'. The 'Last pushed' time is '6 minutes ago'. The 'Tags' section shows one tag, 'latest', which is an 'Image' type, pushed '6 minutes ago'. The 'Automated Builds' section is empty, with a note that it is available with Pro, Team, and Business subscriptions. A 'Public View' button is visible in the 'Docker commands' section.

Tag	OS	Type	Pulled	Pushed
latest		Image	---	6 minutes ago

Errores

Cabe recalcar que para esta actividad se cuenta con experiencia previa acerca de hacer dockers y utilización de Docker compose por tanto los errores presentados son nulos. Se usa una pagina de referencia para implementar los requerimientos y con eso fue suficiente para realizar la actividad.

Referencias

Guia Node + Mongo:

<https://www.digitalocean.com/community/tutorials/containerizing-a-node-js-application-for-development-with-docker-compose>

Repo 1

<https://github.com/fazt/nodejs-rest-auth/tree/master>