



Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
División de Tecnologías para la Integración Ciber-Humana
Departamento de Ciencias Computacionales
Ingeniería en Computación



Docker Container

Ejemplo de uso de Docker Container.

Alumno

Hernández Cortez Kevin Uriel.

217734547.

Materia

Computación Tolerante a Fallas.

D06

Profesor

Lopez Franco Michel Emanuel.

Fecha de entrega

Lunes 22 de abril de 2024.

Introducción

En esta tarea, utilizaré Docker para desarrollar y desplegar una aplicación web basada en Node.js. Docker es una plataforma de código abierto que permite empaquetar, distribuir y ejecutar aplicaciones dentro de contenedores. Estos contenedores son entornos ligeros y autónomos que contienen todo lo necesario para ejecutar una aplicación, incluidas las bibliotecas, las dependencias y el código. Utilizaré Docker para aprender más sobre esta alternativa, y usarla en mis futuros proyectos.

Desarrollo

Códigos

```
//SERVER.JS--
const express = require('express');
const mongoose = require('mongoose');
const app = express();
//para conectar a MongoDB
mongoose.connect('mongodb://mongo-db:27017/mydatabase', { useNewUrlParser:
true, useUnifiedTopology: true })
    .then(() => console.log('Connected to MongoDB'))
    .catch(err => console.error('Error connecting to MongoDB', err));
app.get('/', (req, res) => {
    res.send("Welcome to my awesome app v2!");
});
app.listen(3000, function() {
    console.log("Web service listening on port 3000");
});
```

```
#DOCKERFILE--
FROM node:14-alpine
#copiar archivos de la aplicación
COPY . /app/
#establecer el directorio de trabajo
WORKDIR /app
#instalar express y mongoose
RUN npm install express@4.19.2 mongoose@5.13.2
#comando para iniciar la aplicación
CMD [ "node", "server.js" ]
```

```
//PACKAGE.JSON
{
  "name": "web-service",
  "version": "1.0",
  "dependencies": {
    "express": "4.19.2",
    "mongoose": "^5.13.2"
  }
}
```

}

Explicación de programa

Se hace un ejemplo para Docker Container. Primeramente, utilizo Node.js y npm para desarrollar la aplicación web. Express.js es un framework web para Node.js que simplifica el proceso de desarrollo de aplicaciones web. Mongoose es una biblioteca de modelado de objetos MongoDB para Node.js, que da una solución sencilla y basada en esquemas para modelar los datos de la aplicación.

Una vez entendiendo esto, se crea la ruta específica para el proyecto en una carpeta llamada “web-service”. Una vez estando ahí, se crean los archivos anteriormente descritos: *server.js*, *Dockerfile*, *package.json*.

El archivo **Dockerfile** es un archivo de texto que contiene los comandos necesarios para construir una imagen Docker. En este caso, el Dockerfile especifica cómo construir una imagen que contiene nuestra aplicación web y todas sus dependencias.

El archivo **package.json** es un archivo de metadatos para el proyecto. Tiene información sobre el nombre del proyecto, la versión, las dependencias del proyecto y otros metadatos relevantes.

Finalmente, el archivo **server.js** es el archivo principal de la aplicación. Tiene el código de la aplicación web que será ejecutado por Node.js.





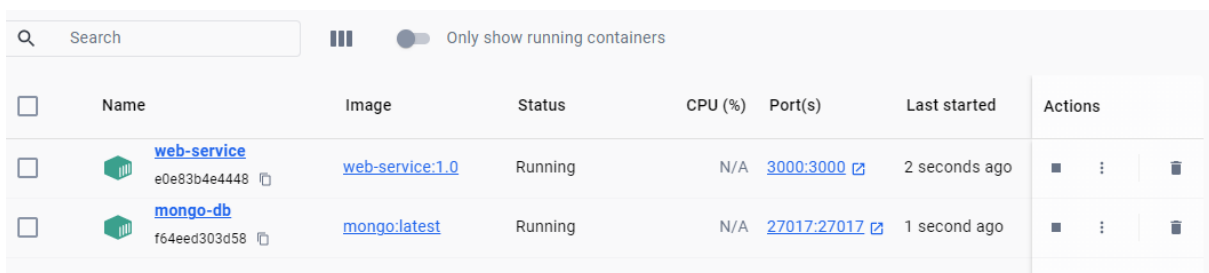
Nombre	Fecha de modificación	Tipo	Tamaño
 .dockerignore	21/04/2024 05:23 p. m.	Archivo DOCKERI...	1 KB
 Dockerfile	22/04/2024 11:07 a. m.	Archivo	1 KB
 package	22/04/2024 11:38 a. m.	Archivo de origen ...	1 KB
 server.js	22/04/2024 11:08 a. m.	JSFile	1 KB

Ilustración 1. Archivos en la ruta del proyecto.

Luego, se crea la imagen Docker. En una nueva terminal, se escribe el comando: *docker build -t node-app:1.0* . Se verifica que se haya creado correctamente con *docker images*. Luego, se ejecuta el contenedor Docker con el comando: *docker run -d -p 3000:3000 node-app:1.0*.

