Práctica 5 - PTI

Microservicios: Kubernetes

Arnau Yepes Huguet

Natalia Dai

Grupo 1

07/03/2023

**Índice**

[**1. -Introducción**](#_m1z69butf8uz) **2**

[**2. -Alternativas para resolverla**](#_mnk3vuovdg75) **4**

[**3. -Implementación**](#_a7fg0vxlkbfq) **4**

# **1. -Introducción**

En esta sesión aprenderemos sobre Kubernetes y cómo usarlo, que es una plataforma o un sistema de orquestación de contenedores, para alojar aplicaciones de microservicios que escalan de forma dinámica.

Nos podemos referir a estos microservicios como subaplicaciones o pequeños servicios independientes de los que está dividida una aplicación, que se comunican a través de una API. Puede consistir en varios contenedores más otros recursos, pero en esta práctica solo usaremos uno.

Siguiendo los pasos del enunciado, hemos podido instalar correctamente Minikube (ejecuta un clúster de Kubernetes de un solo nodo localmente), Docker y Kubectl, para tener el entorno configurado, y seguir el tutorial:



Figura 1 - Minikube instalado

Después de comprobar la comunicación entre Kubectl y el clúster, creamos los ficheros “server.js” y “Dockerfile” con los contenidos proporcionados (“Hello World!”), y con los comandos de docker y cURL pudimos construir la imagen del contenedor, que como podemos observar en la siguiente imagen, devuelve el “Hello World!”:

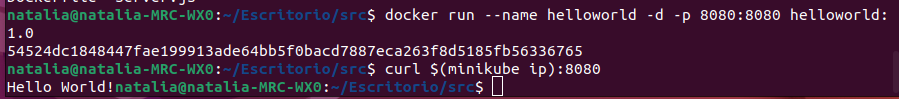


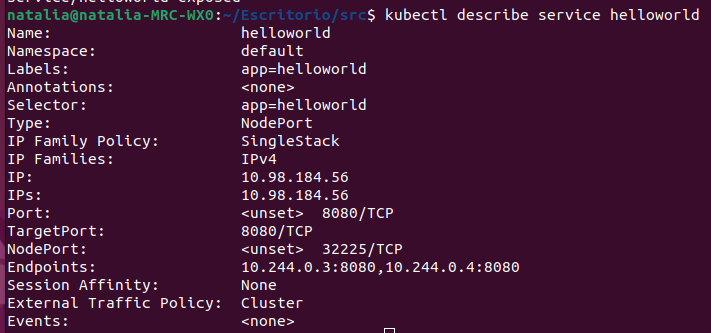
Figura 2 - Prueba “Hello World!”

¿Y qué es un clúster? Un clúster de Kubernetes es un conjunto de máquinas (nodos) que ejecutan aplicaciones en contenedores. A nuestro microservicio “Hello World!” lo podríamos replicar las veces que queramos, que se hacen llamar “Pods”, que se ejecutan dentro de estos nodos.

Por otro lado, puedes crear y actualizar estas instancias a través de una configuración del Kubernetes llamado “Deployment”, o despliegue.

Al hacerlo con nuestro “Hello World!” tendríamos dos “pods”, por ejemplo.

Otra configuración que tiene sería la de “Services”, servicios, que te permite recibir tráfico. Uno para nuestro microservicio sería como se muestran en las siguientes figuras, que con el puerto que podemos ver en la primera imagen (32225) podemos llamar al “Hello World!” en la segunda:





Figuras 3, 4 - Configuración “Services”

Si quisiéramos verlo en el localhost podríamos usar la comanda “kubectl port-forward service/helloworld 7080:8080”:

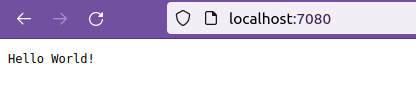


Figura 5 - <http://localhost:7080>

En el caso de que el tráfico incrementara y necesitemos más “pods”, seguiríamos escalando. En este ejemplo, tendríamos ya 4 réplicas:

