Nombre: Eduardo Quetzal Delgado Pimentel

Código: 217239716

Fecha: 21/04/2024

Materia: Computación tolerante a fallas

Ejemplo utilizando Docker

Introduccion:

Docker es una plataforma de software que permite a los desarrolladores empaquetar, distribuir y ejecutar aplicaciones dentro de contenedores. Estos contenedores son entornos ligeros y portátiles que encapsulan todo lo necesario para que una aplicación se ejecute de manera consistente, incluyendo el código, las bibliotecas, las dependencias y las variables de entorno. Con Docker, los desarrolladores pueden eliminar las inconsistencias entre los entornos de desarrollo, pruebas y producción, lo que facilita la creación, implementación y escalado de aplicaciones de manera rápida y eficiente.

Un microservicio es un enfoque arquitectónico para el desarrollo de software en el que una aplicación se divide en componentes pequeños e independientes, cada uno de los cuales se enfoca en realizar una tarea específica. Estos componentes, conocidos como microservicios, son desarrollados, desplegados y gestionados de forma independiente, lo que permite a los equipos de desarrollo trabajar de manera más ágil y eficiente.

Cada microservicio opera como un servicio independiente y se comunica con otros servicios a través de interfaces bien definidas, como APIs (Interfaces de Programación de Aplicaciones) RESTful o mensajes. Esto facilita la integración, el mantenimiento y la escalabilidad de la aplicación, ya que cada microservicio puede ser desarrollado utilizando diferentes tecnologías, escalado de manera independiente y actualizado sin afectar a otros componentes del sistema.

La arquitectura de microservicios promueve el modularidad, la flexibilidad y la escalabilidad, lo que facilita la construcción y el mantenimiento de aplicaciones complejas en entornos de desarrollo distribuidos. Además, permite a las organizaciones adoptar prácticas de desarrollo ágil y DevOps para mejorar la velocidad de entrega y la calidad del software.

Desarrollo:

Lo que tengo pensado es realizar un swarm ya que lo vi en un video de youtube, la idea principal es realizar un Docker swarm con un deploy de tres replicas para ver como va atacando un servidor.

Lo primero es realizar los archivos.

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
    console.log("Hey Youtube!")
  res.send('Hello World!');
});