Una vez hecho esto, podemos verificar el estado del contenedor con el comando *docker logs web-service*. En la aplicación de Docker, se puede verificar que ya se cuenta con los contenedores necesarios para la aplicación.

Apache Airflow



The screenshot shows the Docker Desktop interface. At the top, there is a search bar and a toggle switch labeled 'Only show running containers'. Below this is a table listing the running containers. The table has columns for Name, Image, Status, CPU (%), Port(s), Last started, and Actions. Two containers are listed: 'web-service' and 'mongo-db'. Both are in a 'Running' state. The 'web-service' container is mapped to port 3000 on the host, and the 'mongo-db' container is mapped to port 27017.

	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	web-service e0e83b4e4448	web-service:1.0	Running	N/A	3000:3000	2 seconds ago	■ ⋮ 🗑
<input type="checkbox"/>	mongo-db f64eed303d58	mongo:latest	Running	N/A	27017:27017	1 second ago	■ ⋮ 🗑

Ilustración 2. Contenedores necesarios para la aplicación.

Resultados

Como resultados, obtenemos la página con el ejemplo al hacer clic en el puerto de web-service 3000:3000.

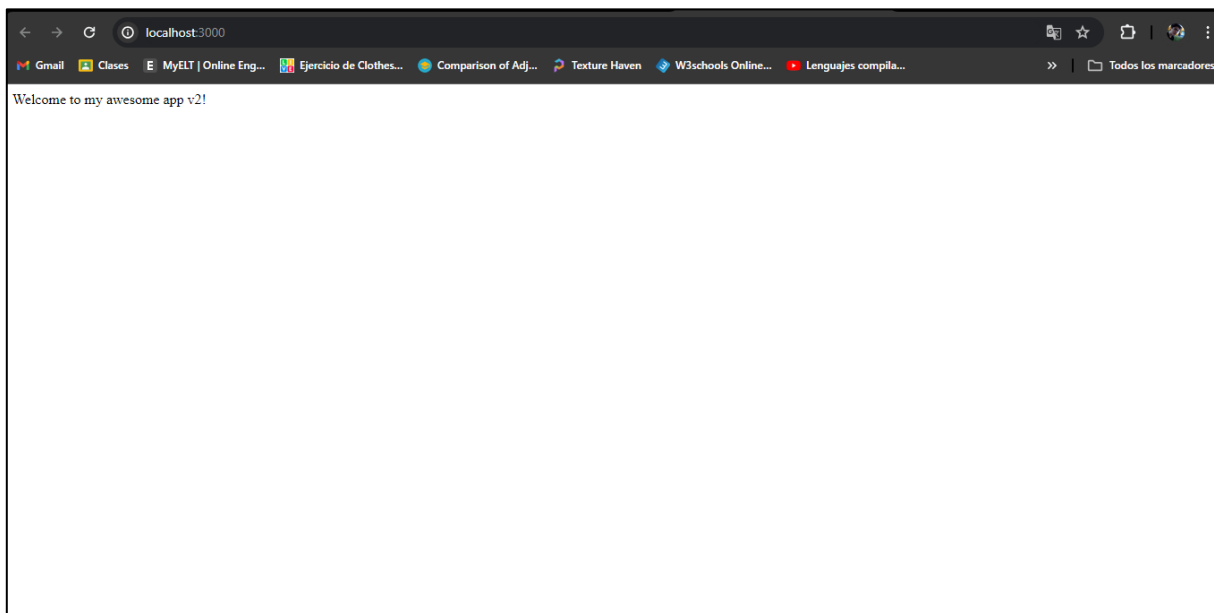


Ilustración 3. Resultados.

Conclusión

En esta tarea, se utilizó Docker para empaquetar una aplicación web basada en Node.js, lo que me permitió crear un entorno de ejecución autónomo para la aplicación. He aprendido que Docker es una gran herramienta al momento de simplificar el proceso de distribución y ejecución de aplicaciones. Este termina garantizando su portabilidad y consistencia en diferentes entornos. Además, el uso de Docker facilita el mantenimiento de la aplicación, al tener un entorno aislado y autocontenido que puede ser fácilmente desplegado en cualquier lugar.

Bibliografía.

- «*docker scout quickview*». (2024, 9 febrero). Docker Documentation.
<https://docs.docker.com/reference/cli/docker/scout/quickview/>
- «*Docker Scout image analysis*». (2024, 3 abril). Docker Documentation.
<https://docs.docker.com/scout/image-analysis/>
- *Docker*. (s. f.).
https://hub.docker.com/_/node?tab=tags&page=1&ordering=last_updated
- Rojas, N. F. O. (2022, 6 enero). Creando un proyecto Node JS con Docker - Nicolás F. Ormeño Rojas - Medium. *Medium*. <https://normeno.medium.com/creando-un-proyecto-node-js-con-docker-8a3f18240206>