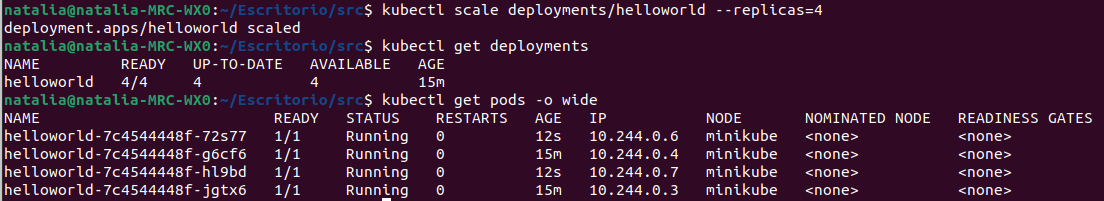


Figura 6 - Ventaja de los microservicios

Después de seguir este tutorial, se nos pide implementar las mismas funcionalidades que hicimos en una práctica pasada (Web APIs) pero esta vez como una aplicación basada en microservicios.

# **2. -Alternativas para resolverla**

Para poder realizar esta práctica podemos haber usado otro ejemplo de CaaS (Container as a Service) como el que proporciona AWS, Azure o Google, y otra base de datos para almacenar la información en vez de usar JSON, como es el caso de MongoDB, como se menciona en el apartado 2.6 del enunciado.

# **3. -Implementación**

Teniendo en cuenta que el código es el mismo que una práctica pasada como se ha mencionado anteriormente, hacemos el mismo procedimiento del tutorial pero con nuestro Dockerfile (debido a que tenemos un subdirectorio para guardar todas las instancias .json, hemos tenido que añadir la línea “RUN mkdir -p Rentals” porque sino no iba) y server.js, llamando al build “rental”, y hacemos dos réplicas:

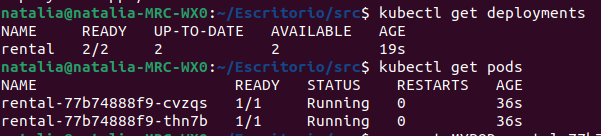


Figura 7 - Pods del “rental”

Siguiendo las comandas descritas en el enunciado, pudimos acceder al <http://localhost:7080/request>, que es donde podemos ver a la lista de los coches que han sido alquilados, siendo la función de List Rentals:

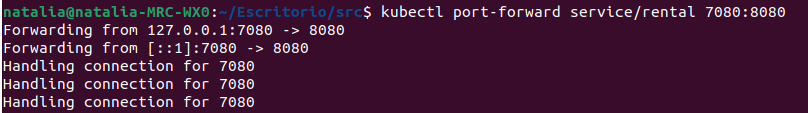


Figura 8 - Acceso a <http://localhost:7080>

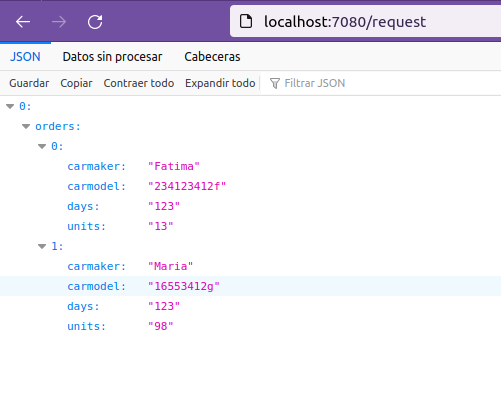


Figura 9 - List Rentals ejemplo

Por otro lado, con la siguiente comanda, podemos ver en funcionamiento la otra función, New Rental, donde se crea correctamente la instancia, que podemos ver en la figura de arriba:

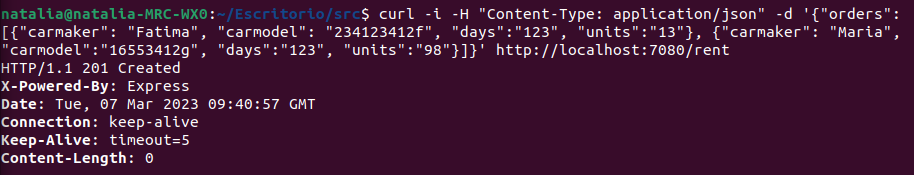


Figura 10, 11 - New Rental ejemplo