app.listen(8080, function () {
  console.log('Example app listening on port 8080!');
});
```

1. Importa el módulo 'express': **var express = require('express');**. Esto carga el framework Express para crear y manejar servidores web en Node.js.
2. Crea una instancia de la aplicación Express: **var app = express();**. Esta línea inicializa una aplicación Express.
3. Define una ruta utilizando el método **app.get()**. En este caso, la ruta raíz **'/'** responde a las solicitudes GET. Cuando un cliente solicita la ruta raíz de este servidor, la función de devolución de llamada se ejecutará. ¡Esta función imprime “Hey Youtube!" en la consola del servidor y luego envía la respuesta "Hello World!" al cliente.
4. Inicia el servidor en el puerto 8080 utilizando **app.listen()**. Esto hace que el servidor escuche las solicitudes entrantes en el puerto 8080. ¡Cuando el servidor se inicia correctamente, imprime “Example app listening on port 8080!" en la consola del servidor.

¡En resumen, este código crea un servidor web simple que responde con “Hello World!" cuando se accede a la ruta raíz '/' y también imprime "Hey Youtube!" en la consola del servidor cada vez que se realiza una solicitud a esa ruta.

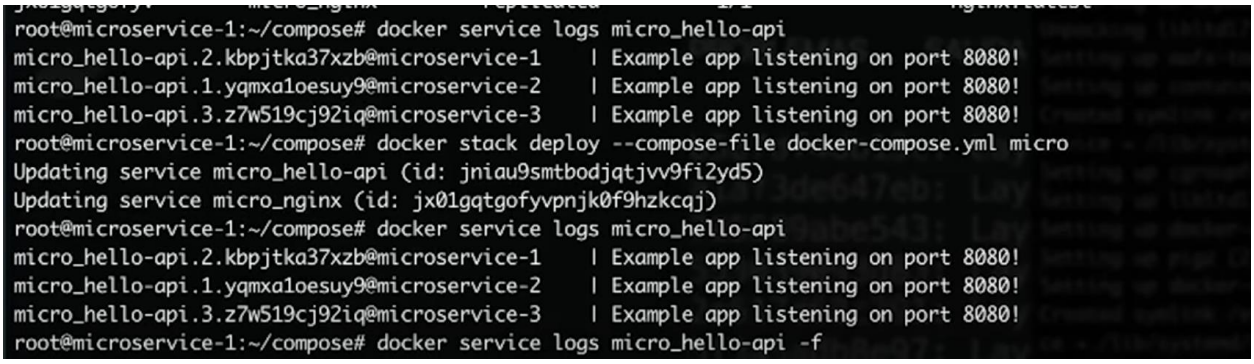
Luego se realiza el dockerfile.

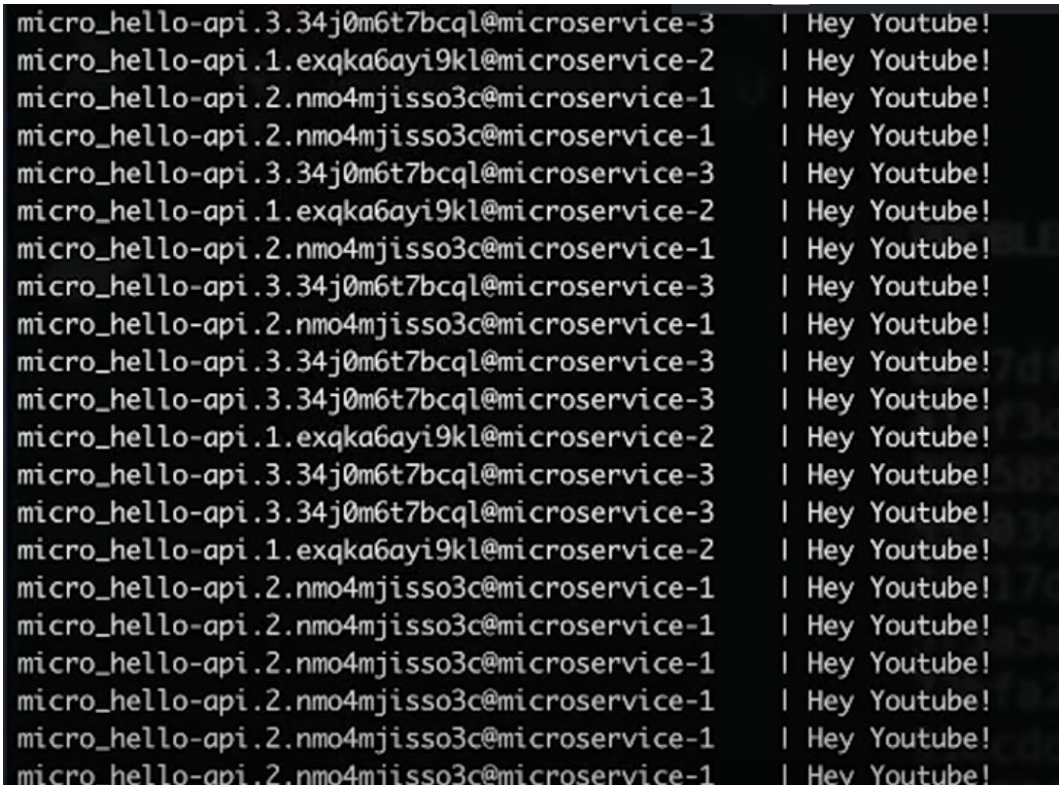
```
FROM node:10
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .

EXPOSE 8080
CMD [ "node", "app.js" ]
```

En resumen, este Dockerfile construye una imagen Docker que contiene una aplicación Node.js, copia los archivos de la aplicación y las dependencias, expone el puerto 8080 y ejecuta la aplicación al iniciar el contenedor.

Resultados:





Conclusión:

En este proyecto, hemos explorado dos conceptos fundamentales en el desarrollo de software moderno: Docker y la arquitectura de microservicios. Utilizando Docker, pudimos empaquetar nuestra aplicación Node.js en un contenedor ligero y portátil, lo que nos permitió distribuir y ejecutar la aplicación de manera consistente en diferentes entornos.

Además, adoptamos la arquitectura de microservicios para descomponer nuestra aplicación en componentes independientes y escalables. Esto nos permitió desarrollar, desplegar y gestionar cada parte de la aplicación de forma independiente, facilitando así el desarrollo ágil y la integración continua.

Al implementar un Docker Swarm con un despliegue de tres réplicas de nuestra aplicación, pudimos observar cómo el enfoque de microservicios puede mejorar la escalabilidad y la resiliencia de una aplicación al distribuir la carga entre varios nodos del clúster.

En conclusión, Docker y la arquitectura de microservicios ofrecen herramientas poderosas para el desarrollo y la implementación de aplicaciones modernas. Al adoptar estas tecnologías, las organizaciones pueden mejorar la flexibilidad, la escalabilidad y la confiabilidad de sus sistemas, lo que les permite ofrecer software de alta calidad de manera más rápida y eficiente.