KUBERNETES

SISTEMAS DISTRIBUIDOS - PRÁCTICA 5

Lizer Bernad Ferrando - 779035
Lucía Morales Rosa – 81690

---

1 INTRODUCCIÓN

En esta práctica se ha desplegado un servicio de almacenamiento clave-valor basado en Raft desarrollado en prácticas anteriores en Kubernetes.

Para esta práctica se ha hecho uso de una máquina virtual Debian 12.

2 MODIFICACIONES EN EL CODIGO GOLANG

Se han realizado algunas modificaciones en el código de Golang implementado en prácticas anteriores para lograr el correcto funcionamiento del despliegue en Kubernetes.

Para empezar, el cliente ha sido modificado añadiendo el nombre DNS de cada uno de los servidores del sistema. De esta forma, el cliente podrá comunicarse con ellos para encontrar al líder y realizar las peticiones de operaciones que desee.

Se ha decidido que el cliente realice dos operaciones, una de escritura y otra de lectura. Para ello, el cliente supondrá que el nodo 0 es el líder y le enviará la primera operación y, en caso de que no lo sea, lo reintentará con los demás nodos hasta encontrar al líder.

El servidor también ha sido modificado para que tome como primer argumento su nombre DNS completo, en lugar de únicamente su identificador de proceso.

El nombre DNS de los servidores es "raft-i.raft-service.default.svc.cluster.local:6000", donde i representa el identificador del nodo y 6000 es el puerto.

3 EJECUCION EN KUBERNETES

Tal y como se indica en el guion de la práctica, es necesario establecer qué controlador se necesita para la puesta en marcha de los *Pods*. Para los servidores, que deben ser tolerantes a fallos y mantener una identidad estable, se ha decidido emplear el controlador *StatefulSet*, mientras que para el cliente se ha elegido un *Pod* básico ya que no requiere una identidad estable ni tolerancia a fallos.

Por otro lado, se ha creado un servicio "*raft-service*" para permitir la comunicación entre los *Pods* del *StatefulSet*, es decir, de los servidores. Concretamente se ha creado un servicio *headless Service* de forma que cada *Pod* tiene su propia entrada DNS con el formato mostrado a continuación:

$$< nombre - pod > . < nombre - servicio > . < namespace > . svc . cluster . local$$

Para los servidores, además de especificar el uso del controlador *StatefulSet*, se ha indicado que se crearán 3 réplicas del *Pod* paralelamente.

Lizer Bernad Ferrando - **779035**
Lucía Morales Rosa - **816906**

4 FICHEROS .sh

Se han creado varios ficheros ejecutables para automatizar el despliegue del sistema. Para empezar, se ha creado un fichero "cerrarCluster.sh" que elimina el *cluster* de Kubernetes creado junto con los contenedores asociados.

El fichero "expods.sh" limpia los recursos existentes creados en ejecuciones anteriores y los recrea empleando el archivo "pods.yaml".

Finalmente, el fichero "deploy.sh" realiza la configuración y despliegue de Kubernetes con *Kind,* incluyendo la creación del *cluster* con un registro privado, la compilación y construcción de imágenes Docker para los servidores y el cliente, y el despliegue de los *Pods* correspondientes en Kubernetes. Primero, reinicia cualquier *cluster* previo y crea uno nuevo con "cerrarCluster.sh" y "kind-with-registry.sh". Luego, compila el código fuente del servidor y del cliente en Go, genera sus imágenes Docker, las sube al registro local, y finalmente ejecuta los *Pods* en el *cluster* usando el script expods.sh